
Elgg Documentation

Version master

Various

oct. 07, 2022

Table des matières

1	Fonctionnalités	3
2	Exemples	5
3	Poursuivre le lecture	7

Elgg ([pronunciation](#)) est un framework de développement rapide pour construire des applications web sociales. C'est un excellent choix pour construire tout type d'application où des utilisateurs se connectent et partagent des informations.

CHAPITRE 1

Fonctionnalités

- **Une API bien documentée** qui permet aux développeurs d'accélérer leurs nouveaux projets avec une courbe d'apprentissage simple
- **Composer** est le gestionnaire de paquets qui simplifie grandement l'installation et la maintenance du coeur d'Elgg et des plugins
- **Un système flexible de hooks et d'événements** qui permet aux plugins d'étendre et de modifier la majorité des aspects du fonctionnement et du comportement des fonctionnalités des applications
- **Un système de vues extensible** qui permet aux plugins de collaborer sur la couche de présentation des applications et de construire des thèmes personnalisés complexes
- **Un système d'assets statiques mis en cache** qui permet aux thèmes et aux plugins de servir des images, feuilles de style, polices et scripts sans charger le moteur
- **L'authentification des utilisateurs** s'appuie sur des modules d'authentification enfichables, ce qui permet aux applications d'implémenter des protocoles d'authentification personnalisés
- **La Sécurité** est assurée par une validation anti CSRF native, des filtres XSS stricts, des signatures HMAC, et les dernières approches cryptographiques pour le hachage des mots de passe
- **Une API côté client** motorisée par des modules JavaScript asynchrones chargés via RequireJS et un service Ajax natif pour une communication aisée avec le serveur
- **Un système d'Entités flexible** qui permet aux applications de prototyper de nouveaux types de contenus et d'interactions entre utilisateurs
- **Un modèle de données très bien conçu** avec une couche d'API consolidée qui permet aux développeurs de créer facilement les interfaces avec la base de données
- **Une système de contrôle d'accès** qui permet aux applications de construire des politiques d'accès aux contenus granulaires, comme de créer des réseaux privés et des intranets
- **Groupes** - support d'emblée pour des groupes d'utilisateurs
- **Stockage de fichiers** supporté par une API flexible qui permet aux plugins de stocker les fichiers créés par les utilisateurs et de les servir/streamer sans démarrer le moteur
- **Un Service de Notifications** qui permet aux applications d'abonner des utilisateurs aux notifications sur le site ou par email, et d'implémenter l'intégration avec d'autres services tierce-partie
- **Des Services Web RPC** qui peuvent être utilisés pour des intégrations complexes avec des applications externes et des clients mobiles
- **L'Internationalisation** et la localisation des applications Elgg est simple et peut être intégrée avec des services tierce-partie tels que Transifex
- **La communauté Elgg** peut aider sur n'importe quelle problématique qui survienne, et héberge un répertoire

de plus de **1000+ plugins open source**

Sous le capot :

- Elgg est un framework modulaire orienté objet (OOP) qui est dirigé par des services DI
- compatible NGINX ou Apache
- Symfony2 HTTP Foundation gère les requêtes et réponses
- RequireJS gère les modules AMD
- Zend Mail gère les emails sortants
- Filtres XSS htmLawed
- DBAL
- Migrations de base de donnée Phinx
- CSS-Crush pour le pré-traitement CSS
- Imagine pour la manipulation d'image
- Cache persistant avec Memcached et/ou Redis
- Gestion des erreurs avec Monolog

CHAPITRE 2

Exemples

Elgg a été utilisé pour construire **tous types d'applications sociales** :

- réseau sociaux ouverts (similaires à Facebook)
- thématiques (comme la communauté Elgg)
- intranets privés ou professionnels
- rencontres
- éducatifs
- blog d'entreprise

Ceci constitue la documentation officielle du projet **Elgg**.

3.1 Pour démarrer

Découvrez si Elgg est adapté pour votre communauté.

3.1.1 Plugins joints

Elgg est livré avec un jeu de plugins. Ils fournissent les fonctionnalités fondamentales pour votre réseau social.

Blog

Un weblog, ou blog, est sans conteste l'un des éléments fondamental d'ADN de tous les types de sites de réseautage social. La forme la plus simple de publication personnelle, qui permet de publier des notes textuelles dans un ordre antéchronologique. Les commentaires sont également une part importante du blogging, en transformant un acte de publication personnel en une conversation.

Le blog d'Elgg étend ce modèle en fournissant des contrôles d'accès pour chaque entrée et un système de tags transverse aux blogs. Vous pouvez contrôler précisément qui peut voir quel article de blog, et identifier des articles écrits par d'autres personnes sur des sujets similaires. Vous pouvez également voir les entrées écrites par vos contacts (auxquels vous avez accès).

Voir aussi :

[Blogging sur Wikipedia](#)

Tableau de bord

Le tableau de bord est livré à la fois avec la version complète et noyau seulement d'Elgg. C'est le portail des utilisateurs vers des activités qui leur sont importantes à la fois à l'intérieur du site et depuis des sources externes. En utilisant la puissante API widgets d'Elgg, il est possible de construire des widgets qui retirent du contenu pertinent depuis l'intérieur d'un site Elgg, ainsi que de récupérer des informations depuis des sources tierce-partie telles que Twitter ou Flickr (dès lors que ces widgets existent). Le tableau de bord des utilisateurs n'est pas la même chose que leur profil : tandis que le profil est destiné à être consulté par les autres, le tableau de bord est un espace strictement personnel que les utilisateurs utilisent pour leurs propres besoins.



Fig. 1 – Un tableau de bord Elgg typique

Diagnostics

Pour l'utilisateur techniquement averti, les diagnostics système permettent d'évaluer rapidement l'environnement serveur, le code d'Elgg, et les plugins d'une installation Elgg. Les diagnostics sont un plugin système du noyau qui est activé par défaut avec Elgg. Pour télécharger le fichier de diagnostics, suivez les étapes ci-dessous. Le fichier est un vidage de tous types d'informations utiles.

Pour l'utiliser :

- Connectez-vous en tant qu'administrateur
- Rendez-vous dans Administration -> Administrer -> Utilitaires ->Diagnostics Système
- Cliquez sur "Télécharger"

Contenu du fichier de vidage des diagnostics système :

- Liste de tous les fichiers d'Elgg ainsi qu'un hash pour chaque fichier
- Liste de tous les plugins
- Superglobales PHP
- Paramètres PHP
- Paramètres Apache
- **Valeurs de Elgg CONFIG**
 - chaînes de traduction
 - paramètres du site
 - paramètres de la base de données
 - hooks des plugins
 - actions
 - vues
 - gestionnaires de pages
 - beaucoup plus

Répertoire de fichiers

Le répertoire de fichiers permet à des membres de charger n'importe quel type de fichier. Comme avec tout dans un système Elgg, vous pouvez aisément filtrer les fichiers par tags et restreindre l'accès de sorte qu'ils soient visibles seulement par les personnes que vous souhaitez. Chaque fichier peut également avoir des commentaires associés.

Il existe plusieurs types d'usages différents pour cette fonctionnalité

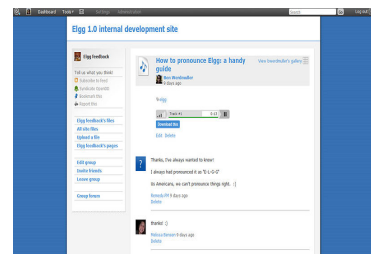


Fig. 2 – Un fichier dans un répertoire de fichiers Elgg

Galerie de photo

Quand un utilisateur charge des photographies ou d'autres images, elles sont automatiquement rassemblées dans une galerie de photos Elgg dans laquelle il est possible de naviguer. Les utilisateurs peuvent aussi voir les photos que leurs contacts ont chargées, ou voir des images attachées à un groupe. Une version plus grande de la photo apparaît en cliquant sur l'un des fichiers.

Podcasts

Un répertoire de fichiers Elgg est doublé automatiquement par un flux RSS, de sorte que vous pouvez vous abonner à du nouveau contenu audio en utilisant des programmes tels que iTunes.

Contenus spéciaux

Il est possible pour d'autres plugins d'ajouter de nouveaux lecteurs et visionneuses pour différents types de contenus. L'auteur d'un plugin peut par exemple intégrer de cette manière une visionneuse pour des documents Word.

Note pour les développeurs

Pour ajouter un lecteur pour un nouveau type de contenu, créez un plugin avec des vues de la forme `file/specialcontent/mime/type`. Par exemple, pour créer une visionneuse pour des documents Word, vous pouvez créer une vue nommée `file/specialcontent/application/msword`, puisque `application/msword` est le type MIME pour les documents Word. A l'intérieur de cette vue, la version `ElggEntity` du fichier sera référencée en tant que `$vars['entity']`. Dès lors, l'URL du fichier téléchargeable est :

```
echo $vars['entity']->getDownloadURL();
```

En utilisant cela, il devrait être possible de développer la plupart des types de visionneuses embarquables.

Contacts

En tant que framework de réseau social, Elgg gère les relations entre membres.

Par défaut tout membre peut se mettre en contact avec n'importe quel autre membre : cela revient à suivre l'activité de l'autre membre.

Après avoir activé les demandes de contact en tant que fonctionnalité du plugin Friends, lorsque l'utilisateur A veut être en contact avec l'utilisateur B, l'utilisateur B doit approuver la demande. Lors de l'approbation, l'utilisateur A sera en contact avec l'utilisateur B et l'utilisateur B sera en contact avec l'utilisateur A.

Groupes

Une fois que vous avez trouvé d'autres personnes avec des centres d'intérêt similaires - ou peut-être que vous faites partie d'un groupe de recherche ou d'un cours - vous pourriez souhaiter disposer d'un environnement plus structuré pour partager du contenu et discuter d'idées. C'est là qu'interviennent les puissantes fonctionnalités d'Elgg en terme de construction de groupes. Vous pouvez créer et modérer autant de groupes que vous le désirez.

- Vous pouvez conserver toute l'activité du groupe privée pour le groupe, ou utiliser l'option "rendre public" pour disséminer ses travaux auprès d'un public plus large.
- Chaque groupe produit des flux RSS granulaires, de sorte qu'il est aisé de suivre les développements des groupes
- Chaque groupe dispose de sa propre URL et page de profil
- Chaque groupe dispose d'un répertoire de fichiers *Répertoire de fichiers*, d'un forum, de pages et d'un panneau d'affichage

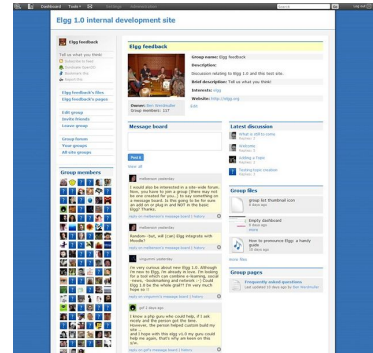


Fig. 3 – Un profil de groupe typique

Messageboard

Le panneau d'affichage - similaire au "Mur" de Facebook ou à un mur de commentaires dans d'autres réseaux - est un plugin qui permet aux membres de placer un widget panneau d'affichage sur leur profil. Les autres membres peuvent alors publier des messages qui vont apparaître sur ce panneau d'affichage. Vous pouvez ensuite répondre directement à n'importe quel message et voir l'historique entre vous et la personne qui a publié le message.

Messages

Des messages privés peuvent être envoyés aux membres en cliquant sur leur avatar ou le lien vers leur profil. Puis, en utilisant l'*éditeur WYSIWYG* natif, il est possible de formater le message. Chaque membre dispose de sa propre boîte de réception et d'envoi. Il est possible d'être notifié par email des nouveaux messages.

Quand les membres s'identifient, ils sont notifiés de tout nouveau message par le mécanisme de notification des messages dans leur barre d'outil supérieure.

Pages

Le plugin pages permet d'enregistrer et de conserver d'une manière organisée et hiérarchisée des pages de texte, et d'en restreindre à la fois les accès en lecture et en écriture. Ceci signifie que vous pouvez créer collaborativement un ensemble de documents avec une liste de personnes choisies, participer à un processus d'écriture au sein d'un groupe formalisé, ou simplement utiliser cette fonctionnalité pour créer un document que vous seul pouvez voir, jusqu'à ce que vous choisissiez de le partager une fois terminé. Le menu de navigation simple vous permet de voir la structure de l'ensemble de documents depuis n'importe quelle page. Vous pouvez créer autant de ces structures que vous le souhaitez ; chaque page individuelle dispose de ses propres contrôles d'accès, de sorte que vous pouvez révéler des portions de la structure tout en conservant les autres invisibles. Dans la ligne des autres éléments dans Elgg, vous pouvez aisément ajouter des commentaires sur une page, ou rechercher des pages par tag.



Fig. 5 – Notification des messages

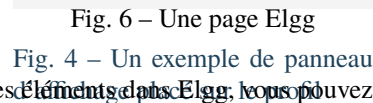


Fig. 6 – Une page Elgg

Fig. 4 – Un exemple de panneau d'affichage dans Elgg, vous pouvez

Usage

Les pages expriment vraiment leur potentiel dans deux domaines, d'abord comme moyen pour les utilisateurs de construire des choses telles qu'un CV, un portfolio, une documentation réflexive, et autres constructions de ce type. La seconde est dans le domaine de la collaboration, tout particulièrement dans le cadre d'un groupe. Avec les puissants outils de contrôle d'accès à la fois en lecture et en écriture, ce plugin est idéal pour la création collaborative de documents.

Note : Les développeurs devraient tenir compte du fait qu'il existe en fait 2 types de pages :

1. Pages racine (avec le sous-type `page_top`)
2. Pages normales (avec le sous-type `page`)

Profil

Le plugin profile est livré à la fois avec la version complète et noyau seulement d'Elgg. L'objectif est qu'il puisse être désactivé et remplacé par un autre plugin de profil si vous le souhaitez. Il fournit un certain nombre d'éléments de fonctionnalités que beaucoup considèrent comme fondamentales pour le concept d'un site de réseautage social, et est unique parmi les plugins parce que l'icône de profil qu'il définit est référencée comme un standard à travers l'ensemble du système.

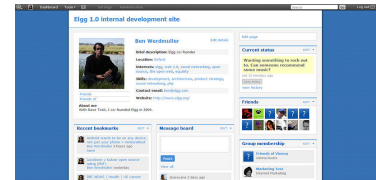


Fig. 7 – Un profil Elgg

Informations utilisateur

Ceci fournit des informations à propos d'un utilisateur, qui est configurable à partir du fichier `start.php` du plugin. Vous pouvez changer les champs de profil disponibles depuis le panneau d'administration. Chaque champ de profil dispose de son propre niveau d'accès, de sorte que les membres puissent choisir exactement qui peut voir quel élément précis. Certains des champs contiennent des tags (par exemple *compétences*) : restreindre l'accès à ce champ va également restreindre les personnes qui peuvent vous trouver via ce tag.

Avatar utilisateur

L'avatar utilisateur représente un utilisateur (ou un groupe) à travers le site. Par défaut, l'avatar intègre un menu contextuel sensible qui vous permet d'effectuer des actions sur l'utilisateur dès lors que vous voyez son avatar. Par exemple vous pouvez l'ajouter comme contact, lui envoyer un message privé, et plus. Chaque plugin peut ajouter des entrées à ce menu contextuel, de sorte que son contenu final va beaucoup dépendre des fonctionnalités actives dans le site Elgg actuel.

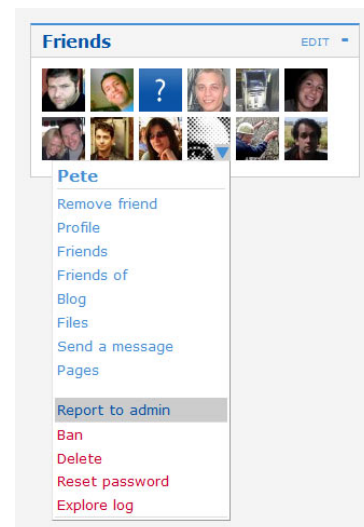


Fig. 8 – Le menu contextuel Elgg

Notes pour les développeurs

Utiliser une icône de profil différente Pour remplacer l’icône de profil, ou fournir plus de contenu, étendre la vue `icon/user/default`.

Ajouter des éléments au menu contextuel Le menu contextuel peut être étendu en enregistrant un *hook de plugin* pour “register” “menu :user_hover”, les sections suivantes ont une signification spéciale :

- **default** pour les liens non actifs (par ex. pour lire un blog)
 - **admin** pour les liens accessibles seulement par les administrateurs
- Dans tous les cas, l’utilisateur en question sera passé en tant que `$params['entity']`.

Le Fil

Le plugin de câble d’Elgg « The Wire » (Le Fil) est un plugin de micro-blogging à la Twitter, qui permet aux utilisateurs de publier des notes vers le fil.

Validation des comptes utilisateur via l’email

Le plugin `uservalidationbyemail` ajoute une étape au processus d’inscription des utilisateurs. Après que l’utilisateur se soit enregistré sur le site, un email est envoyé à son adresse email afin de valider que cette adresse email appartient bien à l’utilisateur. L’utilisateur ne pourra se connecter qu’après avoir cliqué sur le lien de vérification contenu dans l’email.

Déroulement pour l’utilisateur

1. L’utilisateur crée un compte en se rendant sur la page d’inscription de votre site
2. Une fois le compte créé l’utilisateur arrive sur une page qui indique de vérifier la présence de l’email de validation sa boîte de messagerie
3. Dans l’email de validation, un lien permet de confirmer son adresse email
4. Après avoir cliqué sur le lien, le compte est validé
5. Si possible l’utilisateur est connecté

Si l’utilisateur tente de s’identifier avant d’avoir validé son compte, une erreur indique que l’utilisateur doit d’abord vérifier sa boîte mail. De plus, un nouveau mail de validation est envoyé.

Options pour les administrateurs du site

Un administrateur du site peut effectuer certaines actions sur les comptes non validés. Une liste des membres non validés se trouve dans Administration -> Utilisateurs -> Non validés. L’administrateur peut manuellement valider ou supprimer l’utilisateur. Il existe également une option pour renvoyer l’email de validation.

Les plugins suivants sont également livrés avec Elgg, mais ne sont pas (encore) documentés

- `activity` (activité)
- `signets`
- `ckeditor`
- `custom_index` (index personnalisé)
- `developers` (développeurs)
- `discussions`

- embed
- externalpages
- friends_collections
- garbagecollector
- invitefriends
- likes
- members
- notifications
- reportedcontent
- search
- site_notifications
- system_log
- tagcloud
- web_services

3.1.2 License

MIT ou GPLv2

Un package Elgg complet comprenant le framework et un noyau de plugins est disponible sous la version 2 de la licence [GNU General Public License](#) (GPLv2). Nous distribuons également le framework (sans les plugins) sous la licence MIT.

FAQ

Les réponses suivantes vous sont proposées à titre informatif; elles ne constituent pas un conseil légal. Consultez un juriste si vous voulez être sûr(e) des réponses à ces questions. La Fondation Elgg ne peut pas être tenue pour responsable des décisions que vous prendrez sur la base de ce que vous lisez sur cette page.

Pour les questions qui n'ont pas réponse ici, veuillez vous référer à la FAQ officielle pour GPLv2 : [official FAQ for the GPLv2](#).

Combien coûte Elgg ?

Elgg peut être téléchargé, installé et utilisé gratuitement. Si vous souhaitez faire un don, nous apprécions d'avoir des partenaires financiers [financial supporters](#) !

Puis-je enlever les liens / le branding Elgg ?

Oui.

Puis-je modifier le code source ?

Oui, mais en général nous vous recommandons de faire vos modifications sous forme de plugins de sorte que lorsque une nouvelle version d'Elgg est publiée le processus de mise à niveau reste aussi simple que possible.

Puis-je demander des frais d'adhésion à mes utilisateurs ?

Oui.

Si je modifie Elgg, dois-je rendre mes modifications disponibles ?

Non, si vous utilisez Elgg pour fournir un service, vous n'avez pas à rendre les sources disponibles. Si vous distribuez une version modifiée d'Elgg, alors vous devez inclure le code sources des modifications.

Si j'utilise Elgg pour héberger un réseau, est-ce que la Fondation Elgg a quelques droits que ce soit sur mon réseau ?

Non.

Quelle est la différence entre la version MIT et la version GPL ?

Les plugins ne sont pas inclus avec la version MIT.

Vous pouvez distribuer un produit commercial construit avec Elgg en utilisant la version MIT sans devoir rendre disponibles vos modifications.

Avec la version sous licence GPL, vous devez rendre vos modifications du framework publiques si vous redistribuez le framework.

Pourquoi certains plugins sont-ils absents de la version MIT ?

Les plugins ont été développés sous licence GPL, et ne peuvent donc pas être diffusés sous une licence MIT. De plus, certains plugins font appel à des dépendances externes qui ne sont pas compatibles avec la licence MIT.

Ai-je le droit de distribuer un plugin pour Elgg sous une licence commerciale ?

Nous pensons que vous le pouvez, puisque les plugins ne dépendent habituellement que du cœur du framework (core) et que le framework est disponible sous la licence MIT. Ceci dit, nous vous conseillons vraiment de consulter un juriste sur ce point particulier si vous voulez être absolument sûr(e).

Notez que les plugins distribués via la communauté doivent être publiés sous une licence compatible GPLv2. Ils n'ont pas à être publiés sous licence GPLv2, seulement sous une licence compatible (comme la MIT).

Pouvons-nous construire notre propre outil utilisant Elgg et vendre cet outil à nos clients ?

Oui, mais dans ce cas vos clients seront libres de redistribuer cet outil sous les termes de la licence GPLv2.

3.1.3 Installation

Ayez votre propre instance d'Elgg opérationnelle en un rien de temps.

Contenu

- *Pré-requis*
- *Vue d'ensemble*
- *Autres configurations*
- *Dépannage*

Pré-requis

- MySQL 5.5.3+ (5.0.0+ si vous mettez à niveau une installation existante)
- PHP 7.2+ avec les extensions suivantes :
 - GD (pour les opérations graphiques)
 - PDO (pour la connexion à la base de données)
 - JSON (pour les réponses AJAX, etc.)
 - XML (pour lire fichiers manifest des plugins, etc.)
 - [Multibyte String support](#) (pour i18n)
 - La configuration appropriée et la capacité d'envoyer des emails via un MTA (Mail Transport Agent)
- Serveur web avec le support de la réécriture d'URL (URL rewriting)

Le support officiel est fourni pour les configurations suivantes :

- **Serveur Apache**
 - Apache avec le module [rewrite module](#) activé
 - PHP exécuté comme un module Apache
- **Serveur Nginx**
 - Nginx avec PHP-FPM utilisant FastCGI

Par « support officiel », nous entendons que :

- La plupart des développements et des tests sont effectués avec ces configurations
- Une bonne part de la documentation a été écrite en partant du principe qu'Apache ou Nginx est utilisé
- La priorité sur les rapports de bug est donnée aux utilisateurs d'Apache et Nginx si le bug est propre au serveur web (mais ces cas sont rares).

Note : Si vous utilisez RHEL, CentOS ou toute autre distribution avec SELinux activé, vous devrez vous assurer que les contextes et les autorisations appropriés sont configurés :

- Donnez à la racine de votre projet, à votre répertoire de données et à tous leurs enfants le contexte `httpd_sys_rw_content_t`
- Activez les booléens `httpd_can_network_connect` et `httpd_can_network_connect_db`

Politique de support des navigateurs

Les branches de fonctionnalités supportent les 2 dernières versions de tous les principaux navigateurs disponibles au moment de la première publication d'une version stable pour cette branche.

Les versions de correction de bugs ne modifient pas le support des navigateurs, même si une nouvelle version du navigateur a été publiée depuis.

Les principaux navigateurs signifie ici les navigateurs suivants, ainsi que leurs homologues mobiles :

- Navigateur Android
- Chrome
- Firefox
- IE
- Safari

« Support » peut signifier que nous prenons avantage des technologies nouvelles et non implémentées, mais fournissons un polyfill JavaScript pour les navigateurs qui en ont besoin.

Il se peut qu'Elgg fonctionne avec des navigateurs non supportés, cependant la compatibilité peut être perdue à tout moment, y compris lors de la publication d'une correction de bug.

Vue d'ensemble

Chargez Elgg

Avec Composer (recommandé si vous êtes à l'aise avec la ligne de commande) :

```
composer self-update
composer create-project elgg/starter-project:dev-master ./path/to/project/root
cd ./path/to/project/root
composer install
composer install # 2nd call is currently required
vendor/bin/elgg-cli install # follow the questions to provide installation details
```

A partir du fichier ZIP (recommandé si vous n'êtes pas à l'aise avec la ligne de commande) :

- Télécharger la [dernière version d'Elgg](#)
- Chargez le fichier ZPI avec un client FTP sur votre serveur
- Dézippez les fichiers dans la racine web de votre domaine.

Créer un fichier pour les données

Elgg a besoin d'un répertoire particulier pour stocker les fichiers chargés, y compris les images du profil et les photos. Vous devez créer ce répertoire.

Attention : Pour des raisons de sécurité, ce dossier **DOIT** être conservé hors de la racine de vos documents (DocumentRoot). Si vous l'avez créé dans le dossier /www/ ou /public_html/, ce n'est pas la bonne manière de faire.

Une fois que ce répertoire a été créé, vous devez vous assurer que le serveur web sur lequel tourne Elgg a le droit d'écrire et de créer des répertoires à l'intérieur. Ceci ne devrait pas être un problème sur les serveurs basés sur Windows, mais si votre serveur utilise Linux, Mac OS X ou une variante d'UNIX, vous aurez besoin de [définir les droits d'accès du répertoire](#).

Si vous utilisez un client FTP graphique pour charger les fichiers, vous pouvez habituellement définir les droits en faisant un clic droit sur le dossier et en sélectionnant « propriétés » ou « Informations ».

Note : Les répertoires doivent pouvoir être lus et écrits. Les permissions suggérées dépendent de votre serveur et de la configuration des utilisateurs. Si le répertoire de données a pour propriétaire l'utilisateur du serveur web, les droits d'accès recommandés sont 750.

Avertissement : Définir les droits de votre répertoire de données à 777 va fonctionner, mais c'est dangereux et n'est pas recommandé. Si vous hésitez sur la bonne configuration à choisir pour les permissions, contactez votre hébergeur pour plus d'informations.

Créer une base de données MySQL

En utilisant l'outil d'administration de base de données de votre choix (si vous ne savez pas lequel, demandez à votre administrateur système), créez une nouvelle base de données MySQL pour Elgg. Vous pouvez créer une base de données MySQL avec n'importe lequel des outils suivants :

Assurez-vous d'ajouter un utilisateur à la base de données avec tous les privilèges et notez le nom de la base de données, l'identifiant et le mot de passe. Vous aurez besoin de ces informations pour installer Elgg.

Mettre en place le Cron

Elgg utilise des requêtes paramétrées pour effectuer des tâches de fond telles qu'envoyer des notifications ou effectuer des travaux de nettoyage de la base de données. Vous devez configurer le [cron](#) pour pouvoir utiliser ce type de fonctionnalités.

Visiter votre site Elgg

Une fois ces étapes terminées, visitez votre site Elgg dans votre navigateur web. A partir de là, Elgg vous guidera à travers le reste du processus d'installation. Le premier compte que vous créerez à la fin de l'installation sera le compte administrateur.

Une note sur settings.php et .htaccess

L'installateur d'Elgg va essayer de créer deux fichiers pour vous :

- `elgg-config/settings.php`, qui contient la configuration de votre environnement local pour votre installation
- `.htaccess`, qui permet à Elgg de générer des URLs dynamiques

Si ces fichiers ne peuvent pas être générés automatiquement, par exemple parce que le serveur web n'a pas les droits d'accès en écriture dans les répertoires, Elgg vous dira comment les créer. Vous pouvez aussi modifier temporairement les droits d'accès sur la racine du répertoire et le répertoire engine. Définissez les droits d'accès sur ces deux répertoires de sorte que le serveur web puisse écrire ces deux fichiers, terminez le processus d'installation, et modifiez à nouveau les droits d'accès pour rétablir les droits d'origine. Si, pour quelque raison que ce soit, ceci ne fonctionne pas, vous devrez :

- Depuis `elgg-config/`, copiez `settings.example.php` vers `settings.php`, puis ouvrez-le dans un éditeur de texte et complétez les informations de la base de données
- Sur le serveur Apache, copiez `install/config/htaccess.dist` vers `.htaccess`

- Sur un serveur Nginx copiez `install/config/nginx.dist` vers `/etc/nginx/sites-enabled` et modifiez son contenu

Autres configurations

- Cloud9
- Homestead
- EasyPHP
- IIS
- MAMP
- MariaDB
- Nginx
- Ubuntu
- Hôtes virtuels
- XAMPP

Dépannage

Au secours ! J'ai des soucis pour installer Elgg

D'abord :

- Re-vérifiez que votre serveur répond bien aux pré-requis pour Elgg.
- Suivez si besoin les instructions spécifiques à un environnement
- Avez-vous vérifié que `mod_rewrite` est bien chargé ?
- Est-ce que le module `mysql` de apache est bien chargé ?

Conservez des notes sur ce que vous faites pour résoudre les problèmes d'installation. Parfois la modification d'un paramètre ou d'un fichier pour essayer de résoudre un problème peut être à l'origine d'un autre problème plus tard. Si vous devez recommencer depuis le début, supprimez simplement tous les fichiers, videz votre base de données, et commencez à nouveau.

Je ne peux pas enregistrer mon fichier de configuration sur une installation (j'ai une erreur 404 lors de l'enregistrement du fichier de configuration)

Elgg dépend de l'extension Apache `mod_rewrite` pour simuler certaines URLs. Par exemple à chaque fois que vous effectuez une action dans Elgg, or lorsque vous visitez la page de profil d'un utilisateur, l'URL est traduite par le serveur en quelque chose qu'Elgg comprend en interne. Ceci est fait en utilisant des règles définies dans un fichier `.htaccess`, qui est le moyen standard d'Apache pour définir des éléments de configuration supplémentaires pour un site.

Cette erreur suggère que les règles de `mod_rewrite` ne sont pas traitées correctement. Ceci peut arriver pour différentes raisons. Si vous n'êtes pas à l'aise pour mettre en œuvre les solutions indiquées ci-dessous, nous vous recommandons vivement de contacter votre administrateur système ou le support technique et de leur faire suivre cette page.

Le `.htaccess`, s'il n'est pas créé automatiquement (cela se produit quand vous avez un problème avec `mod_rewrite`), vous pouvez le créer en renommant le fichier `install/config/htaccess.dist` distribué avec Elgg en `.htaccess`. Par ailleurs, si vous trouvez un fichier `.htaccess` dans le répertoire d'installation, mais avez toujours une erreur 404, vérifiez que le contenu de `.htaccess` est identique à celui de `install/config/htaccess.dist`.

`mod_rewrite` n'est pas installé.

Contrôlez votre `httpd.conf` pour vérifier que ce module est bien chargé par Apache. Vous pouvez avoir besoin de redémarrer Apache pour qu'il tienne compte de tout changement de configuration. Vous pouvez également utiliser [PHP info](#) pour contrôler que le module est chargé.

Les règles définies dans `.htaccess` ne sont pas respectées.

Dans les paramètres de configuration de votre hôte virtuel (qui peut être intégrée dans `httpd.conf`), modifiez la paramètre `AllowOverride` pour qu'il ressemble à :

```
AllowOverride all
```

Ceci va indiquer à Apache de prendre les règles de `mod_rewrite` depuis `.htaccess`.

Elgg n'est pas installé à la racine de votre répertoire web (« **Elgg is not installed in the root of your web directory** ») (par ex. : `http://example.org/elgg/` au lieu de `http://example.org/`)

Le script d'installation me redirige vers « action » alors que cela devrait être vers « actions »

Il s'agit d'un problème avec votre configuration de `mod_rewrite`.

Attention : NE CHANGEZ PAS, RÉPÉTEZ, NE CHANGEZ PAS quelque nom de répertoire que ce soit !

J'ai installé Elgg dans un sous-répertoire et mon action d'installation ne fonctionne pas !

Si vous installez Elgg de sorte qu'il soit accessible avec une adresse comme `http://example.org/monsite/` plutôt que `http://example.org/`, il existe une faible possibilité que les règles de réécriture dans `.htaccess` ne soient pas traitées correctement. Ceci est généralement lié à l'utilisation d'un alias dans Apache. Vous pouvez avoir besoin d'indiquer à `mod_rewrite` où se situe votre installation Elgg.

- Ouvrez `.htaccess` dans un éditeur de texte
- Lorsqu'on vous y invite, ajoutez une ligne comme `RewriteBase /chemin/vers/votre/installation/elgg/` (N'oubliez pas le slash final)
- Enregistrez le fichier et rafraîchissez votre navigateur.

Veuillez noter que le chemin que vous utilisez est le chemin **web**, moins l'hôte.

Par exemple, si votre installation Elgg s'affiche sur `http://example.org/elgg/`, vous devriez définir la base comme ceci :

```
RewriteBase /elgg/
```

Veuillez noter qu'installer dans un sous-répertoire ne nécessite pas d'utiliser `RewriteBase`. Il y a seulement quelques rares circonstances dans lesquelles c'est imposé par la configuration du serveur.

J'ai tout fait ! `mod_rewrite` fonctionne correctement, mais j'ai toujours l'erreur 404

Il y a peut-être un problème avec le fichier `.htaccess`. Parfois la routine d'installation d'Elgg n'arrive pas à en créer un ni à vous le signaler. Si vous en êtes à ce stade et avez essayé tout ce qui est indiqué ci-dessus :

- vérifiez qu'il s'agit bien du `.htaccess` créé par Elgg (et pas d'un fichier d'exemple fourni par le fournisseur du serveur)
- s'il ne s'agit pas du fichier `htaccess` fourni par Elgg, utilisez `htaccess_dist` (renommez-le en `.htaccess`)

J'ai un message d'erreur indiquant que le test de réécriture a échoué après la page de vérification des pré-requis

J'ai les messages suivants après l'étape de vérification des pré-requis (étape 2) de l'installation :

Nous pensons que votre serveur utilise un serveur web Apache.

Le test de réécriture a échoué et la cause la plus probable est que AllowOverride n'est pas définie à All pour le répertoire d'Elgg. Ceci empêche Apache de traiter le fichier .htaccess qui contient les règles de réécriture.

Une cause moins probable est qu'Apache est configuré avec un alais pour votre répertoire Elgg et que vous deviez définir RewriteBase dans votre .htaccess. Vous trouverez des instructions complémentaires dans le fichier .htaccess de votre répertoire Elgg.

Après cette erreur, toute interaction avec l'interface web produit une erreur 500 (Erreur interne du serveur - Internal Server Error)

Ceci est probablement causé par le non-chargement du module de filtre en dé-commentant la ligne

```
#LoadModule filter_module modules/mod_filter.so
```

ligne dans le fichier « httpd.conf ».

le fichier de journal d'Apache « error.log » va contenir une entrée similaire à :

```
... .htaccess : Invalid command "AddOutputFilterByType", perhaps misspelled or defined by a module not included in the server configuration (Commande "AddOutputFilterByType" invalide, peut-être mal écrite ou définie par un module qui n'est pas inclus dans la configuration du serveur)
```

Il y a une page blanche après que j'ai soumis ma configuration de base de données

Vérifiez que le module d'Apache mysql est installé et est bien chargé.

J'obtiens une erreur 404 avec une URL très longue

Si vous voyez une erreur 404 pendant l'installation ou lors de la création du premier compte utilisateur, avec une URL telle que : `http://example.com/homepages/26/d147515119/htdocs/elgg/action/register` ceci signifie que l'URL de votre site est incorrecte dans la table `sites_entity` table de votre base de données. Cette valeur a été définie par vous lors de la deuxième étape de l'installation. Elgg essaie de deviner la bonne valeur mais a des difficultés avec les hébergements mutualisés. Utilisez phpMyAdmin pour modifier cette valeur avec la bonne URL de base du site.

J'ai des difficultés pour définir le chemin vers le répertoire de données

Ceci est fortement spécifique au serveur utilisé aussi il est difficile de donner des conseils spécifiques. Si vous avez créé un répertoire pour les données, vérifiez que votre serveur HTTP arrive bien à y accéder. Le moyen le plus simple (mais le moins sécurisé) d'y parvenir est de lui donner les droits d'accès 777. Il est largement préférable de donner au serveur web la propriété du répertoire et de limiter les droits d'accès.

Avertissement : Définir les permissions du répertoire à 777 permet à **L'INTÉGRALITÉ** d'Internet de placer des fichiers dans la structure de votre répertoire et ainsi d'infecter votre serveur avec des malwares et autres virus. Définir les permissions à 750 devrait être plus que suffisant.

L'origine la plus probable de ce problème est que PHP est configuré pour interdire l'accès à la plupart des répertoires qui utilisent [open_basedir](#). Vous pouvez souhaiter vérifier ce point avec votre fournisseur d'hébergement.

Assurez-vous que le chemin est correct et se termine par un /. Vous pouvez vérifier le chemin dans la table config de votre base de données.

Si vous n'avez qu'un accès FTP à votre serveur et avez créé un répertoire mais n'en connaissez pas le chemin, vous devriez pouvoir le trouver à partir du chemin `www` défini dans la table config de votre base de données. A ce stade il est recommandé de demander de l'aide à l'équipe support de votre fournisseur d'hébergement.

Je ne peux pas valider mon compte admin car je n'ai pas de serveur d'email !

Bien qu'il soit exact que les comptes utilisateur normaux (à l'exception de ceux créés depuis le panneau d'administration) nécessitent que leur adresse email soit authentifiée avant de pouvoir se connecter, ceci n'est pas nécessaire pour le compte administrateur.

Une fois que vous avez créé votre premier compte utilisateur vous pouvez vous connecter en utilisant les accès que vous avez fourni !

J'ai essayé toutes ces suggestions et je n'arrive toujours pas à installer Elgg

Il est possible que quelque chose d'autre ait été cassé lors du débogage de l'installation. Essayez de faire une nouvelle installation propre :

- supprimez votre base de données Elgg
- supprimez votre répertoire de données
- supprimez les fichiers source d'Elgg
- recommencez l'installation

SI cette méthode ne fonctionne pas, demandez de l'aide à la [communauté Elgg](#). Pensez à indiquer quelle version d'Elgg vous essayez d'installer, à fournir des détails sur le serveur et la plateforme utilisés, ainsi que tous les messages d'erreur que vous pouvez avoir reçus, y compris ceux du journal d'erreur de votre serveur.

3.1.4 Vue d'ensemble pour les développeurs

Cette page constitue une rapide introduction à Elgg pour les développeurs. Elle couvre les approches de base pour travailler avec Elgg en tant que framework, et mentionne certains des termes et des technologies utilisées.

Voyez [Guides de développement](#) pour des tutoriels ou [Docs de conception](#) pour une discussion approfondie sur le design.

Base de données et persistance

Elgg utilise MySQL 5.5 ou supérieur pour la persistance des données, et les données de la base sont transcrites (mapées) dans des Entités (une représentation d'une unité d'information atomique) et des Extenseurs (Extenders) (des informations additionnelles et des descriptions à propos des Entités). Elgg supporte des informations additionnelles telles que des relations entre des Entités, des flux d'activités, et divers types de réglages.

Plugins

Les Plugins modifient le comportement ou l'apparence d'Elgg en surchargeant les vues, ou en gérant des événements et des hooks des plugins. Toutes les modifications d'un site Elgg devraient être implémentées via des plugins pour garantir que la mise à niveau du coeur reste simple.

Actions

Les actions sont le premier moyen pour les utilisateurs d'interagir avec un site Elgg. Les actions sont définies par les plugins.

Événements et hooks du plugin

Les Événements et les Hooks des plugins sont utilisés dans les plugins Elgg pour interagir avec le moteur d'Elgg dans certaines circonstances. Les Événements et les Hooks sont activés à des moments stratégiques lors des processus de démarrage et d'exécution d'Elgg, et permettent aux plugins de modifier ou d'annuler le comportement par défaut.

Vues

Les vues sont la première couche de présentation pour Elgg. Les vues peuvent être surchargées ou étendues par les Plugins. Les vues sont des catégories d'un type de vue (Viewtype), qui définit quel type de sortie devrait être généré par la vue.

JavaScript

Elgg utilise un système JavaScript compatible AMD fourni par RequireJs. jQuery, jQuery UI, jQuery Form, et jQuery UI Autocomplete sont livrés avec Elgg.

Les plugins peuvent charger leurs propres bibliothèques JS.

Internationalisation

L'interface d'Elgg supporte de multiples langues, et utilise [Transifex](#) pour la traduction.

Mise en cache

Elgg utilise deux caches pour améliorer les performances : un cache système et SimpleCache.

Bibliothèques tierce-partie

L'utilisation de bibliothèques tierce-partie est gérée par le gestionnaire de dépendances [Composer](#). jQuery, RequireJs ou Zend mail sont des exemples de bibliothèques tierce-partie.

Pour une liste complète des dépendances d'Elgg, consultez la page [Packagist](#) d'Elgg.

Génération de données de test

Elgg fournit un mécanisme de génération de données (ensemencement de base de données, ou database seeding) pour remplir la base de données avec des entités à des fins de test.

Vous pouvez exécuter les commandes suivantes pour générer des données de test et les supprimer de la base de données.

```
# seed the database
vendor/bin/elgg-cli database:seed

# unseed the database
vendor/bin/elgg-cli database:unseed
```

Les plugins peuvent enregistrer leurs propres graines via le hook 'seeds', 'database'. Le gestionnaire doit retourner le nom de la classe de la graine, qui doit étendre la classe \Elgg\Database\Seeder\Seed.

3.1.5 Elgg CLI

Contenu

- *elgg-cli outils en ligne de commande*
- *Commandes disponibles*
- *Ajouter des commandes personnalisées*

elgg-cli outils en ligne de commande

Selon la manière dont vous avez installé Elgg et la configuration de votre serveur vous pouvez accéder aux binaires de elgg-cli de l'une des manières suivantes depuis la racine de votre installation Elgg :

```
php ./elgg-cli list
./elgg-cli list
php ./vendor/bin/elgg-cli list
./vendor/bin/elgg-cli list
```

Note : Soyez informé que lors de l'utilisation elgg-cli il peut être nécessaire d'exécuter la commande avec le même utilisateur que le serveur Web pour éviter les problèmes avec les droits des fichiers.

Commandes disponibles

```
cd /path/to/elgg/

# Get help
vendor/bin/elgg-cli --help

# List all commands
vendor/bin/elgg-cli list
```

(suite sur la page suivante)

```
# Install Elgg
vendor/bin/elgg-cli install [-c|--config CONFIG]

# Seed the database with fake entities
vendor/bin/elgg-cli database:seed [-l|--limit LIMIT]

# Remove seeded faked entities
vendor/bin/elgg-cli database:unseed

# Optimize database tables
# Request garbagecollector plugin
vendor/bin/elgg-cli database:optimize

# Run cron jobs
vendor/bin/elgg-cli cron [-i|--interval INTERVAL] [-q|--quiet]

# Clear caches
vendor/bin/elgg-cli cache:clear

# Invalidate caches
vendor/bin/elgg-cli cache:invalidate

# Purge caches
vendor/bin/elgg-cli cache:purge

# System upgrade
# -v/-vv/-vvv control verbosity of the command (helpful for debugging upgrade scripts)
vendor/bin/elgg-cli upgrade [-v]

# Upgrade and execute all async upgrades
vendor/bin/elgg-cli upgrade async [-v]

# List all, active or inactive plugins
# STATUS = all | active | inactive
vendor/bin/elgg-cli plugins:list [-s|--status STATUS]

# Activate plugins
# List plugin ids separating them with spaces: vendor/bin/elgg-cli plugins:activate_
↳ activity blog
# use -f flag to resolve conflicts and dependencies
vendor/bin/elgg-cli plugins:activate [<plugins>] [-f|--force]

# Deactivate plugins
# List plugin ids separating them with spaces: vendor/bin/elgg-cli plugins:deactivate_
↳ activity blog
# use -f flag to also disable dependents
vendor/bin/elgg-cli plugins:deactivate [<plugins>] [-f|--force]
```

Ajouter des commandes personnalisées

Les plugins peuvent ajouter leurs commandes à l'application CLI, en ajoutant le nom de la classe de commande via une configuration dans `elgg-plugin.php` ou via le hook `'commands', 'cli'`. La classe de commande doit étendre `\Elgg\CLI\Command`.

```
class MyCommand extends \Elgg\cli\Command {  
  
}  
  
elgg_register_plugin_hook_handler('commands', 'cli', function($hook, $type, $return) {  
  
    $return[] = MyCommand::class;  
  
    return $return;  
});
```

Les commandes personnalisées sont basées sur [Symfony Console Commands](#). Veuillez consulter leur documentation pour plus de détails.

3.2 Guides pour les administrateurs

Bonnes pratiques pour gérer efficacement un site construit avec Elgg.

3.2.1 Prise en main

Vous avez installé Elgg et réglé toutes les potentiels problèmes initiaux. Et maintenant ? Voici quelques suggestions pour vous aider à vous familiariser avec Elgg.

Concentrez-vous d'abord sur les fonctionnalités principales

Quand vous démarrez avec Elgg, le mieux est de commencer par explorer les fonctionnalités de base du noyau et des plugins livrés conjointement avant d'installer des plugins tierce-partie. Il est tentant d'installer chaque plugin intéressant du site communautaire, mais explorer les fonctionnalités du noyau vous permet de vous familiariser avec le comportement habituel d'Elgg, et évite d'introduire des bugs ou de la confusion dans votre nouveau réseau Elgg.

Elgg s'installe avec un jeu de plugins de réseau activés : des blogs, des marque-pages partagés, des fichiers, des groupes, des « likes », des messages, des pages de type wiki, des profils utilisateurs, et du microblogging. Pour changer les plugins activés, connectez-vous avec un compte admin, puis utilisez le menu supérieur pour accéder à Administration, puis à Plugins dans la barre latérale droite.

Note : L'utilisateur que vous créez pendant l'installation est un utilisateur admin.

Créez les utilisateurs de test

Les utilisateurs peuvent être créés de deux manières dans un Elgg standard :

1. Terminez le processus d'inscription en utilisant une autre adresse email et un autre nom d'utilisateur. (Déconnectez-vous d'abord ou utilisez un autre navigateur !)
2. Ajoutez un utilisateur dans la section Admin en naviguant vers Administration -> Utilisateurs -> Ajouter un nouvel utilisateur.

Note : Les utilisateurs qui s'inscrivent eux-même doivent valider leur compte avant de pouvoir se connecter. Les utilisateurs qu'un admin crée sont déjà validés.

Explorez les fonctionnalités de l'utilisateur

Utilisez vos utilisateurs de test pour créer des articles de blog, ajouter des widgets à votre profil ou à votre tableau de bord, publier sur le Fil (« The Wire », microblogging) et créer des pages (création de pages de type wiki). Examinez les Paramètres (« Settings ») dans la barre de menu supérieure. C'est à cet endroit qu'un utilisateur définit ses préférences et configure ses outils (qui peuvent être vides parce qu'aucun des plugins par défaut n'ajoute de contrôle à cet endroit).

Explorez les fonctionnalités de l'administration

Tous les outils d'administration sont accessibles en cliquant sur Administration dans le menu supérieur. L'administrateur a un tableau de bord avec un widget qui explique les diverses sections. Changez les options dans le menu Configurer pour changer l'apparence et le comportement de Elgg.

Étendre les fonctionnalités d'Elgg

Après avoir exploré ce qu'Elgg peut faire « out of the box », installez quelques thèmes et plugins. Vous pouvez trouver de nombreux plugins et thèmes sur le site de la communauté Elgg qui ont été développés par d'autres personnes. Ces plugins font tout, de la modification de chaînes de traductions à un re-design complet de l'interface d'Elgg, en passant par l'ajout d'un chat, etc. Puisque ces plugins ne sont pas officiels, assurez-vous de vérifier les commentaires pour vous assurer que vous installez des plugins bien écrits par des développeurs de haute qualité.

3.2.2 Installation avec Composer

Le moyen le plus simple pour garder votre site Elgg à jour est d'utiliser [Composer](#). Composer va s'occuper d'installer toutes les dépendances requises pour tous les plugins et pour Elgg, tout en conservant également ces dépendances à jour sans avoir de conflit.

Contenu

- *Installer Composer*
- *Installer Elgg en tant que projet Composer*
- *Mettre en place le contrôle de version*
- *Installer les plugins*
- *Commitez*
- *Déployer en production*

Installer Composer

<https://getcomposer.org/download/>

Installer Elgg en tant que projet Composer

```
composer self-update
composer create-project elgg/starter-project:dev-master ./path/to/my/project
cd ./path/to/my/project
composer install
```

Ceci va créer un fichier `composer.json` sur la base du projet de démarrage Elgg `Elgg starter project` qui a les bases pour installer Elgg.

Ouvrez votre navigateur

Allez sur votre navigateur et installez Elgg via l'interface d'installation

Mettre en place le contrôle de version

Cette étape est optionnelle mais vivement recommandée. Elle vous permettra de gérer simplement l'installation du même plugin entre différents environnements (développement/tests/production).

```
cd ./path/to/my/project
git init
git add .
git commit -a -m 'Initial commit'
git remote add origin <git repository url>
git push -u origin master
```

Installer les plugins

Installez les plugins sous forme de dépendances Composer. Ceci suppose qu'un plugin a été enregistré sur [Packagist](#)

```
composer require hypejunction/hypefeed
composer require hypejunction/hypeinteractions
# whatever else you need
```

Commitez

Assurez-vous que `composer.lock` n'est pas ignoré dans `.gitignore`

```
git add .
git commit -a -m 'Add new plugins'
git push origin master
```

Déployer en production

Déploiement initial

```
cd ./path/to/www

# you can also use git clone
git init
git remote add origin <git repository url>
git pull origin master

composer install
```

Déploiements subséquents

```
cd ./path/to/www
git pull origin master

# never run composer update in production
composer install
```

3.2.3 Mettre à niveau Elgg

Ce document va vous guider à travers les étapes nécessaires pour mettre à niveau votre installation Elgg vers la dernière version.

Si vous avez écrit des plugins personnalisés, vous devriez également lire les guides de développement pour les *informations sur la mise à jour du code des plugins* pour la dernière version d’Elgg.

Contenu

- *Conseil*
- *De 2.3 à 3.0*
 - *1. Mettez à niveau composer.json*
 - *2. Mettez à jour .htaccess*
 - *3a. Mise à niveau via Composer (recommandé)*
 - *3b. Mise à niveau manuelle (approche traditionnelle)*
- *Application d’un correctif à l’aide de Composer*
- *Versions précédentes*

Conseil

- **Sauvegardez votre base de données, votre répertoire de données et le code source**
- Faites attentions aux commentaires spécifiques pour certaines versions ci-dessous
- Les versions inférieures à 2.0 sont invitées à n'effectuer qu'**une seule mise à niveau mineure à la fois**
- Vous pouvez mettre à niveau depuis n'importe quelle version mineure vers n'importe quelle version mineure supérieure au sein de la même version majeure (2.0 -> 2.1 ou 2.0 -> 2.3)
- Vous pouvez mettre à niveau uniquement la dernière version mineure de la version majeure précédente vers n'importe quelle version mineure de la version majeure suivante (2.3 -> 3.0 ou 2.3 -> 3.2, mais pas 2.2 -> 3.x).
- A partir de Elgg 2.3.* vous pouvez mettre à niveau vers n'importe quelle version ultérieure de Elgg sans devoir passer par chacune des versions mineures (par ex. vous pouvez mettre à niveau directement de la 2.3.8 vers la 3.2.5, sans devoir mettre à niveau vers les versions 3.0 et 3.1)
- Essayez la nouvelle version sur un site de test avant d'effectuer une mise à niveau
- Signalez tout problème dans les plugins aux auteurs de plugins
- Si vous êtes auteur de plugins, vous pouvez [signaler tout problème de rétro-compatibilité sur GitHub](#)

De 2.3 à 3.0

1. Mettez à niveau composer.json

Si vous avez utilisé le projet de démarrage Elgg pour installer Elgg 2.3, vous pouvez avoir besoin de mettre à niveau votre `composer.json` :

- modifiez les pré-requis de la plateforme vers PHP ≥ 7.0
- facultativement, définissez les paramètres d'optimisation de l'autoloader
- facultativement, désactivez le plugin `fxp-asset` au bénéfice de `asset-packagist`

Votre `composer.json` devrait ressembler à quelque chose comme ceci (en fonction des changements que vous avez vous-même introduits) :

```
{
    "type": "project",
    "name": "elgg/starter-project",
    "require": {
        "elgg/elgg": "3.*"
    },
    "config": {
        "process-timeout": 0,
        "platform": {
            "php": "7.0"
        },
        "fxp-asset": {
            "enabled": false
        },
        "optimize-autoloader": true,
        "apcu-autoloader": true
    },
    "repositories": [
        {
            "type": "composer",
            "url": "https://asset-packagist.org"
        }
    ]
}
```

2. Mettez à jour .htaccess

Trouvez la ligne :

```
RewriteRule ^(.*)$ index.php?__elgg_uri=$1 [QSA,L]
```

Et remplacez-la par :

```
RewriteRule ^(.*)$ index.php [QSA,L]
```

3a. Mise à niveau via Composer (recommandé)

Si vous aviez installé votre projet 2.3 avec composer, vous pouvez suivre la séquence suivante :

Sauvegardez votre base de données, votre répertoire de données, et le code

```
composer self-update

cd ./path/to/project/root
composer require elgg/elgg:~3.0.0
composer update
vendor/bin/elgg-cli upgrade async -v
```

Note : Dans certains cas, la mise à niveau via la ligne de commande échoue car certaines modifications de schéma de base de données doivent d'abord être appliquées. Dans ce cas, vous devez exécuter les migrations *Phinx migrations* manuellement

3b. Mise à niveau manuelle (approche traditionnelle)

Les mises à niveau manuelles sont une opération pénible pour les administrateurs de sites. Nous vous déconseillons de maintenir un site Elgg en utilisant des archives ZIP. Economisez-vous du temps en apprenant comment utiliser `composer` et les systèmes de contrôle de version, tels que `git`. Cette tâche sera également plus complexe si vous utilisez des plugins tierce-partie et/ou si vous avez modifié les fichiers du noyau !

1. **Sauvegardez votre base de données, votre répertoire de données, et le code**
2. Identifiez-vous sur votre site en tant qu'administrateur
3. Téléchargez la nouvelle version de Elgg depuis <http://elgg.org>
4. **Mettez à jour les fichiers**
 - Si vous mettez à niveau vers une version majeure, vous devrez remplacer tous les fichiers du noyau et supprimer tous les fichiers qui ont été retirés du noyau d'Elgg, car ils peuvent interférer avec le bon fonctionnement de votre site.
 - Si vous mettez à jour vers une version mineure ou un patch, vous devez remplacer tous les fichiers du code.
5. **Fusionnez tous les nouveaux changements des règles de réécriture (rewrite rules)**
 - Pour Apache depuis `install/config/htaccess.dist` vers `.htaccess`
 - Pour Nginx depuis `install/config/nginx.dist` vers la configuration de votre serveur (habituellement dans `/etc/nginx/sites-enabled`)
6. Visitez <http://your-elgg-site.com/upgrade.php>
7. Exécutez les mises à niveau asynchrones via <http://your-elgg-site.com/admin/upgrades>

Note : Toute modification devrait avoir été faite au sein de plugins, de sorte qu'elles ne soient pas perdues lors de l'écrasement des fichiers. Si ce n'est pas le cas, prenez soin de maintenir vos modifications.

Note : Si vous n'arrivez pas à accéder au script `upgrade.php` et recevez une erreur, ajoutez `$CONFIG->security_protect_upgrade = false;` à votre fichier `settings.php` et supprimez-le après avoir terminé toutes les étapes de l'installation.

Note : Si vous rencontrez des soucis avec des plugins lors du processus de mise à niveau, ajoutez un fichier nommé `disabled` dans votre dossier `/mod/`. Ceci va désactiver tous les plugins, de sorte que vous puissiez terminer la mise à niveau du noyau. Vous pouvez ensuite gérer les mises à niveau de chacun des plugins l'un après l'autre.

Si vous avez installé Elgg en utilisant une archive d'installation mais souhaitez maintenant basculer vers composer :

- Mettez à niveau votre installation en utilisant la Méthode manuelle
- Déplacez votre code vers un emplacement provisoire
- Créez un nouveau projet composer en utilisant le projet Elgg de démarrage en suivant les [instruction d'installation](#) dans le répertoire racine de votre installation actuelle
- Copiez les plugins tierce-partie depuis votre ancienne installation vers le répertoire `/mod`
- Exécutez l'installateur de Elgg en utilisant votre navigateur ou l'outil `elgg-cli`
- Quand vous arrivez à l'étape de la base de données, indiquez les mêmes accès que vous avez utilisés pour l'installation manuelle, Elgg va comprendre qu'il y a une installation existante et ne remplacera aucune valeur de la base de données
- Facultativement, commitez votre nouveau projet vers un suivi de version

Application d'un correctif à l'aide de Composer

La définition d'un correctif se trouve dans la documentation [Release policy](#).

Les pré-requis de votre `composer.json` pour Elgg doivent être `~3.y.0` (où `y` est la version mineure `0`, `1`, etc. que vous souhaitez installer). Cela vous permettra d'installer facilement des correctifs sans risque d'installer la prochaine version mineure.

```
{
    "require": {
        "elgg/elgg": "~3.0.0"
    }
}
```

Juste pour vous en assurer vous pouvez d'abord vérifier ce qui sera installé / mis à niveau en exécutant la commande suivante

```
# to get a full list of all packages which can be upgraded
composer update --dry-run

# or if you only wish to check for Elgg
composer update elgg/elgg --dry-run
```

Pour mettre à niveau Elgg, exécutez simplement

```
# to upgrade all packages
composer update

# or to only upgrade Elgg
composer update elgg/elgg
```

Versions précédentes

Vérifiez la documentation Elgg qui correspond à la version Elgg vers laquelle vous souhaitez mettre à niveau, en changeant la version de documentation dans le coin inférieur gauche de [Upgrading docs](#)

3.2.4 Plugins

Les plugins peuvent modifier le comportement d’Elgg et ajouter de nouvelles fonctionnalités.

Contenu

- *Où trouver des plugins*
- *La Communauté Elgg*
 - *Trouver des plugins*
 - *Évaluer les Plugins*
- *Types de plugins*
 - *Thèmes*
 - *Packs de langues*
- *Installation*
- *Ordre des plugins*

Où trouver des plugins

Des plugins peuvent être obtenus depuis :

- *La Communauté Elgg*
- [Github](#)
- Sites tierce-partie (généralement pour un certain prix)

Si aucun plugin existant ne correspond à vos besoins, vous pouvez solliciter un développeur [hire a developer](#) ou créer le vôtre *create your own*.

La Communauté Elgg

Trouver des plugins

Ordre de tri basé sur la popularité

Sur la page des plugins de la communauté, vous pouvez trier par date de mise en ligne (Filtre : Newest) ou par nombre de téléchargements (Filtre : Most downloads). Trier par nombre de téléchargement est une bonne idée si vous débutez avec Elgg et voulez voir quels plugins sont fréquemment utilisés par d’autres administrateurs. Ceux-ci sont souvent (mais pas toujours) des plugins de meilleure qualité qui offrent des possibilités significatives.

Utilisez la recherche de plugin par tag

Une boîte de recherche se trouve à côté des options de filtrage de la page des plugins. Elle vous permet de rechercher des plugins par tags. Les auteurs des plugins définissent les tags.

Identifiez les auteurs de plugins particuliers

La qualité des plugins varie de manière substantielle. Si vous trouvez un plugin qui fonctionne bien sur votre site, vous pouvez regarder les autres réalisations que l'auteur du plugin a développées en cliquant sur son nom lorsque vous êtes sur la page d'un plugin.

Évaluer les Plugins

Regardez les commentaires et les notations

Avant de télécharger et d'utiliser un plugin, c'est toujours une bonne idée de lire les commentaires que les autres ont laissés. Si vous voyez des personnes qui se plaignent que le plugin ne fonctionne pas ou qu'il rend leur site instable, vous devriez probablement éviter ce plugin. Néanmoins, certaines utilisateurs ignorent les instructions d'installation ou installent un plugin de manière incorrecte puis laissent un feedback négatif. De plus, certains auteurs de plugins choisissent de ne pas autoriser les commentaires.

Installer sur une site de test

Si vous essayez un plugin pour la première fois, c'est une mauvaise idée de l'installer sur un site de production. Vous devriez maintenir un site de test séparé pour évaluer les plugins. C'est une bonne idée de mettre en place progressivement les plugins sur votre site de production même après qu'ils aient passé l'évaluation sur un site de test. Ceci vous permet d'isoler les problèmes potentiels introduits par un nouveau plugin.

Types de plugins

Thèmes

Les thèmes sont des plugins qui modifient l'apparence et le comportement (look-and-feel) de votre site. Ils comprennent généralement des feuilles de styles, des scripts côté client et des vues qui modifient la présentation et le comportement par défaut d'Elgg.

Packs de langues

Les packs de langue (ou de traduction) sont des plugins qui fournissent le support d'autres langues.

Les packs de traduction peuvent étendre et inclure des traductions pour les chaînes de traduction du noyau, des plugins du noyau et/ou de plugins tiers-parties.

Quelques packs de langues sont déjà compris dans le noyau, et se trouvent dans le dossier `languages` du répertoire racine d'Elgg. Les plugins individuels ont leurs traductions dans le répertoire `languages` de la racine du plugin.

Cette structure facilite la création de nouveaux packs de langues qui remplacent les chaînes de traduction existantes ou ajoutent le support de nouvelles langues.

Installation

Tous les plugins sont dans le dossier mod de votre installation Elgg.

Pour installer un nouveau plugin :

- décompresser (unzip) le contenu du package de distribution du plugin
- copiez/transférez via FTP le dossier décompressé dans le dossier mod de votre installation Elgg, en vous assurant que `manifest.xml` est directement placé dans le dossier du plugin (par ex. si vous alliez installer un plugin appelé `my_elgg_plugin`, le manifest du plugin aurait besoin de se trouver à `mod/my_elgg_plugin/manifest.xml`)
- activez le plugin depuis votre panneau d'administration

Pour activer un plugin :

- Connectez-vous à votre site Elgg avec votre compte administrateur
- Allez dans Administration -> Configurer -> Plugins
- Trouvez votre plugin dans la liste des plugins installés et cliquez sur le bouton "activer".

Ordre des plugins

Les plugins sont chargés selon l'ordre dans lequel ils sont listés sur la page Plugins. L'ordre initial après une installation est plus ou moins aléatoire. AU fur et à mesure que de nouveaux plugins sont ajoutés par un administrateur, ils sont placés en fin de liste.

Quelques règles générales pour l'ordre des plugins :

- Un plugin de thème devrait être placé en dernier ou au moins proche du bas de la liste
- Un plugin qui modifie le comportement d'un autre plugin devrait être placé plus bas dans la liste des plugins

3.2.5 Performance

Faites que votre site fonctionne sans heurt et de manière aussi réactive que possible.

Contenu

- *Est-ce Elgg peut passer à l'échelle de X millions d'utilisateurs ?*
- *Mesurez d'abord*
- *Régler MySQL*
- *Activer la mise en cache*
 - *Cache simple (Simplecache)*
 - *Cache système (System cache)*
 - *Cache de démarrage*
 - *Cache des requêtes de base de données*
 - *Etags et headers Expires*
 - *Memcached*
 - *Squid*
 - *Mise en cache du Bytecode*
 - *Servir directement les fichiers*
 - *Optimisation de Composer Autoloader*
- *Hébergement*
 - *Mémoire, CPU et bande passante*
 - *Configuration*
- *Vérifiez les plugins au mauvais comportement*
- *Utilisez du HTML rendu côté client*

Est-ce Elgg peut passer à l'échelle de X millions d'utilisateurs ?

Les gens demandent souvent si Elgg convient pour de grands sites.

Tout d'abord, nous pourrions vous arrêter pour demander : « où et comment prévoyez-vous d'obtenir tous ces utilisateurs ? » Sérieusement, c'est un problème très intéressant. Faire évoluer Elgg est avant tout une question d'ingénierie technique. C'est intéressant, et plus ou moins un problème résolu. L'informatique ne fonctionne pas différemment pour Elgg que pour Google, par exemple. Obtenir des millions d'utilisateurs ? C'est comme le Saint Graal de toute l'industrie de la technologie.

Deuxièmement, comme avec la plupart des choses dans la vie, la réponse est « ça dépend » :

- Que font vos utilisateurs ?
- Sur quelle machine et avec quelle configuration fonctionne Elgg ?
- Est-ce que vos plugins se comportent bien ?

« **Améliorer l'efficacité du moteur Elgg** » est un projet en cours, bien qu'il y ait des limites à la quantité que n'importe quel script peut faire.

Si vous êtes sérieux au sujet de l'évolutivité, vous aurez probablement envie de considérer un certain nombre de choses vous-même.

Mesurez d'abord

Il ne sert à rien de jeter des ressources sur un problème si vous ne connaissez pas :

- quel est le problème
- de quelles ressources ce problème a besoin
- où ces ressources sont nécessaires

Investissez dans une sorte de profilage afin de vous faire connaître où votre goulot d'étranglement se situe, surtout si vous envisagez d'injecter de l'argent important sur un problème.

Régler MySQL

Elgg fait un usage intensif de la base de données en back-end, faisant, à chaque chargement de page, de nombreux appels et allers et retours entre le serveur web et la base de données. Ceci est parfaitement normal et un serveur de base de données bien configuré sera capable de faire face à des milliers de requêtes par seconde.

Voici quelques astuces de configuration qui pourraient aider :

- Assurez-vous que MySQL utilise un fichier de configuration my.cnf adapté à la taille de votre site web.
- Augmentez la quantité de mémoire disponible pour PHP et de même pour MySQL (vous aurez dans tous les cas à augmenter la quantité de mémoire disponible pour le processus de php)

Activer la mise en cache

En règle générale, si un programme est lent, c'est parce qu'elle effectue à plusieurs reprises un calcul ou une opération coûteuse. La mise en cache permet au système d'éviter de faire ce travail encore et encore à chaque demande grâce à la mise en cache des résultats précédents. En utilisant la mémoire pour stocker les résultats, il est alors facile de récupérer en mémoire le résultat de la demande et le passer aux demandes suivantes, économisant ainsi à chaque demande tout le travail de calcul. Ci-dessous, nous discutons de plusieurs solutions de mise en cache généralement disponibles et pertinentes pour Elgg.

Cache simple (Simplecache)

Par défaut, les vues sont mises en cache dans le répertoire de données de Elgg pour une période de temps donnée. Cela supprime la nécessité de régénérer chaque vue à chaque chargement de la page.

Ceci peut être désactivé en définissant `$CONFIG->simplecache_enabled = false;` Pour de meilleures performances, assurez-vous que cette valeur est bien définie sur `true`.

Cela conduit à des artefacts pendant le développement/la programmation si par exemple vous modifiez un thème dans votre plugin puisque la version mise en cache sera utilisée de préférence à la nouvelle fournie par votre plugin.

Simple cache peut être désactivée via le menu d'administration. Il est recommandé de le faire sur votre plate-forme de développement si vous écrivez des plugins Elgg.

Ce cache est automatiquement vidé à chaque fois qu'un plugin est activé, désactivé ou réorganisé, ou quand le script `upgrade.php` est exécuté.

Pour de meilleures performances, vous pouvez également créer un lien symbolique depuis `/cache/` dans votre répertoire racine web vers le répertoire `assetroot` spécifié dans votre configuration (par défaut il est situé dans `/path/to/dataroot/caches/views_simplecache/` :

```
cd /path/to/wwwroot/  
ln -s /path/to/dataroot/caches/views_simplecache/ cache
```

Si votre serveur web supporte les liens symboliques (« symlinks »), ceci va servir les fichiers directement depuis le disque sans démarrer PHP à chaque fois.

Pour des raisons de sécurité, quelques serveurs web (par ex. Apache dans la version 2.4) pourraient par défaut ne suivre les liens symboliques que si le propriétaire de la source et de la cible du lien correspondent. Si le lien symbolique du cache ne fonctionne pas sur votre serveur web, vous pouvez changer le propriétaire du lien symbolique du cache lui-même (et non du répertoire `/views_simplecache/`) avec

```
cd /path/to/wwwroot/  
chown -h wwwrun:www cache
```

Dans cet exemple on prend l'hypothèse que le répertoire `/views_simplecache/` du répertoire de données appartient au compte `wwwrun` qui fait partie du groupe `www`. Si ce n'est pas le cas sur votre serveur, vous devez modifier la commande `chown` en conséquence.

Cache système (System cache)

L'emplacement des vues est mis en cache afin de retrouver rapidement les vues (en effet, le profilage a indiqué que le temps de chargement d'une page prend une quantité non-linéaire du temps plus il y a de plugins activés, ceci en raison de la recherche des vues). Elgg met également en cache des informations telles que la cartographie de la langue et de la carte des classes.

Ceci peut être désactivé en définissant `$CONFIG->system_cache_enabled = false;` Pour de meilleures performances, assurez-vous que cette valeur est bien définie sur `true`.

Les emplacements des vues sont actuellement stockés dans des fichiers placés dans votre dossier de données (toutes fois des versions ultérieures de Elgg peuvent utiliser `memcache`). Comme avec le cache simple, le cache système est automatiquement vidé chaque fois qu'un plugin est activé, désactivé ou réorganisé, ou quand le script `upgrade.php` est exécuté.

Le cache du système peut être désactivé via le menu d'administration, et il est recommandé de le faire sur votre plate-forme de développement si vous écrivez des plugins Elgg.

Cache de démarrage

Elgg a la capacité de mettre en cache de nombreuses ressources créées et récupérées pendant le processus de démarrage. Pour configurer la durée de validité de ce cache vous devez définir la valeur du TTL dans votre fichier `settings.php` :

```
$CONFIG->boot_cache_ttl = 3600;
```

Regardez les documentation de [Stash](#) pour plus d'informations sur le TTL.

Cache des requêtes de base de données

Pour la durée de vie de l'exécution d'une page donnée, un cache de toutes les requêtes `SELECT` est conservé. Cela signifie que pour le chargement d'une page donnée, une requête sélectionnée donnée ne sortira dans la base de données qu'une seule fois, même si elle est exécutée plusieurs fois. Toute écriture dans la base de données videra ce cache. Ce cache sera automatiquement effacé à la fin du chargement d'une page.

Vous pouvez rencontrer des problèmes de mémoire si vous utilisez le framework Elgg comme bibliothèque dans un script PHP CLI. Ceci peut être désactivé en définissant `$CONFIG->db_disable_query_cache = true;`

Etags et headers Expires

Ces technologies indiquent aux navigateurs de vos utilisateurs de mettre en cache des ressources statiques (CSS, JS, images) localement. Avoir ceci activé réduit fortement la charge du serveur et améliore la performance perçue par les utilisateurs.

Utilisez le **plugin Firefox yslow** ou les Outils de Développement de Chrome DevTools Audits pour confirmer les technologies utilisées actuellement sur votre serveur.

Si les ressources statiques ne sont pas mises en cache :

- Vérifiez que vous avez ces extensions et activées sur votre hébergement
- Mettez à jour votre fichier `.htaccess`, si vous faites une mise à niveau depuis une version précédente d'Elgg
- Activez *Simplecache*, qui transforme les vues sélectionnées en ressources (assets) pouvant être mis en cache par le navigateur

Memcached

Libmemcached a été créé par Brian Aker et a été conçu dès le départ pour offrir la meilleure performance possible aux utilisateurs de Memcached.

Voir aussi :

<http://libmemcached.org/About.html> and <https://secure.php.net/manual/en/book.memcached.php>

Pré-requis pour l'installation :

- php-memcached
- libmemcached
- memcached

Configuration :

Dé-commentez et renseignez les sections suivantes dans le fichier `settings.php`

```
$CONFIG->memcache = true;

$CONFIG->memcache_servers = array (
    array('server1', 11211),
```

(suite sur la page suivante)

(suite de la page précédente)

```
array('server2', 11211)
);
```

De manière optionnelle, si vous avez plusieurs installations Elgg mais n'utilisez qu'un seul serveur Memcache, vous pouvez vouloir ajouter un préfixe d'espace de nom (namespace prefix). Afin de faire cela, décommentez la ligne suivante

```
$CONFIG->memcache_namespace_prefix = '';
```

Squid

Nous avons obtenu de bons résultats en utilisant [Squid](#) pour mettre en cache les images.

Mise en cache du Bytecode

Il existe de nombreux caches de code disponibles sur le marché. Ceux-ci accélèrent votre site en mettant en cache le byte code compilé depuis votre script, ce qui signifie que votre serveur n'a pas à compiler le code PHP à chaque fois qu'il est exécuté.

Servir directement les fichiers

Si votre serveur peut être configuré pour supporter les headers X-Sendfile ou X-Accel, vous pouvez configurer leur utilisation dans `settings.php`. Ceci permet à votre serveur web de diffuser directement les fichiers au client au lieu d'utiliser la fonction PHP `readfile()`.

Optimisation de Composer Autoloader

Le chargeur automatique Composer est responsable du chargement des classes fournies par les dépendances d'Elgg. La façon dont le chargeur automatique fonctionne est qu'il recherche un nom de classe dans les dépendances installées. Bien qu'il s'agisse principalement d'un processus rapide, il peut être optimisé.

Vous pouvez optimiser l'autoloader de 2 façons différentes. La première est dans la ligne de commande, l'autre est dans le `composer.json` de votre projet.

Si vous souhaitez optimiser le chargeur automatique à l'aide de la ligne de commande, utilisez le drapeau `-o`. L'inconvénient est que vous devez ajouter le drapeau `-o` à chaque fois que vous exécutez Composer.

```
# During the installation
composer install -o

# Or during the upgrade process
composer upgrade -o
```

La deuxième option consiste à ajouter l'optimisation à votre fichier `composer.json`, de cette façon vous ne l'oubliez jamais.

```
{
    "config": {
        "optimize-autoloader": true,
        "apcu-autoloader": true
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
}
}
```

Voir aussi :

Consultez la page [`Autoloader Optimization`](#) pour plus d'informations sur l'optimisation du chargeur automatique Composer.

Note : À partir d'Elgg 3.0 tous les *téléchargements* d'Elgg à partir du site disposent du chargeur automatique optimisé.

Hébergement

N'espérez pas de faire tourner un site pour des millions d'utilisateurs sur un hébergement mutualisé à bas prix. Vous aurez besoin de votre propre serveur d'hébergement et d'avoir la main sur la configuration, ainsi que de beaucoup de bande passante et de mémoire disponibles.

Mémoire, CPU et bande passante

De par la nature de la mise en cache, toutes les solutions de mise en cache auront besoin de mémoire. C'est un investissement plutôt économique d'augmenter la mémoire et le CPU.

Sur un matériel puissant il est probable que la bande passante soit le goulet d'étranglement (bottleneck) avant le serveur lui-même. Assurez-vous que votre hébergement peut supporter le trafic que vous attendez.

Configuration

Enfin, jetez un coup d'œil à votre configuration car il y a quelques points d'attention qui peuvent surprendre les gens.

Par exemple, Apache peut d'emblée gérer une charge plutôt élevée. Cependant, la majorité des distributions Linux sont livrées avec mysql configuré pour de petits sites. Ceci peut donner lieu à des processus Apache bloqués qui attendent de pouvoir parler à un processus MySQL très surchargé.

Vérifiez les plugins au mauvais comportement

Des plugins peuvent être développés d'une manière très naïve et ceci peut ralentir l'ensemble du site.

Essayez de désactiver quelques plugins pour voir si cela améliore notablement les performances. Une fois que vous avez trouvé un responsable potentiel, rendez-vous sur la page de l'auteur du plugin et signalez vos résultats.

Utilisez du HTML rendu côté client

Nous avons constaté qu'à un certain niveau, une bonne partie du temps passé sur le serveur est simplement le temps de construction du HTML de la page avec le système de vues d'Elgg.

Il est très difficile de mettre en cache les sorties des modèles (templates) dans la mesure où elles prennent généralement des entrées arbitraires. Au lieu d'essayer mettre en cache la sortie de certaines pages ou vues, la suggestion est de basculer vers un système de modèles basé sur HTML de sorte que le navigateur de l'utilisateur puisse lui-même mettre en cache les modèles. Puis demandez à l'ordinateur de l'utilisateur de générer le résultat en appliquant des données JSON à ces modèles.

Ceci peut être très efficace, mais présente l'inconvénient de demander un coup de développement supplémentaire significatif. L'équipe d'Elgg envisage d'intégrer cette stratégie directement dans Elgg, dans la mesure où ceci est aussi efficace, tout particulièrement avec les pages avec du contenu répété ou caché.

3.2.6 Cron

Contenu

- *Qu'est-ce que ça fait ?*
- *Comment ça marche ?*

Qu'est-ce que ça fait ?

Cron est un programme disponible sur les systèmes d'exploitation basés sur Unix qui permet aux utilisateurs d'exécuter des commandes et des scripts à des intervalles particuliers ou à des heures spécifiques.

Le gestionnaire de cron d'Elgg permet aux administrateurs et aux développeurs de plugins de mettre en place des tâches qui ont besoin d'être exécutées à des intervalles définis.

Les exemples les plus courants de tâches cron dans Elgg incluent :

- l'envoi des notifications en file d'attente
- la rotation des journaux système dans la base de données
- le nettoyage des déchets dans la base de données (compactage de la base de données par la suppression des entrées qui ne sont plus requises)

Les plugins peuvent ajouter des tâches en enregistrant un gestionnaire de hook plugin pour l'un des intervalles de cron suivants :

- `minute` - Exécuté toutes les minutes
- `fiveminute` - Exécuté toutes les 5 minutes
- `fifteenmin` - Exécuté toutes les 15 minutes
- `halfhour` - Exécuté toutes les 30 minutes
- `hourly` - Exécuté toutes les heures
- `daily` - Exécuté tous les jours
- `weekly` - Exécuté toutes les semaines
- `monthly` - Exécuté tous les mois
- `yearly` - Exécuté tous les ans

```
elgg_register_plugin_hook_handler('cron', 'hourly', function() {  
  
    $events = my_plugin_get_upcoming_events();  
  
    foreach ($events as $event) {  
        $attendees = $event->getAttendees();  
  
        // notify  
    }  
});
```

Comment ça marche ?

crontab doit être configuré de telle manière qu'il active le gestionnaire de cron d'Elgg chaque minute, ou à un intervalle spécifique. Une fois que l'entrée du cron active le travail du cron, Elgg exécute chaque gestionnaire de hook attaché à cet intervalle.

Si vous avez un accès SSH à vos serveurs Linux, tapez `crontab -e` et ajoutez votre configuration crontab.

```
* * * * * path/to/phpbin path/to/elgg/elgg-cli cron -q
```

La commande suivante va s'exécuter chaque minute et activer toutes les tâches du cron attendues.

Facultativement, vous pouvez activer les gestionnaire pour un intervalle spécifique :

```
0 * * * * path/to/phpbin path/to/elgg/elgg-cli cron -i hourly -q
```

Vous pouvez trouver plus d'informations sur cron sur :

3.2.7 Sauvegarde et Restauration

Contenu

- *Introduction*
- *Pourquoi*
- *Quoi*
- *Hypothèses*
- *Créer un backup utilisable - automatiquement*
 - *Personnalisez le script de sauvegarde automatique (backup)*
 - *Configurez la tâche périodique (cron) de backup*
 - *Configurez la tâche périodique (cron) de nettoyage*
- *Restauration depuis un backup*
 - *Préparez vos fichiers de backup*
 - *Restaurez les fichiers*
 - *Restaurez la base de données MySQL*
 - *Modifiez le backup MySQL*
 - *Créez la nouvelle base de données*
 - *Restaurez la base de données de production*
 - *Combiner l'ensemble*
 - *Finaliser la nouvelle installation*
- *Félicitations !*
- *Connexe*

Introduction

Pourquoi

Les fournisseurs d'hébergements mutualisés ne fournissent généralement pas de moyen automatisé de sauvegarder votre installation Elgg. Cet article propose une méthode permettant d'accomplir cette tâche.

Dans l'IT il existe souvent de nombreuses manières d'accomplir la même chose. Gardez cela à l'esprit. Cet article va expliquer une méthode pour sauvegarder et restaurer votre installation Elgg sur un hébergement mutualisé qui utilise

l'application CPanel. Cependant, les idées présentées ici peuvent être ajustées pour d'autres applications également. Ce qui suit sont des situations typiques qui peuvent exiger une procédure telle que celle-ci :

- Récupération après une catastrophe
- Déplacer votre site Elgg vers un nouvel hôte
- Dupliquer une installation

Quoi

Sujets traités :

- Des backups complets des répertoires d'Elgg et des bases de données MySQL sont effectués tous les jours (automatisé)
- Les backups sont envoyés sur un autre site via FTP (automatisé)
- Les backups locaux sont supprimés après un transfert réussi vers le site distant (automatisé)
- Cinq jours de backups seront conservés (automatisé)
- Restauration des données sur le nouvel hôte (manuel)

Ce processus a été composé à partir d'articles d'assistance précédemment publiés dans le wiki de documentation d'Elgg.

Hypothèses

Les hypothèses suivantes ont été faites :

- Le répertoire du programme Elgg est `/home/userx/public_html`
- Le répertoire de données Elgg est `/home/userx/elggdata`
- Vous avez créé un répertoire local pour vos backups à `/home/userx/sitebackups`
- Vous disposez d'un server FTP distant auquel envoyer les fichiers de backup
- Le répertoire dans lequel vous allez enregistrer les backups distants est `/home/usery/sitebackups/`
- Vous allez restaurer le site chez un deuxième fournisseur d'hébergement mutualisé dans le répertoire `/home/usery/public_html`

Important : Assurez-vous de remplacer `userx`, `usery`, `http://mynewdomain.com` et tous les mots de passe par des valeurs qui correspondent à votre propre installation !

Créer un backup utilisable - automatiquement

Personnalisez le script de sauvegarde automatique (backup)

Le script que vous allez utiliser peut être trouvé [ici](#) .

Copiez simplement le script dans un fichier texte et renommez le fichier avec une extension `.pl`. Vous pouvez utiliser n'importe quel éditeur de texte pour modifier le fichier.

Modifiez les valeurs suivantes pour qu'elles correspondent à votre structure de répertoires :

```
# ENTER THE PATH TO THE DIRECTORY YOU WANT TO BACKUP, NO TRAILING SLASH
$directory_to_backup = '/home/userx/public_html';
$directory_to_backup2 = '/home/userx/elggdata';
# ENTER THE PATH TO THE DIRECTORY YOU WISH TO SAVE THE BACKUP FILE TO, NO TRAILING SLASH
$backup_dest_dir = '/home/userx/sitebackups';
```

Modifiez les valeurs suivantes pour qu'elles correspondent à vos paramètres de base de données :

```
# MYSQL BACKUP PARAMETERS
$dbhost = 'localhost';
$dbuser = 'userx_elgg';
$dbpwd = 'dbpassword';
# ENTER DATABASE NAME
$database_names_elgg = 'userx_elgg';
```

Modifiez les valeurs suivantes pour qu'elles correspondent aux paramètres de votre serveur FTP distant :

```
# FTP PARAMETERS
$ftp_host = "FTP HOSTNAME/IP";
$ftp_user = "ftpuser";
$ftp_pwd = "ftppassword";
$ftp_dir = "/";
```

Enregistrez le fichier avec l'extension `.pl` (dans le cadre de cet article nous appellerons ce fichier : `elgg-ftp-backup-script.pl`) et chargez-le dans le répertoire `/home/userx/sitebackups`

Soyez conscient(e) que vous pouvez désactiver le FTP et modifier un peu le script de sorte qu'il ne supprime pas le fichier de backup local dans le cas où vous ne voulez pas utiliser de stockage distant pour vos backups.

Configurez la tâche périodique (cron) de backup

Connectez-vous à votre application CPanel et cliquez sur le lien « Cron Jobs ». Dans le menu déroulant Paramètres communs (Common Settings), choisissez « Une fois par jour » (« Once a day ») et tapez ce qui suit dans le champ de commande `/usr/bin/perl /home/userx/sitebackups/elgg-ftp-backup-script.pl`

Cliquez sur le bouton « Add New Cron Job ». Des sauvegardes quotidiennes complètes sont désormais planifiées et seront transférées hors du site.

Configurez la tâche périodique (cron) de nettoyage

Si vous envoyez vos sauvegardes, via FTP, à un autre hébergeur mutualisé qui utilise l'application CPanel ou si vous avez totalement désactivé FTP, vous pouvez configurer votre conservation des données comme suit.

Connectez-vous à votre application CPanel pour votre site FTP, ou localement si vous n'utilisez pas FTP, et cliquez sur le lien « Cron Jobs ». Dans le menu déroulant Paramètres communs (Common Settings) choisissez Une fois par jour (« Once a day ») et saisissez ce qui suit dans le champ de commande `find /home/usery/sitebackups/full_* -mtime +4 -exec rm {} \;`

Le paramètre `-mtime X` va définir le nombre de jours de rétention des sauvegardes. Tous les fichiers plus anciens que le nombre de jour `x` sera supprimé. Cliquez sur le bouton « Add New Cron Job ». Vous avez maintenant configuré la durée de rétention de vos sauvegardes.

Restauration depuis un backup

Préparez vos fichiers de backup

L'hypothèse est que vous restaurez votre site vers un autre prestataire d'hébergement mutualisé avec CPanel.

Lorsque le script a sauvegardé les fichiers, la structure originelle des répertoires est conservée dans le fichier zip. Nous devons faire un petit nettoyage. Effectuer ce qui suit :

- Téléchargez le fichier de backup à partir duquel vous souhaitez effectuer la restauration
- Décompressez le contenu du fichier de backup
- **Naviguez pour trouver la sauvegarde de votre site et la sauvegarde SQL. Extrayez les deux archives. Vous aurez alors :**
 - un dump MySQL avec une extension « .sql »
 - **une autre structure de répertoire avec pour contenu :**
 - « /home/userx/public_html »
 - /home/userx/elggdata
- **Re-compressez le contenu du répertoire /home/userx/public_html sous forme de fichier zip de sorte que les fichiers soient à la racine du fichier zip**
 - La raison de cette démarche est simple. Il est beaucoup plus efficace de télécharger un seul fichier zip que d'envoyer par FTP sur votre nouvel hôte tout le contenu du répertoire / home/userx/public_html.
- Re-compressez le contenu du répertoire /home/userx/elggdata directory sous forme de fichier zip de sorte que les fichiers soient à la racine du fichier zip

Vous devriez maintenant avoir les fichiers suivants :

- le fichier « .sql »
- le fichier zip avec le contenu de /home/userx/public_html à la racine
- le fichier zip avec le contenu de /home/userx/elggdata ` à la racine

Restaurez les fichiers

Ceci est écrit dans l'hypothèse que vous restaurez vers un hébergement différent mais conservez la même structure de répertoires. Faites comme suit :

- Connectez-vous à l'application CPanel de l'hébergement sur lequel vous souhaitez restaurer le site et ouvrez le gestionnaire de fichiers (File Manager).
- **Naviguez vers /home/usery/public_html**
 - Chargez le fichier zip qui contient les fichiers de /home/userx/public_html
 - **Décompressez le fichier zip** Vous devriez maintenant voir tous les fichiers dans /home/usery/public_html
 - Supprimez le fichier zip
- **Naviguez vers /home/usery/elggdata**
 - Chargez le fichier zip qui contient les fichiers de /home/userx/elggdata
 - **Décompressez le fichier zip** Vous devriez maintenant voir tous les fichiers dans /home/usery/elggdata
 - Supprimez le fichier zip

La restauration des fichiers du programme et des données est terminée

Restaurez la base de données MySQL

Note : A nouveau, l'hypothèse ici est que vous restaurez votre installation Elgg vers un autre fournisseur d'hébergement mutualisé. Chaque fournisseur d'hébergement mutualisé ajoute le nom du compte du détenteur comme préfixe aux bases de données associées avec ce compte. Par exemple, le nom d'utilisateur de notre premier hébergement est `userx` de sorte que l'hébergeur va ajouter le préfixe `userx_` pour nous donner un nom de base de données de `userx_elgg`. Quand nous restaurons vers notre second fournisseur d'hébergement mutualisé nous le faisons avec un nom d'utilisateur de `usery` de sorte que le nom de notre base de données sera `usery_elgg`. Les fournisseurs d'hébergement ne vous permettent pas de modifier ce comportement. De sorte que le processus ici n'est pas aussi simple que simplement restaurer la base de données depuis le backup vers le compte usery. Néanmoins, une fois ceci dit, ce n'est pas si difficile non plus.

Modifiez le backup MySQL

Ouvrez le fichier ``.sql`` que vous avez extrait de votre backup dans votre éditeur de texte favori. Commentez les lignes suivantes avec un dièse :

```
#CREATE DATABASE /*!32312 IF NOT EXISTS*/ `userx_elgg` /*!40100 DEFAULT CHARACTER SET_
↪latin1 */;
#USE `userx_elgg`;
```

Enregistrez le fichier.

Créez la nouvelle base de données

Effectuez les actions suivantes :

- Connectez-vous à l'application CPANEL du nouvel hôte et cliquez sur l'icône « Bases de données MySQL (« MySQL Databases »)
- Renseignez le nom de la base de données et cliquez sur le bouton créer (« create »). Pour notre exemple, nous allons conserver `elgg` ce qui nous donne pour nom de base de données `usery_elgg`
- Vous pouvez associer un utilisateur existant à la nouvelle base de données, ou pour créer un nouvel utilisateur vous pouvez :
 - Allez dans la section « Ajouter Nouvel utilisateur » de la page « Bases de données MySQL »
 - Saisissez le nom d'utilisateur et le mot de passe. Pour notre exemple nous allons rester simples et conserver à nouveau `elgg`. Ce qui va nous donner un nom d'utilisateur `usery_elgg`
- Associez le nouvel utilisateur avec la nouvelle base de données
 - Allez dans la section Ajouter un nouvel utilisateur à la base de données (« Add User To Database ») de la page Bases de données MySQL (« MySQL Databases »). Ajoutez l'utilisateur `usery_elgg` à la base de données `usery_elgg`
 - Sélectionnez « Tous les Privilèges » (« All Privileges ») et cliquez sur le bouton « Effectuer les changements » (« Make Changes »)

Restaurez la base de données de production

Maintenant il est temps de restaurer le fichier de backup MySQL en l'important dans notre nouvelle base de données nommée « usery_elgg ».

- **Connectez-vous à l'application CPanel du nouvel hôte et cliquez sur l'« icône phpMyAdmin**
 - Choisissez la base de données usery_elgg dans la colonne de gauche
 - Cliquez sur l'onglet « import » en haut de la page
 - Naviguez jusqu'au backup .sql dans votre arborescence locale et sélectionnez-le
 - Cliquez sur le bouton « Go » en bas à droite de la page

Vous devriez maintenant voir un message indiquant que l'opération avec succès

Combiner l'ensemble

L'installation elgg restaurée ne connaît **rien** du nom de la nouvelle base de données, du nom d'utilisateur, de la structure des répertoires, etc. C'est ce dont nous allons traiter ici.

Modifiez /public_html/elgg-config/settings.php sur le nouvel hébergement pour reproduire les informations de la base de données que vous venez de créer.

```
// Database username
$CONFIG->dbuser = 'usery_elgg';

// Database password
$CONFIG->dbpass = 'dbpassword';

// Database name
$CONFIG->dbname = 'usery_elgg';

// Database server
// (For most configurations, you can leave this as 'localhost')
$CONFIG->dbhost = 'localhost';

// (For most configurations, you can leave this as 3306)
$CONFIG->dbport = 3306;

$CONFIG->wwwroot = 'http://your.website.com/'
```

Chargez à nouveau le fichier settings.php vers le nouvel hôte - en remplaçant le fichier existant.

Ouvrez l'outil phpMyAdmin sur le nouvel hôte depuis CPanel. Sélectionnez la base de données usery_elgg sur la gauche, et cliquez sur l'onglet SQL en haut de la page. Exécutez les requêtes SQL suivantes sur la base de données usery_elgg :

Modifiez le chemin d'installation

```
UPDATE `elgg_config` SET `value` = REPLACE(`value`, "/home/userx/public_html/grid/", "/home/userx/public_html/grid/") WHERE `name` = "path";
```

Modifiez le répertoire de données

```
UPDATE `elgg_config` SET `value` = REPLACE(`value`, "/home/userx/elggdata/", "/home/userx/elggdata/") WHERE `name` = "dataroot";
```

Modifiez le répertoire de données de stockage des fichiers

```
UPDATE elgg_metadata SET value = '/home/userx/elggdata/' WHERE name = 'filestore::dir_
↳root';
```

Finaliser la nouvelle installation

Exécutez le script de mise à niveau en visitant l'URL suivante : `http://mynewdomain.com/upgrade.php`. Effectuez cette étape deux fois - depuis le début.

Mettez à jour vos enregistrements DNS de sorte que votre nom d'hôte se résolve toujours vers l'adresse IP du nouvel hôte si c'est un déplacement permanent.

Félicitations !

Si vous avez suivi les étapes soulignées ici vous devriez maintenant avoir une copie totalement fonctionnelle de votre installation Elgg d'origine.

Connexe

Script de backup FTP

Voici un script automatisé pour sauvegarder une installation Elgg.

```
#!/usr/bin/perl -w

# FTP Backup

use Net::FTP;

# DELETE BACKUP AFTER FTP UPLOAD (0 = no, 1 = yes)
$delete_backup = 1;

# ENTER THE PATH TO THE DIRECTORY YOU WANT TO BACKUP, NO TRAILING SLASH
$directory_to_backup = '/home/userx/public_html';
$directory_to_backup2 = '/home/userx/elggdata';

# ENTER THE PATH TO THE DIRECTORY YOU WISH TO SAVE THE BACKUP FILE TO, NO TRAILING SLASH
$backup_dest_dir = '/home/userx/sitebackups';

# BACKUP FILE NAME OPTIONS
($a,$d,$d,$day,$month,$yearoffset,$r,$u,$o) = localtime();
$year = 1900 + $yearoffset;
$site_backup_file = "$backup_dest_dir/site_backup-$day-$month-$year.tar.gz";
$full_backup_file = "$backup_dest_dir/full_site_backup-$day-$month-$year.tar.gz";

# MYSQL BACKUP PARAMETERS
$dbhost = 'localhost';
$dbuser = 'userx_elgg';
$dbpwd = 'dbpassword';
$mysql_backup_file_elgg = "$backup_dest_dir/mysql_elgg-$day-$month-$year.sql.gz";
```

(suite sur la page suivante)

(suite de la page précédente)

```

# ENTER DATABASE NAME
$database_names_elgg = 'userx_elgg';

# FTP PARAMETERS
$ftp_backup = 1;
$ftp_host = "FTP HOSTNAME/IP";
$ftp_user = "ftpuser";
$ftp_pwd = "ftppassword";
$ftp_dir = "/";

# SYSTEM COMMANDS
$cmd_mysql_dump = '/usr/bin/mysqldump';
$cmd_gzip = '/usr/bin/gzip';

# CURRENT DATE / TIME
($a,$d,$d,$day,$month,$yearoffset,$r,$u,$o) = localtime();
$year = 1900 + $yearoffset;

# BACKUP FILES
$syscmd = "tar --exclude $backup_dest_dir" . "/* -czf $site_backup_file $directory_to_
↳ backup $directory_to_backup2";

# elgg DATABASE BACKUP
system($syscmd);
$syscmd = "$cmd_mysql_dump --host=$dbhost --user=$dbuser --password=$dbpwd --add-drop-
↳ table --databases $database_names_elgg -c -l | $cmd_gzip > $mysql_backup_file_elgg";

system($syscmd);

# CREATING FULL SITE BACKUP FILE
$syscmd = "tar -czf $full_backup_file $mysql_backup_file_elgg $site_backup_file";
system($syscmd);

# DELETING SITE AND MYSQL BACKUP FILES
unlink($mysql_backup_file_elgg);
unlink($site_backup_file);

# UPLOADING FULL SITE BACKUP TO REMOTE FTP SERVER
if($ftp_backup == 1)
{
    my $ftp = Net::FTP->new($ftp_host, Debug => 0)
        or die "Cannot connect to server: $@";

    $ftp->login($ftp_user, $ftp_pwd)
        or die "Cannot login ", $ftp->message;

    $ftp->cwd($ftp_dir)
        or die "Can't CWD to remote FTP directory ", $ftp->message;

    $ftp->binary();

    $ftp->put($full_backup_file)

```

(suite sur la page suivante)

(suite de la page précédente)

```

    or warn "Upload failed ", $ftp->message;

    $ftp->quit();
}

# DELETING FULL SITE BACKUP
if($delete_backup = 1)
{
    unlink($full_backup_file);
}

```

Dupliquer une installation

Contenu

- *Introduction*
- *Pourquoi dupliquer une installation Elgg ?*
- *Ce qui n'est pas traité dans ce tutoriel*
- *Avant de commencer*
- *Copiez le code d'Elgg vers le serveur de test*
- *Copiez les données d'Elgg vers le serveur de test*
- *Editez settings.php*
- *Copier la base de données d'Elgg*
- *Entrées de la base de données*
 - *Modifiez le chemin d'installation*
 - *Modifiez le répertoire de données*
- *Vérifier le fichier .htaccess*
- *Mettez à jour la configuration du serveur web*
- *Exécutez upgrade.php*
- *Astuces*
- *Connexe*

Introduction

Pourquoi dupliquer une installation Elgg ?

Il existe beaucoup de raisons pour lesquelles vous pouvez vouloir dupliquer une installation Elgg : déplacer le site vers un autre serveur, créer un serveur de test ou de développement, et créer des backups fonctionnels sont les plus habituelles. Pour créer un double fonctionnel d'un site Elgg, 3 choses doivent être copiées :

- Base de données
- Données du répertoire de données
- Code

Il faut modifier au moins 5 éléments d'information pour une installation copiée :

- le fichier `elgg-config/settings.php` qui peut aussi se trouver à l'emplacement `engine/settings.php` pour les versions pré-2.0
- le fichier `.htaccess` (Apache) ou la configuration de Nginx en fonction du serveur utilisé
- l'entrée de la base de donnée pour l'entité de votre site
- l'entrée de la base de donnée pour le chemin d'installation

- l'entrée de la base de donnée pour le chemin des données

Ce qui n'est pas traité dans ce tutoriel

Ce tutoriel nécessite une connaissance minimale d'Apache, MySQL, et des commandes Linux. A ce titre, certaines choses ne seront pas couvertes dans ce tutoriel. Ceci comprend :

- Comment sauvegarder (backup) et restaurer des bases de données MySQL
- Comment configurer Apache pour fonctionner avec Elgg
- Comment transférer des fichiers vers et depuis votre serveur de production

Avant de commencer

Avant de commencer, assurez-vous que votre installation Elgg est pleinement fonctionnelle. Vous aurez aussi besoin des éléments suivants :

- Un backup de la base de données de votre site Elgg actif
- Un emplacement pour copier la base de données active
- **Un serveur qui convienne pour installer le site Elgg dupliqué** (Qui peut être le même serveur que celui de votre installation Elgg en production)

Les backups de la base de données peuvent être obtenus de diverses manières, y compris via phpMyAdmin, l'interface officielle de MySQL, et la ligne de commande. Contactez votre hébergeur pour des informations sur comment sauvegarder et restaurer des bases de données, ou utilisez un moteur de recherche pour trouver de l'information à ce sujet.

Au cours de ce tutoriel, nous allons faire les hypothèses suivante à propos du site Elgg en production :

- L'URL est `http://www.myselgg.org/`
- Le chemin d'installation est `/var/www/elgg/`
- Le chemin du répertoire de données est `/var/data/elgg/`
- L'hôte de la base de données est `localhost`
- Le nom de la base de données est `production_elgg`
- L'utilisateur de la base de données est `db_user`
- Le mot de passe de la base de données est `db_password`
- Le préfixe des tables de la base de données est `elgg`

A la fin du tutoriel, les renseignements de notre installation Elgg de test seront :

- L'URL est `http://test.myselgg.org/`
- Le chemin d'installation est `/var/www/elgg_test/`
- Le répertoire de données est `/var/data/elgg_test/`
- L'hôte de la base de données est `localhost`
- Le nom de la base de données est `test_elgg`
- L'utilisateur de la base de données est `db_user`
- Le mot de passe de la base de données est `db_password`
- Le préfixe des tables de la base de données est `elgg`

Copiez le code d'Elgg vers le serveur de test

La toute première étape est de dupliquer le code du site Elgg de production. Dans notre exemple, c'est aussi simple que de copier `/var/www/elgg/` vers `/var/www/elgg_test/`.

```
cp -a /var/www/elgg/ /var/www/elgg_test/
```

Copiez les données d'Elgg vers le serveur de test

Dans cet exemple, c'est aussi simple que copier `/var/data/elgg/` vers `/var/data/elgg_test/`.

```
cp -a /var/data/elgg/ /var/data/elgg_test/
```

Si vous n'avez pas d'accès console (shell) à votre serveur et devez transférer les données par FTP, vous pouvez avoir besoin de modifier le propriétaire et les permissions des fichiers.

Note : Vous devez aussi supprimer les dossier de cache du disque. Ceux-ci correspondent aux dossiers `cacheroot` et `assetroot` dans votre configuration.

Editez settings.php

Le fichier `elgg-config/settings.php` contient les renseignements de configuration de la base de données. Ceux-ci doivent être ajustés pour votre nouvelle installation Elgg de test. Dans notre exemple, nous allons regarder dans `/var/www/elgg_test/elgg-config/settings.php` et trouver les lignes qui ressemblent à ceci :

```
// Database username
$CONFIG->dbuser = 'db_user';

// Database password
$CONFIG->dbpass = 'db_password';

// Database name
$CONFIG->dbname = 'elgg_production';

// Database server
// (For most configurations, you can leave this as 'localhost')
$CONFIG->dbhost = 'localhost';
// (For most configurations, you can leave this as 3306)
$CONFIG->dbport = 3306;

// Database table prefix
// If you're sharing a database with other applications, you will want to use this
// to differentiate Elgg's tables.
$CONFIG->dbprefix = 'elgg';
```

Nous devons modifier ces lignes pour qu'elles correspondent à notre nouvelle installation :

```
// Database username
$CONFIG->dbuser = 'db_user';
```

(suite sur la page suivante)

(suite de la page précédente)

```
// Database password
$CONFIG->dbpass = 'db_password';

// Database name
$CONFIG->dbname = 'elgg_test';

// Database server
// (For most configurations, you can leave this as 'localhost')
$CONFIG->dbhost = 'localhost';
// (For most configurations, you can leave this as 3306)
$CONFIG->dbport = 3306;

// Database table prefix
// If you're sharing a database with other applications, you will want to use this
// to differentiate Elgg's tables.
$CONFIG->dbprefix = 'elgg';

$CONFIG->wwwroot = 'http://your.website.com/'
```

Note : Notez que \$CONFIG->dbname a changé pour correspondre à notre nouvelle base de données.

Copier la base de données d'Elgg

Maintenant la base de données doit être copiée depuis `elgg_production` vers `elgg_test`. Voyez la documentation de votre gestionnaire MySQL préféré pour savoir comment dupliquer une base de données. Vous allez généralement exporter les tables de la base de donnée actuelle vers un fichier, créer la nouvelle base de données, puis importer les tables que vous avez précédemment exportées.

Vous avez deux possibilités pour mettre à jour les valeurs dans la base de données. Vous pourriez changer les valeurs dans le fichier exporté, ou vous pourriez modifier ces valeurs par des requêtes sur la base de données. Un avantage de modifier directement le fichier exporté est que vous pouvez aussi modifier les liens que les utilisateurs ont créé vers du contenu au sein de votre site. Par exemple, si des personnes ont mis en marque-page des pages avec le plugin bookmarks, les marque-pages vont pointer vers l'ancien site tant que vous n'aurez pas mis à jour leurs URLs.

Entrées de la base de données

Nous devons maintenant modifier 4 entrées dans la base de données. Ceci peut être fait aisément en exécutant 4 commandes SQL simples :

Modifiez le chemin d'installation

```
UPDATE `elgg_config` SET `value` = REPLACE(`value`, "/var/www/elgg_production/", "/var/
↳ www/elgg_test/") WHERE `name` = "path";
```

Modifiez le répertoire de données

```
UPDATE `elgg_config` SET `value` = REPLACE(`value`, "/var/data/elgg_production/", "/var/
↳ data/elgg_test/") WHERE `name` = "dataroot";
```

Vérifier le fichier .htaccess

Si vous aviez apporté des modifications au fichier .htaccess qui modifient des chemins, assurez-vous que vous les mettez à jour dans l'installation de test.

Mettez à jour la configuration du serveur web

Pour cet exemple, vous devez modifier la config Apache pour activer un sous-domaine avec pour racine des documents (document root) /var/www/elgg_test/. Si vous prévoyez d'installer dans un sous-répertoire de votre racine des documents, cette étape n'est pas nécessaire.

Si vous utilisez Nginx, vous aurez besoin de mettre à jour la config du serveur pour correspondre aux nouveaux chemins basés sur install/config/nginx.dist.

Exécutez upgrade.php

Pour régénérer les données mises en cache, assurez-vous d'exécuter <http://test.myclgg.org/upgrade.php>

Astuces

C'est une bonne idée de garder un serveur de test à portée de main pour expérimenter l'installation de nouveaux plugins et faire du développement. Si vous automatisez les restaurations de la base de données elgg_test, modifier les valeurs de \$CONFIG et ajouter les lignes suivantes à la fin du fichier elgg_test/elgg-config/settings.php va permettre de réécrire sans couture les entrées de la base de données MySQL.

```
$con = mysql_connect($CONFIG->dbhost, $CONFIG->dbuser, $CONFIG->dbpass);
mysql_select_db($CONFIG->dbname, $con);

$sql = "UPDATE {$CONFIG->dbprefix}config
  SET value = REPLACE(`value`, "/var/www/elgg_production/", "/var/www/elgg_test/")
  WHERE name = 'path'";
```

(suite sur la page suivante)

(suite de la page précédente)

```
mysql_query($sql);
print mysql_error();

$sql = "UPDATE {$CONFIG->dbprefix}config
    SET value = REPLACE(`value`, "/var/data/elgg_production/", "/var/data/elgg_test/")
    WHERE name = 'dataroot'";
mysql_query($sql);
print mysql_error();
```

Connexe

Voir aussi :

Sauvegarde et Restauration

3.2.8 Trouver de l'aide

Vous avez un problème avec Elgg ? Le meilleur moyen d'obtenir de l'aide est de demander sur le [Site de la Communauté](#). Ce site communautaire est animé par un grand nombre de volontaires. Voici quelques astuces pour vous aider à trouver l'aide dont vous avez besoin.

Contenu

- *Trouver de l'aide*
- *Recommandations*
- *Bonnes idées*

Trouver de l'aide

Ne soyez pas un **`Vampire de l'Aide`** _

Nous avons tous été des débutants à un moment, mais nous pouvons apprendre. Le fait de ne pas montrer que vous faites des tentatives pour apprendre par vous-même ou faites vos propres recherches est décourageant pour ceux qui aident. De plus, il est quasiment impossible de répondre à des questions très génériques comme « Comment je construis un forum ? ».

Faites d'abord des recherches

Assurez-vous de faire des recherches dans la documentation (ce site), sur le [Site de la Communauté](#), et dans un moteur de recherche avant de poser une question. Les nouveaux utilisateurs d'Elgg se posent souvent les mêmes questions, aussi merci de faire des recherches. Les membres sont moins enclins à répondre à une demande à laquelle il a déjà été répondu maintes fois ou à laquelle on peut facilement trouver une réponse via un moteur de recherche.

Demandez une seule fois

Poser les mêmes questions à plusieurs endroits rend plus difficile le fait de vous répondre. Posez votre questions à un seul endroit. Les questions dupliquées peuvent être modérées.

Indiquez la version d'Elgg

Les différentes versions d'Elgg ont différentes fonctionnalités (et différents bugs). Préciser la version d'Elgg que vous utilisez va aider les personnes qui aident.

Ayez un profil raisonnable

Les profils qui ressemblent à du spam ou ont des noms loufoques seront souvent ignorés. La jovialité est bienvenue, mais les membres ont plus de chance d'aider Michael que 1337elggHax0r.

Publiez dans le forum approprié

Assurez-vous de publier dans le bon forum. Si vous avez une question sur comment créer un plugin, n'écrivez pas dans le forum Elgg Feedback. Si vous avez besoin d'aide pour installer Elgg, écrivez dans le groupe Technical Support (support technique) et pas dans le groupe Theming (thèmes).

Utilisez un titre de sujet descriptif

De bons titres de sujets décrivent de manière concise votre problème ou question. De mauvais titres sont vagues, ne contiennent que les lettres capitales, et une ponctuation excessive.

Bon titre : « Écran blanc après la mise à niveau vers 1.7.4 »

Mauvais titre : « URGENT!!!! site cassé;-(je perds de l'argent à l'aide!!!!!!!!!! »

Soyez détaillé

Ajoutez autant de détails que possible à propos de votre problème. Si vous avez un site en ligne, incluez un lien. Soyez accueillant si des membres de la communauté vous demandent plus d'informations. Nous ne pouvons pas vous aider si vous ne donnez pas de détails !

Gardez cela public

C'est un forum public pour le bien du projet Elgg. Gardez les sujets publics. Il n'y a aucune raison pour qui que ce soit de vous envoyer des messages privés ou des emails. De même, il n'y a aucune raison de demander à qui que ce soit de vous envoyer un email privé. Publiez en public.

Recommandations

En plus des [mentions légales et de la charte du site](#), suivre des recommandations aident à garder notre site communautaire utile et sûr pour tout le monde.

Contenu

Tous les contenus doivent être tous publics : « PG » (Parental Guidance - classification des contenus adultes) aux USA et au Royaume-Uni. Si votre site Elgg a du contenu adulte et qu'on vous a demandé de publier un lien, veuillez le marquer « NSFW » (Not Safe For Work) de sorte que les personnes le sachent.

L'utilisation excessive de jurons dans n'importe quelle langue ne sera pas tolérée.

Humeur

Travailler avec des problèmes techniques peut être frustrant. Veuillez laisser le site de la communauté libre de frustration. Si vous ressentez de l'anxiété, passez le cap et faites autre chose. Menacer ou attaquer des membres de la communauté, des développeurs du noyau, ou des développeurs de plugins n'aidera pas à résoudre votre problème et risque très probablement de vous faire bannir.

Publicité

Les annonces publicitaires ne sont pas autorisées. La publication de tout type d'annonce sera modérée.

Demander de l'argent / Offrir de payer

Ne demandez pas d'argent sur le site de la communauté. De même, ne proposez pas de payer pour des réponses. Si vous recherchez des développements sur mesure, publiez dans le groupe « Professional Services » (Services Professionnels). Les publications qui demandent de l'argent ou recommandent un plugin commercial peuvent être modérées.

Liens

Si vous avez un problème avec un site en ligne, merci de fournir un lien vers ce site.

Ceci dit, la communauté n'est pas un service de backlinks ni un outil SEO. Un excès de liens sera modéré et votre compte peut être banni.

Signatures

Il y a une raison pour laquelle Elgg n'a pas d'option pour les signatures : elles causent du désordre et détournent l'attention de la conversation. Il est recommandé aux utilisateurs de ne pas utiliser de signature sur le site communautaire, et les signatures avec des liens ou de la publicité seront retirées.

Relancer, +1, moi aussi

Ne faites pas cela. Si votre question n'a pas reçu de réponse, voyez le début de ce document pour des astuces. Ces types de réponses n'ajoutent rien à la conversation et peuvent être modérées.

Publier du Code

De longs extraits de code rendent la lecture confuse dans le cadre d'un forum. Veuillez utiliser <http://elgg.pastebin.com> pour publier de longs extraits de code et donnez le lien PasteBin au lieu de publier directement le code.

Bonnes idées

Pas de règles strictes, mais des bonnes idées.

Dites merci

Est-ce que quelqu'un vous a aidé ? Assurez-vous de remercier cette personne : le site de la communauté est animé par des volontaires. Personne n'a la devoir de vous aider avec votre problème. Assurez-vous de montrer votre reconnaissance !

Donnez en retour

Vous avez une astuce pour Elgg ? Vous voyez quelqu'un qui a le même problème que vous aviez eu ? Vous êtes passé par là et pouvez les aider, alors donnez-leur un coup de main !

3.2.9 Sécurité

A partir de Elgg 3.0 plusieurs paramètres de sécurisation ont été ajoutés à Elgg. Vous pouvez activer/désactiver ces paramètres comme vous le souhaitez.

Contenu

- *Protection des mises à niveau*
- *Protection du Cron*
- *Désactiver l'autocomplétion du mot de passe*
- *Le changement d'adresse email requiert le mot de passe*
- *Le changement d'adresse e-mail nécessite une confirmation*
- *Icônes liées à la session*
- *Notification aux administrateurs du site*
- *Notifications à l'utilisateur*
 - *Administrateur du site*
 - *Bannir/Réintégrer*
- *Longueur minimale du nom d'utilisateur*
- *Exigences minimales en matière de mot de passe*
- *Sécurisation de l'accès aux fichiers via .htaccess*

Protection des mises à niveau

L'URL de <http://your-elgg-site.com/upgrade.php> peut être protégée par un jeton unique. Ceci évite que des utilisateurs aléatoires aient la possibilité d'exécuter ce fichier. Le jeton n'est pas nécessaire pour les administrateurs du site connectés.

Protection du Cron

Les URLs du `cron` peuvent être protégées par un jeton unique. Ceci évite que des utilisateurs aléatoires puissent exécuter le cron. Le jeton n'est pas nécessaire que le cron est lancé depuis la ligne de commande du serveur.

Désactiver l'autocomplétion du mot de passe

Les données saisies dans ces champs seront mises en cache par le navigateur. Un attaquant qui a accès au navigateur de la victime pourrait subtiliser ces informations. Ceci est particulièrement important si l'application est utilisée habituellement depuis des ordinateurs partagés tels que des cybercafés ou des terminaux d'aéroports. Si vous désactivez ceci, les outils de gestion des mots de passe ne pourront plus renseigner automatiquement ces champs. Le support pour l'attribut « autocomplete » peut dépendre des navigateurs.

Le changement d'adresse email requiert le mot de passe

Quand un utilisateur souhaite changer l'adresse email associée à son compte, il doit également donner son mot de passe actuel.

Le changement d'adresse e-mail nécessite une confirmation

Lorsqu'un utilisateur souhaite modifier l'adresse e-mail associée à son compte, il doit confirmer la nouvelle adresse e-mail. Cela se fait en envoyant un e-mail à la nouvelle adresse avec un lien de validation. Après avoir cliqué sur ce lien, la nouvelle adresse e-mail sera utilisée.

Icônes liées à la session

Les icônes d'entité peuvent être liées par défaut à une session. Cela signifie que les URL générées contiennent également des informations sur la session en cours. Le fait d'avoir des icônes liées à la session rend les URL des icônes non partageables entre les sessions. L'effet secondaire est que la mise en cache de ces URL ne bénéficiera qu'à la session active.

Notification aux administrateurs du site

Quand un nouvel administrateur de site est ajouté ou quand un administrateur du site est retiré tous les administrateurs du site reçoivent une notification à ce sujet.

Notifications à l'utilisateur

Administrateur du site

Quand le rôle d'administrateur du site est ajouté ou retiré à un compte, envoie une notification à l'utilisateur dont le compte est affecté.

Bannir/Réintégrer

Quand le compte d'un utilisateur est banni ou réintégré, permet à l'utilisateur affecté d'être informé de cette action.

Longueur minimale du nom d'utilisateur

Vous pouvez configurer la longueur minimale que le nom d'utilisateur doit avoir lors de l'inscription d'un utilisateur.

Exigences minimales en matière de mot de passe

Vous pouvez configurer plusieurs exigences pour les nouveaux mots de passe des utilisateurs

- **longueur** : le mot de passe doit comporter au moins x caractères
- **minuscule** : nombre minimal de caractères minuscules (a-z) dans le mot de passe
- **majuscule** : nombre minimal de caractères majuscules (A-Z) dans le mot de passe
- **nombres** : nombre minimal de nombres (0-9) dans le mot de passe
- **specials** : nombre minimal de caractères spéciaux (comme !@##\$%^&* etc.) dans le mot de passe

Sécurisation de l'accès aux fichiers via .htaccess

Dans le fichier .htaccess, un ensemble de règles de durcissement de l'accès aux fichiers a été ajouté pour empêcher l'accès direct aux fichiers dans certains dossiers. L'activation de ces règles *ne devrait pas* causer des problèmes lorsque tous les plugins que vous utilisez suivent les directives de codage Elgg.

Voici quelques exemples de règles :

- le dossier `vendor`. Ce dossier contient uniquement des bibliothèques qu'Elgg utilise et il n'est pas nécessaire d'accéder directement à ce dossier. Toutes les dépendances requises sont chargées à partir de Elgg
- le dossier `languages`. Ce dossier contient les principaux fichiers linguistiques d'Elgg. Ces fichiers sont chargés à partir de Elgg

3.2.10 Validation des utilisateurs

Les plugins peuvent influencer la manière dont les utilisateurs sont validés avant de pouvoir utiliser le site.

Contenu

- *Liste des utilisateurs non validés*
- *Exiger la validation de l'administrateur*

Liste des utilisateurs non validés

Dans la section Admin du site vous trouverez une liste des utilisateurs non validés. Plusieurs actions peuvent être effectuées sur les utilisateurs, telles que les supprimer du système ou les valider.

Les plugins ont la possibilité d'ajouter de nouvelles fonctionnalités à cette liste.

Voir aussi :

Un exemple de ceci est le plugin *Validation des comptes utilisateur via l'email* qui ne permet pas aux utilisateurs de se connecter sur le site tant que leur adresse email n'est pas validée.

Exiger la validation de l'administrateur

Dans les paramètres du site sous la section Utilisateurs, il existe un paramètre qui peut être activé pour exiger la validation par l'administrateur d'un nouveau compte utilisateur avant que celui-ci puisse utiliser son compte. Après l'enregistrement, l'utilisateur est informé que son compte est en attente de validation par un administrateur.

Les administrateurs de site peuvent recevoir une notification par email indiquant qu'il y a des utilisateurs en attente de validation.

Après validation, l'utilisateur est notifié qu'il peut utiliser son compte.

3.2.11 Spam

Gardez le spam sous contrôle.

Le spam est un problème très commun que les administrateurs doivent traiter et que nous visons à améliorer dans le noyau, mais il y a quelques mesures que les administrateurs de site peuvent prendre pour atténuer le problème.

Installez un plugin anti-spam

Il y en a plusieurs disponible dans la communauté

<http://community.elgg.org/plugins/search?f{{c}}{{}}{}{}=spam&sb=Search>

Modifier l'url d'inscription

Certaines personnes ont dit avoir obtenu de bons résultats en changeant l'URL d'inscription, puisque les spammeurs sont des bots naïfs qui ne peuvent plus trouver où créer les faux comptes.

Désactiver l'inscription libre

Si le flux d'inscriptions est assez faible, vous pouvez vérifier tous les utilisateurs qui s'inscrivent pour conserver une qualité d'interaction élevée.

Contribuer aux mesures anti-span dans le noyau

<https://github.com/Elgg/Elgg/issues?labels=spam&state=open>

3.3 Guides de développement

Personnalisez le comportement de Elgg avec des plugins.

3.3.1 Ne modifiez pas le cœur

Avertissement : De manière générale, vous ne devriez pas modifier de fichier autre que ceux de configuration fournis avec des logiciels tiers comme Elgg.

La meilleure façon de personnaliser le comportement de Elgg est d’*installer Elgg en tant que dépendance composer* et d’utiliser le répertoire racine pour stocker les modifications spécifiques à votre application, et modifier le comportement à travers la riche API plugin de Elgg.

Si vous souhaitez partager des personnalisations entre les sites ou même publier vos modifications en tant que package réutilisable pour la communauté, créez un :doc : *plugin<plugins>* qui utilise la même API de plugin et la même structure de fichiers.

Cela rend difficile d’obtenir de l’aide

Lorsque vous ne partagez pas la même base de code que tout le monde, il est impossible pour les autres de savoir ce qui se passe dans votre système et si vos modifications sont à blâmer. Cela peut frustrer ceux qui offrent de l’aide parce que cela complique considérablement le processus de soutien.

Cela rend la mise à niveau délicate et potentiellement désastreuse

Vous voudrez certainement ou aurez besoin de mettre à niveau Elgg pour profiter de

- patches de sécurité
- nouvelles fonctionnalités
- nouvelles APIs de plugin
- nouvelle amélioration de la stabilité
- amélioration des performances

Si vous avez modifié des fichiers du noyau, alors vous devez être très prudent lors de la mise à niveau que vos modifications ne sont pas remplacées et qu’elles sont compatibles avec le nouveau code de Elgg. Si vos modifications sont perdues ou incompatibles, la mise à niveau peut supprimer les fonctionnalités que vous avez ajoutées et même briser complètement votre site.

Cela peut également être une pente glissante. Beaucoup de modifications peuvent vous conduire à un processus de mise à niveau si complexe qu’il en devient pratiquement impossible. Il existe beaucoup de sites coincés avec d’anciennes versions du logiciel à cause du choix de cette voie.

Cela peut casser des plugins

Vous pourriez ne réaliser que beaucoup plus tard que votre « solution rapide » a cassé des fonctionnalités apparemment sans rapport dont les plugins dépendaient.

Résumé

- **Résistez à la tentation** L'édition des fichiers existants est rapide et facile, mais cela compromet fortement la maintenance, la sécurité et la stabilité de votre site.
- Lorsque vous recevez des conseils, considérez si la personne qui vous dit de modifier le noyau sera là pour vous sauver si vous rencontrez des ennuis plus tard !
- **Appliquez ces principes aux logiciels en général.** Si vous pouvez l'éviter, ne modifiez pas non plus les plugins tiers, pour les mêmes raisons : les auteurs de plugin publient également de nouvelles versions, et vous voudrez pouvoir utiliser ces mises à jour.

3.3.2 Listes de Contrôle d'Accès

Une Liste de Contrôle d'Accès (ACL - Access Control Lists) permet d'octroyer à un ou plusieurs utilisateurs l'accès à une entité ou à une annotation dans la base de données.

Contenu

- *Créer une ACL*
- *Sous-types d'ACL*
- *Ajouter des membres à une ACL*
- *Retirer des membres d'une ACL*
- *Récupérer une ACL*
- *Accès en lecture*
- *Ignorer l'accès*

Voir aussi :

Database Access Control

Créer une ACL

Une collection d'accès peut être créée en utilisant la fonction `create_access_collection()`.

```
$owner_guid = elgg_get_logged_in_user_guid();  
$acl = create_access_collection("Sample name", $owner_guid, 'collection_subtype');
```

Sous-types d'ACL

Les ACLs peuvent avoir un sous-type, ceci permet de différencier différents usages de l'ACL. Il est fortement recommandé de définir un sous-type pour un ACL.

Le noyau d'Elgg a trois exemples d'usage de sous-types

- `group_acl` un ACL détenu par un `ElggGroup` qui donne aux membres du groupe l'accès aux contenus partagés dans le groupe
- `friends` un ACL détenu par un `ElggUser` qui donne aux contacts de l'utilisateur l'accès aux contenus partagés avec les contacts
- `friends_collection` un ACL détenu par un `ElggUser` qui donne à des contacts spécifiques l'accès aux contenus partagés avec cet ACL

Ajouter des membres à une ACL

Si vous avez un ACL vous devez encore lui ajouter des utilisateurs pour donner accès à ces utilisateurs au contenu avec l'`access_id` de l'ACL `id`.

```
// creating an ACL
$owner_guid = elgg_get_logged_in_user_guid();

$acl_id = create_access_collection("Sample name", $owner_guid, 'collection_subtype');

// add other user (procedural style)
add_user_to_access_collection($some_user_guid, $acl_id);

// add other user (object oriented style)
/* @var $acl ElggAccessCollection */
$acl = get_access_collection($acl_id);

$acl->addMember($some_other_user_guid);
```

Retirer des membres d'une ACL

Si vous souhaitez ne plus donner l'accès à un utilisateur donné dans un ACL, vous pouvez facilement retirer cet utilisateur de la liste.

```
// remove a user from an ACL (procedural style)
remove_user_from_access_collection($user_guid_to_be_removed, $acl_id);

// remove a user from an ACL (object oriented style)
/* @var $acl ElggAccessCollection */
$acl = get_access_collection($acl_id);

$acl->removeMember($user_guid_to_be_removed);
```

Récupérer une ACL

Afin de gérer un ACL, ou ajouter l’ID d’un ACL à une liste d’accès il existe plusieurs fonctions disponibles pour récupérer un ACL depuis la base de données.

```
// get ACL based on known id
$acl = get_access_collection($acl_id);

// get all ACLs of an owner (procedural style)
$acls = elgg_get_access_collections([
    'owner_guid' => $some_owner_guid,
]);

// get all ACLs of an owner (object oriented style)
$acls = $some_owner_entity->getOwnedAccessCollections();

// add a filter for ACL subtype
// get all ACLs of an owner (procedural style)
$acls = elgg_get_access_collections([
    'owner_guid' => $some_owner_guid,
    'subtype' => 'some_subtype',
]);

// get all ACLs of an owner (object oriented style)
$acls = $some_owner_entity->getOwnedAccessCollections([
    'subtype' => 'some_subtype',
]);

// get one ACL of an owner (object oriented style)
// for example the group_acl of an ElggGroup
// Returns the first ACL owned by the entity with a given subtype
$acl = $group_entity->getOwnedAccessCollection('group_acl');
```

Accès en lecture

Le système d’accès d’Elgg ajoute automatiquement toutes les ACL dont un utilisateur est membre aux contrôles d’accès. Par exemple, si un utilisateur est membre d’un groupe et est en contact avec 3 autres utilisateurs, tous les ACL correspondants sont ajoutés afin de vérifier l’accès aux entités lors de leur récupération (par exemple, la liste de tous les articles de blog).

Ignorer l’accès

Si dans certains cas vous avez besoin d’entités récupérées en ignorant les règles d’accès, vous pouvez envelopper votre code dans `elgg_call`. Il existe différents drapeaux que vous pouvez utiliser.

- `ELGG_IGNORE_ACCESS` : aucune règle d’accès n’est appliquée
- `ELGG_ENFORCE_ACCESS` : les règles d’accès sont forcées à être appliquées
- `ELGG_SHOW_DISABLED_ENTITIES` : récupérera les entités désactivées
- `ELGG_HIDE_DISABLED_ENTITIES` : ne récupérera jamais les entités désactivées

```
$options = [
    'type' => 'user'
];
```

(suite sur la page suivante)

(suite de la page précédente)

```
$entities = elgg_call(ELGG_IGNORE_ACCESS, function() use ($options) {
    return elgg_get_entities($options);
});
```

Vous pouvez également combiner des drapeaux.

```
$entities = elgg_call(ELGG_IGNORE_ACCESS | ELGG_SHOW_DISABLED_ENTITIES, function() {
    return elgg_get_entities([
        'type' => 'user'
    ]);
});
```

3.3.3 Accessibilité

Cette page a pour but d'énumérer et de documenter les règles d'accessibilité et les meilleures pratiques pour aider les développeur du noyau et des pugins à faire d'Elgg le meilleur cadre applicatif de moteur social dont tout le monde rêve.

Note : Il s'agit d'un travail en cours, veuillez contribuer sur [Github](#) si vous avez des compétences dans ce domaine !

Ressources + références

- Aperçu des guides d'accessibilité officiels WCAG - Official WCAG Accessibility Guidelines Overview
- Guides d'accessibilité officiels WCAG - Official WCAG Accessibility Guidelines
- Ressources pour préparer et implémenter l'accessibilité
- Astuces pratiques du W3C pour améliorer l'accessibilité
- Revue préliminaire de sites pour l'accessibilité
- Outils pour vérifier l'accessibilité des sites web
- Liste de techniques pratiques pour implémenter l'accessibilité (Ce serait génial si quelqu'un pouvait les examiner et filtrer celles qui sont pertinentes pour Elgg)

Astuces pour implémenter l'accessibilité

- Tous les tickets relatifs à l'accessibilité devraient être tagués avec « a11y », raccourci pour « accessibility » (accessibilité)
- Utilisez les vues du cœur telles que `output/*`, et `input/*` pour générer le balisage, dans la mesure où nous pouvons intégrer les règles d'accessibilité dans ces vues
- Toutes les images doivent avoir un attribut descriptif `alt`. Les graphiques d'espacement ou purement décoratifs doivent avoir un attribut `alt` vide
- Toutes les balises `<a>` doivent avoir du texte ou une image accessible à l'intérieur. Sinon, les lecteurs d'écran devront lire l'URL, ce qui est une mauvaise expérience. Les balises `<a>` devraient contenir du texte descriptif, si possible, par opposition au texte générique comme « Cliquez ici »
- Le balisage devrait être valide
- Les thèmes de doivent pas réinitialiser « outline » à rien. :focus mérite un traitement visuel spécial afin que les utilisateurs handicapés puissent savoir où ils se trouvent

Astuces pour tester l'accessibilité

- Utilisez les outils disponibles à partir de la section ressources. [Exemple de rapport pour community.elgg.org du 16 juin 2012](#)
- Essayez différentes configurations de taille de police et de zoom dans votre navigateur pour être sûr que votre thème reste utilisable
- Désactivez css pour vous assurer que l'ordre séquentiel de la page est logique

Objectifs et principes de la documentation

- Principales règles d'accessibilité
- collecter et documenter les bonnes pratiques
- Fournissez des exemples de code
- Conservez le document simple et utilisable
- Rendez-le utilisable à la fois pour les développeurs débutants et les experts (des changements les plus courants et les plus faciles aux techniques élaborées)

3.3.4 Formulaires + Actions

Créer, mettre à jour, ou supprimer du contenu.

Les formulaires Elgg sont envoyés aux actions. Les actions définissent le comportement pour l'envoi des formulaires.

Ce guide suppose une familiarité de base avec :

- *Plugins*
- *Vues*
- *Internationalisation*

Contenu

- *Enregistrer des actions*
 - *Enregistrer des actions en utilisant le fichier de configuration du plugin*
 - *Permissions*
 - *Écrire des fichiers d'action*
 - *Personnaliser des actions*
- *Actions disponibles dans le noyau*
 - *entity/delete*
- *Formulaire*
 - *Entrées*
 - *Types d'entrées*
- *Fichiers et images*
- *Formulaire persistant*
 - *Fonctions pratiques*
 - *Aperçu*
 - *Exemple : Inscription d'un utilisateur*
 - *Exemple : Signets (bookmarks)*
- *Ajax*
- *Sécurité*
- *Jetons de sécurité*
- *URLs signées*

Enregistrer des actions

Les actions doivent être enregistrées avant de pouvoir être utilisées.

Il existe deux manières d'enregistrer des actions :

Utilisez `elgg_register_action()`

```
elgg_register_action("example", __DIR__ . "/actions/example.php");
```

Le fichier de script `mod/example/actions/example.php` sera maintenant utilisé à chaque fois qu'un formulaire est envoyé vers `http://localhost/elgg/action/example`.

Utilisez `elgg-plugin.php`

```
return [
    'actions' => [
        // defaults to using an action file in /actions/myplugin/action_a.php
        'myplugin/action_a' => [
            'access' => 'public',
        ],

        // define custom action path
        'myplugin/action_b' => [
            'access' => 'admin',
            'filename' => __DIR__ . '/actions/action.php'
        ],

        // define a controller
        'myplugin/action_c' => [
            'controller' => \MyPlugin\Actions\ActionC::class,
        ],
    ],
];
```

Avvertissement : Un point d'achoppement pour de nombreux nouveaux développeurs est l'URL pour les actions. L'URL utilise toujours `/action/` (singulier) et jamais `/actions/` (pluriel). Toutefois, les fichiers de script d'action sont généralement enregistrés dans le répertoire `/actions/` (pluriel) et ont toujours une extension. Utilisez `elgg_generate_action_url()` pour éviter toute confusion.

Enregistrer des actions en utilisant le fichier de configuration du plugin

Vous pouvez également enregistrer des actions via le *elgg-plugin* fichier de configuration. Pour cela vous devez fournir une section action dans le fichier de configuration. L'emplacement des fichiers d'action est situé dans le dossier `/actions` du plugin.

```
<?php

return [
    'actions' => [
        'blog/save' => [], // all defaults
        'blog/delete' => [] // all custom
```

(suite sur la page suivante)

(suite de la page précédente)

```

        'access' => 'admin',
        'filename' => __DIR__ . 'actions/blog/remove.php',
    ],
],
];

```

Permissions

Par défaut, les actions ne sont accessibles qu'aux utilisateurs connectés.

Pour qu'une action soit disponible pour des utilisateurs non identifiés, passez "public" comme troisième argument :

```
elgg_register_action("example", $filepath, "public");
```

Pour restreindre l'action aux seuls administrateurs, passez "admin" pour le dernier paramètre :

```
elgg_register_action("example", $filepath, "admin");
```

Écrire des fichiers d'action

Utilisez la fonction `get_input()` pour accéder aux paramètres de la requête :

```
$field = get_input('input_field_name', 'default_value');
```

Vous pouvez ensuite utiliser l'api *Base de données* pour charger des entités et effectuer des actions dessus en conséquence.

Pour indiquer une action réussie, utilisez `elgg_ok_response()`. Cette fonction accepte comme paramètre les données que vous souhaitez mettre à la disposition du client pour les appels XHR (ces données seront ignorées pour les appels non XHR)

```

$user = get_entity($guid);
// do something

$action_data = [
    'entity' => $user,
    'stats' => [
        'friends' => $user->getFriends(['count' => true]);
    ],
];

return elgg_ok_response($action_data, 'Action was successful', 'url/to/forward/to');
```

Pour indiquer une erreur utilisez `elgg_error_response()`

```

$user = elgg_get_logged_in_user_entity();
if (!$user) {
    // show an error and forward the user to the referring page
    // send 404 error code on AJAX calls
    return elgg_error_response('User not found', REFERER, ELGG_HTTP_NOT_FOUND);
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

if (!$user->canEdit()) {
    // show an error and forward to user's profile
    // send 403 error code on AJAX calls
    return elgg_error_response('You are not allowed to perform this action', $user->
    ↪getURL(), ELGG_HTTP_FORBIDDEN);
}

```

Personnaliser des actions

Avant d'exécuter une action, Elgg déclenche un hook :

```
$result = elgg_trigger_plugin_hook('action:validate', $action, null, true);
```

Où \$action est l'action appelée. Si le hook renvoie false alors l'action ne sera pas exécutée. Ne retournez rien si votre validation passe.

Exemple : Captcha

Le module captcha utilise ceci pour intercepter les actions register et user/requestnewpassword et les rediriger vers une fonction qui vérifie le code captcha. Cette vérification renvoie false si la validation captcha échoue (ce qui bloque l'exécution de l'action associée).

Ceci est fait comme suit :

```

elgg_register_plugin_hook_handler("action:validate", "register", "captcha_verify_action_
↪hook");
elgg_register_plugin_hook_handler("action:validate", "user/requestnewpassword", "captcha_
↪verify_action_hook");

...

function captcha_verify_action_hook(\Elgg\Hook $hook) {
    $token = get_input('captcha_token');
    $input = get_input('captcha_input');

    if (($token) && (captcha_verify_captcha($input, $token))) {
        return;
    }

    register_error(elgg_echo('captcha:captchafail'));

    return false;
}

```

Cela permet à un plugin d'étendre une action existante sans qu'il soit nécessaire de remplacer l'ensemble de l'action. Dans le cas du plugin captcha, cela permet de proposer un support captcha couplé de façon souple.

Actions disponibles dans le noyau

entity/delete

Si votre plugin n'implémente aucune logique personnalisée lors de la suppression d'une entité, vous pouvez utiliser l'action de suppression groupée

```
$guid = 123;
// You can provide optional forward path as a URL query parameter
$forward_url = 'path/to/forward/to';
echo elgg_view('output/url', array(
    'text' => elgg_echo('delete'),
    'href' => elgg_generate_action_url('entity/delete', [
        'guid' => $guid,
        'forward_url' => $forward_url,
    ]),
    'confirm' => true,
));
```

Vous pouvez personnaliser les clés du message de succès pour votre type d'entité et votre sous-type, en utilisant les clés "entity:\$type:\$subtype:success" et "'entity:delete:\$type:success'".

```
// to add a custom message when a blog post or file is deleted
// add the translations keys in your language files
return [
    'entity:delete:object:blog:success' => 'Blog post has been deleted',
    'entity:delete:object:file:success' => 'File titled %s has been deleted',
];
```

Formulaires

Pour afficher un formulaire, utilisez la fonction `elgg_view_form` fonction comme ceci :

```
echo elgg_view_form('example');
```

Faire ceci génère quelque chose comme le balisage suivant :

```
<form action="http://localhost/elgg/action/example">
  <fieldset>
    <input type="hidden" name="__elgg_ts" value="1234567890" />
    <input type="hidden" name="__elgg_token" value="3874acfc283d90e34" />
  </fieldset>
</form>
```

Elgg fait automatiquement plusieurs choses pour vous quand vous générez un formulaire de cette manière :

1. Il définit l'action vers l'URL appropriée, sur la base du nom de l'action que vous lui avez passé
2. Il ajoute des jetons anti-csrf (`__elgg_ts` et `__elgg_token`) pour aider à garder vos action sûres
3. Il recherche automatiquement le corps du formulaire dans la vue `forms/example`.

Ajoutez le contenu de votre formulaire dans la vue `forms/example` de votre plugin :

```
// /mod/example/views/default/forms/example.php
echo elgg_view('input/text', array('name' => 'example'));

// defer form footer rendering
// this will allow other plugins to extend forms/example view
elgg_set_form_footer(elgg_view('input/submit'));
```

Désormais quand vous appelez `elgg_view_form('example')`, Elgg va produire :

```
<form action="http://localhost/elgg/action/example">
  <fieldset>
    <input type="hidden" name="__elgg_ts" value="...">
    <input type="hidden" name="__elgg_token" value="...">

    <input type="text" class="elgg-input-text" name="example">
    <div class="elgg-foot elgg-form-footer">
      <input type="submit" class="elgg-button elgg-button-submit" value="Submit">
    </div>
  </fieldset>
</form>
```

Entrées

Pour rendre une entrée de formulaire, utilisez l'une des vues d'entrée intégrées, qui couvrent tous les éléments d'entrée HTML standard. Voir les fichiers individuels des vues pour une liste de paramètres acceptés.

```
echo elgg_view('input/select', array(
  'required' => true,
  'name' => 'status',
  'options_values' => [
    'draft' => elgg_echo('status:draft'),
    'published' => elgg_echo('status:published'),
  ],
  // most input views will render additional parameters passed to the view
  // as tag attributes
  'data-rel' => 'blog',
));
```

L'exemple ci-dessus va afficher une liste déroulante :

```
<select required="required" name="status" data-rel="blog" class="elgg-input-select">
  <option value="draft">Draft</option>
  <option value="published">Published</option>
</select>
```

Pour assurer la cohérence du balisage du champ, utilisez les paramètres `elgg_view_field()`, qui accepte tous les paramètres de l'entrée à afficher, ainsi que les paramètres `#label` et `#help` (qui sont tous deux facultatifs et acceptent du HTML ou du texte).

```
echo elgg_view_field([
  '#type' => 'select',
  '#label' => elgg_echo('blog:status:label'),
```

(suite sur la page suivante)

(suite de la page précédente)

```

'#help' => elgg_view_icon('help') . elgg_echo('blog:status:help'),
'required' => true,
'name' => 'status',
'options_values' => [
    'draft' => elgg_echo('status:draft'),
    'published' => elgg_echo('status:published'),
],
'data-rel' => 'blog',
]);

```

Ce qui précède générera le balisage suivant :

```

<div class="elgg-field elgg-field-required">
  <label for="elgg-field-1" class="elgg-field-label">Blog status<span title="Required"
  ↳class="elgg-required-indicator">*</span></label>
  <div class="elgg-field-input">
    <select required="required" name="status" data-rel="blog" id="elgg-field-1" class=
    ↳"elgg-input-select">
      <option value="draft">Draft</option>
      <option value="published">Published</option>
    </select>
  </div>
  <div class="elgg-field-help elgg-text-help">
    <span class="elgg-icon-help elgg-icon"></span>This indicates whether or not the
    ↳blog is visible in the feed
  </div>
</div>

```

Types d'entrées

Une liste des types/vues de saisie groupés :

- input/text - affiche une entrée texte <input type="text">
- input/plaintext - affiche une zone de texte <textarea></textarea>
- input/longtext - affiche une zone de texte riche (éditeur WYSIWYG)
- input/url - affiche une entrée de type adresse web (URL) <input type="url">
- input/email - affiche une entrée de type email <input type="email">
- input/checkbox - affiche une case à cocher <input type="checkbox">
- input/checkboxes - affiche un jeu de cases à cocher portant le même nom
- input/radio - affiche un ou plusieurs bouton(s) <input type="radio">
- input/submit - affiche un bouton d'envoi <button type="submit">
- input/button - affiche un bouton <button></button>
- input/file - affiche un sélecteur de fichier <input type="file">
- input/select - affiche une liste déroulante <select></select>
- input/hidden - affiche une entrée invisible <input type="hidden">
- input/password - affiche une entrée de type mot de passe <input type="password">
- input/number - affiche une entrée de type nombre <input type="number">
- input/date - affiche un sélecteur de date jQuery

Elgg offre quelques types d'aide à la saisie

- input/access - affiche une liste de niveaux d'accès Elgg
- input/tags - affiche une entrée de type tags
- input/autocomplete - affiche un sélecteur d'entités Elgg

- `input/captcha` - vue destinée à être étendue par des plugins
- `input/friendpicker` - affiche un sélecteur de contact Elgg par autocomplétion
- `input/userpicker` - affiche un sélecteur d'utilisateur Elgg
- `input/location` affiche une entrée de type adresse

Fichiers et images

Utilisez la vue `input/file` dans la vue du contenu de votre formulaire.

```
// /mod/example/views/default/forms/example.php
echo elgg_view('input/file', ['name' => 'icon']);
```

Si vous souhaitez télécharger une icône pour l'entité, vous pouvez utiliser la vue `entity/edit/icon`. Cette vue affiche une entrée de fichier pour le téléchargement d'une nouvelle icône pour l'entité, une miniature de l'icône actuelle et l'option de supprimer l'icône actuelle.

La vue prend en charge certaines variables pour contrôler la sortie

- `entity` - l'entité à laquelle ajouter ou supprimer l'icône. Si fourni en fonction de cette entité, la miniature et l'option de suppression s'affichent
- `entity_type` - le type d'entité pour lequel l'icône sera téléchargée. Des plugins pourraient trouver cela utile, peut-être pour valider les dimensions d'icônes
- `entity_subtype` - le sous-type d'entité pour lequel l'icône sera téléchargée. Des plugins pourraient trouver cela utile, peut-être pour valider les dimensions d'icônes
- `icon_type` - le type de l'icône (par défaut : `icon`)
- `name` - nom de `input/file` (par défaut : `icon`)
- `remove_name` - nom de l'icône de bascule de suppression (par défaut : `$vars["name"]."_remove"`)
- `required` - est-ce que l'icône de téléchargement est requise (par défaut : `false`)
- `show_remove` - afficher l'option supprimer l'icône (par défaut : `true`)
- `show_thumb` - afficher la miniature de l'entité si disponible (par défaut : `true`)
- `thumb_size` - la taille de l'icône à utiliser pour la miniature (par défaut : `medium`)

Si vous utilisez la vue, vous pouvez utiliser le code suivant dans votre action pour enregistrer l'icône dans l'entité ou supprimer l'icône actuelle.

```
if (get_input('icon_remove')) {
    $entity->deleteIcon();
} else {
    $entity->saveIconFromUploadedFile('icon');
}
```

Définissez le type d'encodage du formulaire sur `multipart/form-data` :

```
echo elgg_view_form('example', array(
    'enctype' => 'multipart/form-data'
));
```

Note : Le type d'encodage `enctype` de tous les formulaires qui utilisent la méthode POST est par défaut `multipart/form-data`.

Dans votre fichier d'action, utilisez `elgg_get_uploaded_file('nom-de-votre-input')` pour accéder au fichier téléchargé :

```
$icon = elgg_get_uploaded_file('icon');
```

Formulaires persistants

Les formulaires persistants sont des formulaires qui conservent l'entrée de l'utilisateur en cas d'échec de l'enregistrement. Ils sont « persistants » parce que les données de l'utilisateur « persistent » dans le formulaire après soumission, bien qu'elles n'aient jamais été enregistrées dans la base de données. Cela améliore considérablement l'expérience utilisateur en minimisant la perte de données. Elgg inclut des fonctions d'aide afin que vous puissiez rendre n'importe quel formulaire persistant.

Fonctions pratiques

Les formulaires persistants sont implémentés dans Elgg par les fonctions suivantes :

- `elgg_make_sticky_form($name)` - Indique au moteur de persister les valeurs de toutes les entrées d'un formulaire.
- `elgg_clear_sticky_form($name)` - Indique au moteur de supprimer toutes les valeurs d'un formulaire.
- `elgg_is_sticky_form($name)` - Vérifie si \$name est un formulaire persistant valide.
- `elgg_get_sticky_values($name)` - Renvoie toutes les valeurs persistantes enregistrées pour \$name par `elgg_make_sticky_form($name)`.

Aperçu

Le déroulement de l'utilisation des formulaires persistants est :

1. Appelez `elgg_make_sticky_form($name)` au début des actions pour les formulaires que vous voulez rendre persistants.
2. Utilisez `elgg_is_sticky_form($name)` et `elgg_get_sticky_values($name)` pour obtenir des valeurs persistantes lors du rendu d'une vue de formulaire.
3. Appelez `elgg_clear_sticky_form($name)` une fois l'action terminée avec succès ou après que les données ont été chargées par `elgg_get_sticky_values($name)`.

Exemple : Inscription d'un utilisateur

Les formulaires persistants simples nécessitent peu de logique pour déterminer les valeurs d'entrée du formulaire. Cette logique est placée en haut de la vue du corps de formulaire.

L'affichage du formulaire d'inscription définit d'abord les valeurs par défaut pour les entrées, puis vérifie s'il y a des valeurs persistantes. Si c'est le cas, il charge les valeurs persistantes avant de supprimer le formulaire persistant :

```
// views/default/forms/register.php
$password = $password2 = '';
$username = get_input('u');
$email = get_input('e');
$name = get_input('n');

if (elgg_is_sticky_form('register')) {
    extract(elgg_get_sticky_values('register'));
    elgg_clear_sticky_form('register');
}
```

Les ensembles d'action d'inscription créent le formulaire persistant et l'effacent une fois l'action terminée :

```
// actions/register.php
elgg_make_sticky_form('register');

...

$guid = register_user($username, $password, $name, $email, false, $friend_guid,
↳ $invitecode);

if ($guid) {
    elgg_clear_sticky_form('register');
    ....
}
```

Exemple : Signets (bookmarks)

Le formulaire et l'action d'enregistrement du plugin Bookmarks (Signets) est un exemple de formulaire persistant complexe.

La vue du formulaire pour l'action d'enregistrement d'un signet utilise `elgg_extract()` pour extraire des valeurs du tableau `$vars` :

```
// mod/bookmarks/views/default/forms/bookmarks/save.php
$title = elgg_extract('title', $vars, '');
$desc = elgg_extract('description', $vars, '');
$address = elgg_extract('address', $vars, '');
$tags = elgg_extract('tags', $vars, '');
$access_id = elgg_extract('access_id', $vars, ACCESS_DEFAULT);
$container_guid = elgg_extract('container_guid', $vars);
$guid = elgg_extract('guid', $vars, null);
$shares = elgg_extract('shares', $vars, array());
```

Les scripts du gestionnaire de page préparent les variables de formulaire et appellent `elgg_view_form()` en passant les valeurs correctes :

```
// mod/bookmarks/pages/add.php
$vars = bookmarks_prepare_form_vars();
$content = elgg_view_form('bookmarks/save', array(), $vars);
```

De même, `mod/bookmarks/pages/edit.php` utilise la même fonction, mais passe l'entité qui est en cours de modification comme argument :

```
$bookmark_guid = get_input('guid');
$bookmark = get_entity($bookmark_guid);

...

$vars = bookmarks_prepare_form_vars($bookmark);
$content = elgg_view_form('bookmarks/save', array(), $vars);
```

Le fichier de bibliothèque définit `bookmarks_prepare_form_vars()`. Cette fonction accepte une `ElggEntity` comme argument et fait 3 choses :

1. Définit les noms et les valeurs par défaut pour les champs de formulaire.
2. Extrait les valeurs d'un objet signet s'il est passé.

3. Extrait les valeurs d'un formulaire persistant si il existe.

```
// mod/bookmarks/lib/bookmarks.php
function bookmarks_prepare_form_vars($bookmark = null) {
    // input names => defaults
    $values = array(
        'title' => get_input('title', ''), // bookmarklet support
        'address' => get_input('address', ''),
        'description' => '',
        'access_id' => ACCESS_DEFAULT,
        'tags' => '',
        'shares' => array(),
        'container_guid' => elgg_get_page_owner_guid(),
        'guid' => null,
        'entity' => $bookmark,
    );

    if ($bookmark) {
        foreach (array_keys($values) as $field) {
            if (isset($bookmark->$field)) {
                $values[$field] = $bookmark->$field;
            }
        }
    }

    if (elgg_is_sticky_form('bookmarks')) {
        $sticky_values = elgg_get_sticky_values('bookmarks');
        foreach ($sticky_values as $key => $value) {
            $values[$key] = $value;
        }
    }

    elgg_clear_sticky_form('bookmarks');

    return $values;
}
```

L'action save vérifie l'entrée, puis efface le formulaire persistant en cas de réussite :

```
// mod/bookmarks/actions/bookmarks/save.php
elgg_make_sticky_form('bookmarks');

...

if ($bookmark->save()) {
    elgg_clear_sticky_form('bookmarks');
}
```


Ajax

Voir le *guide Ajax* pour savoir comment appeler les actions depuis JavaScript.

Sécurité

Pour une sécurité renforcée, toutes les actions nécessitent un jeton CSRF. Les appels aux URLs d'action qui n'incluent pas les jetons de sécurité seront ignorés et un avertissement sera généré.

Quelques vues et fonctions génèrent automatiquement des jetons de sécurité :

```
elgg_view('output/url', array('is_action' => true));
elgg_view('input/securitytoken');
$url = elgg_add_action_tokens_to_url("http://localhost/elgg/action/example");
$url = elgg_generate_action_url('myplugin/myaction');
```

Dans de rares cas, vous devrez peut-être générer des jetons manuellement :

```
$_elgg_ts = elgg()->csrf->getCurrentTime()->getTimestamp();
$_elgg_token = elgg()->csrf->generateActionToken($_elgg_ts);
```

Vous pouvez aussi accéder aux jetons par javascript :

```
elgg.security.token.__elgg_ts;
elgg.security.token.__elgg_token;
```

Ceux-ci sont rafraîchis périodiquement aussi ils devraient être à jour.

Jetons de sécurité

À l'occasion, nous devons transmettre des données par l'intermédiaire d'une tierce partie non digne de confiance ou générer un jeton impossible à deviner basé sur certaines données. L'algorithme HMAC est le bon outil pour cela. Il nous permet de vérifier que les données reçues ont été générées par notre site, et n'ont pas été trafiqués. Notez que même les fonctions de hachage fortes comme SHA-2 ne devraient *pas* être utilisés sans HMAC pour ces tâches.

Elgg fournit `elgg_build_hmac()` pour générer et valider des codes d'authentification des messages HMAC qui ne peuvent être pas devinés sans la clé privée du site.

```
// generate a querystring such that $a and $b can't be altered
$a = 1234;
$b = "hello";
$query = http_build_query([
    'a' => $a,
    'b' => $b,
    'mac' => elgg_build_hmac([$a, $b])->getToken(),
]);
$url = "action/foo?$query";

// validate the querystring
$a = (int) get_input('a', '', false);
$b = (string) get_input('b', '', false);
$mac = get_input('mac', '', false);
```

(suite sur la page suivante)

(suite de la page précédente)

```
if (elgg_build_hmac([$a, $b])->matchesToken($mac)) {
    // $a and $b have not been altered
}
```

Remarque : si vous utilisez une non-chaîne comme des données HMAC, vous devez utiliser les types de manière cohérente. Considérez ce qui suit :

```
$mac = elgg_build_hmac([123, 456])->getToken();

// type of first array element differs
elgg_build_hmac(["123", 456])->matchesToken($mac); // false

// types identical to original
elgg_build_hmac([123, 456])->matchesToken($mac); // true
```

URLs signées

Les URL signées offrent un niveau de sécurité limité pour les situations où les jetons d'action ne conviennent pas, par exemple lors de l'envoi d'un lien de confirmation par e-mail. Les signatures de l'URL vérifient que l'URL a été générée par votre installation Elgg (en utilisant le secret du site) et que les éléments de requête URL n'ont pas été trafiqués.

URLs signées avec une clef SHA-256 HMAC impossible à deviner. Pour plus de détails, consultez *Jetons de Sécurité*.

```
$url = elgg_http_add_url_query_element(elgg_normalize_url('confirm'), [
    'user_guid' => $user_guid,
]);

$url = elgg_http_get_signed_url($url);

notify_user($user_guid, $site->guid, 'Confirm', "Please confirm by clicking this link:
↪ $url");
```

Avertissement : Les URL signées n'offrent pas de protection CSRF et ne doivent pas être utilisées à la place des jetons d'action.

3.3.5 Ajax

Le module AMD elgg/Ajax (introduit dans Elgg 2.1) fournit un ensemble de méthodes pour communiquer avec le serveur d'une manière concise et uniforme, ce qui permet aux plugins de collaborer sur les données de requête, la réponse du serveur et les données renvoyées côté client.

Le code client et serveur écrit pour l'API d'origine ne doit pas avoir besoin de modification.

Contenu

- *Aperçu*
- *Effectuer des actions*
- *Récupérer des données*
- *Récupérer des vues*

- *Récupérer des formulaires*
- *Envoyer des formulaires*
- *Redirections*
- *Réagir (piggybacking) lors d'une requête Ajax*
- *Réagir (piggybacking) à une réponse Ajax*
- *Gérer les erreurs*
- *Requérir des modules AMD*
- *APIs elgg.ajax historiques*
 - *elgg.action d'origine*
 - *Récupération d'une vue héritée*
 - *Récupération de formulaire d'origine*
 - *Fonctions d'aide d'origine*

Aperçu

Toutes les méthodes ajax effectuent ce qui suit :

1. Côté client, l'option `data` (si elle est donnée en tant qu'objet) est filtrée par le hook `ajax_request_data`.
2. La requête est faite au serveur, que ce soit en affichant une vue ou un formulaire, en appelant une action, ou en chargeant un chemin d'accès.
3. La méthode renvoie un objet `jqXHR`, qui peut être utilisé comme promesse (Promise).
4. Le contenu généré par le serveur est transformé en objet de réponse (`Elgg\Services\AjaxResponse`) contenant une chaîne (ou une valeur parsée en JSON).
5. L'objet de réponse est filtré par le hook `ajax_response`.
6. L'objet de réponse est utilisé pour créer la réponse HTTP.
7. Côté client, les données de réponse sont filtrées par le hook `ajax_response_data`.
8. La promesse (promise) `jqXHR` est résolue et toutes les fonctions de rappel `success` sont appelées.

Plus de notes :

- Tous les hooks ont un type en fonction de la méthode et du premier argument. Voyez ci-dessous.
- Par défaut, le module `elgg/spinner` est automatiquement utilisé lors des requêtes.
- Les messages destinés à l'utilisateur générés par `system_message()` et `register_error()` sont collectés et affichés sur le client.
- Elgg intègre un gestionnaire d'erreurs par défaut qui affiche un message générique en cas d'échec de la sortie.
- Les exceptions PHP ou les codes d'erreur HTTP d'accès refusé, ce qui entraîne l'utilisation du gestionnaire d'erreurs côté client.
- La méthode HTTP par défaut est POST pour les actions, sinon GET. Vous pouvez changer cela via `options.method`.
- Si une valeur `options.data` non vide est donnée, la méthode par défaut est toujours POST.
- Pour la mise en cache côté client, définissez la méthode `options.method` sur GET et `options.data.elgg_response_ttl` sur l'âge maximal que vous souhaitez en secondes.
- Pour enregistrer des messages système pour le chargement de la page suivante, définissez `options.data.elgg_fetch_messages = 0`. Vous pourriez vouloir faire cela si vous avez l'intention de rediriger l'utilisateur en fonction de la réponse.
- Pour empêcher l'API côté client d'exiger des modules AMD requis côté serveur avec `elgg_require_js()`, définissez `options.data.elgg_fetch_deps = 0`.
- Toutes les méthodes acceptent une chaîne de requête dans le premier argument. Cette chaîne est transmise à l'URL de récupération, mais n'apparaît pas dans les types de hooks.

Effectuer des actions

Considérez cette action :

```
// in myplugin/actions/do_math.php

elgg_ajax_gatekeeper();

$arg1 = (int)get_input('arg1');
$arg2 = (int)get_input('arg2');

// will be rendered client-side
system_message('We did it!');

echo json_encode([
    'sum' => $arg1 + $arg2,
    'product' => $arg1 * $arg2,
]);
```

Pour l'exécuter, utilisez `ajax.action('<action_name>', options)` :

```
var Ajax = require('elgg/Ajax');
var ajax = new Ajax();

ajax.action('do_math', {
    data: {
        arg1: 1,
        arg2: 2
    },
}).done(function (output, textStatus, jqXHR) {
    if (jqXHR.AjaxData.status == -1) {
        return;
    }

    alert(output.sum);
    alert(output.product);
});
```

Notes pour les actions :

- **Tous les hooks ont le type `action:<action_name>`. Donc, dans ce cas, trois hooks seront déclenchés :**
 - côté client `"ajax_request_data"`, `"action:do_math"` pour filtrer les données de la requête (avant qu'elle soit envoyée)
 - côté serveur `"ajax_response"`, `"action:do_math"` pour filtrer la réponse (après que l'action a été exécutée)
 - côté client `"ajax_response_data"`, `"action:do_math"` pour filtrer les données de la réponse (avant que le code appelant ne les reçoive)
- Les jetons CSRF sont ajoutés aux données de la requête.
- La méthode par défaut est POST.
- Une URL d'action absolue peut être donnée à la place du nom d'action.
- L'utilisation de `forward()` dans une action envoie simplement la réponse. L'URL indiquée n'est pas renvoyée au client.

Note : Lorsque vous définissez `data`, utilisez `ajax.objectify($form)` au lieu de `$form.serialize()`. Cela per-

met au hook de plugin `ajax_request_data` d’être déclenché, et à d’autres plugins de modifier / réagir à la requête.

Récupérer des données

Considérez ce script PHP qui s’exécute à `http://example.org/myplugin_time`.

```
// in myplugin/elgg-plugin.php
return [
    'routes' => [
        'default:myplugin:time' => [
            'path' => '/myplugin_time',
            'resource' => 'myplugin/time',
        ],
    ],
];

// in myplugin/views/default/resources/myplugin/time.php
elgg_ajax_gatekeeper();

echo json_encode([
    'rfc2822' => date(DATE_RFC2822),
    'day' => date('l'),
]);

return true;
```

Pour récupérer sa sortie, utilisez `ajax.path('<url_path>', options)`.

```
var Ajax = require('elgg/Ajax');
var ajax = new Ajax();

ajax.path('myplugin_time').done(function (output, statusText, jqXHR) {
    if (jqXHR.AjaxData.status == -1) {
        return;
    }

    alert(output.rfc2822);
    alert(output.day);
});
```

Notes pour les chemins :

- Les 3 hooks (voir Actions ci-dessus) auront le type `path:<url_path>`. Dans ce cas, « chemin :myplugin_time ».
- Si le gestionnaire de page renvoie une page Web normale, `output` sera une chaîne contenant le HTML de la page.
- Une URL absolue peut être donnée à la place du nom du chemin.

Récupérer des vues

Considérez cette vue :

```
// in myplugin/views/default/myplugin/get_link.php

if (empty($vars['entity']) || !$vars['entity'] instanceof ElggObject) {
    return;
}

$object = $vars['entity'];
/* @var ElggObject $object */

echo elgg_view('output/url', [
    'text' => $object->getDisplayName(),
    'href' => $object->getUrl(),
    'is_trusted' => true,
]);
```

Comme il s'agit d'un fichier PHP, nous devons d'abord l'enregistrer pour Ajax :

```
// in myplugin_init()
elgg_register_ajax_view('myplugin/get_link');
```

Pour récupérer la vue, utilisez `ajax.view('<view_name>', options)` :

```
var Ajax = require('elgg/Ajax');
var ajax = new Ajax();

ajax.view('myplugin/get_link', {
    data: {
        guid: 123 // querystring
    },
}).done(function (output, statusText, jqXHR) {
    if (jqXHR.AjaxData.status == -1) {
        return;
    }

    $('.myplugin-link').html(output);
});
```

Notes pour les vues :

- Les 3 hooks (voir Actions ci-dessus) auront le type `view:<view_name>`. Dans ce cas, « `view :myplugin/get_link` ».
- `output` sera une chaîne avec la vue générée.
- Les données de la requête sont injectées dans `$vars` dans la vue.
- Si les données de la requête contiennent `guid`, le système définit `$vars['entity']` à l'entité correspondante ou à `false` si elle ne peut pas être chargée.

Avertissement : Dans les vues et les formulaires ajax, notez que `$vars` peut être peuplé par les saisies du client. Les données sont filtrées comme `get_input()`, mais peuvent ne pas avoir le type que vous attendez ou peuvent avoir des clefs inattendues.

Récupérer des formulaires

Considérez que nous avons une vue de formulaire. Nous l'enregistrons pour Ajax :

```
// in myplugin_init()
elgg_register_ajax_view('forms/myplugin/add');
```

Pour récupérer ceci utilisez `ajax.form('<action_name>', options)`.

```
var Ajax = require('elgg/Ajax');
var ajax = new Ajax();

ajax.form('myplugin/add').done(function (output, statusText, jqXHR) {
    if (jqXHR.AjaxData.status == -1) {
        return;
    }

    $('.myplugin-form-container').html(output);
});
```

Notes pour les formulaires :

- Les 3 hooks (voir Actions ci-dessus) auront le type `form:<action_name>`. Dans ce cas, « `form :myplugin/add` ».
- `output` sera une chaîne avec la vue générée.
- Les données de la requête sont injectées dans `$vars` dans la vue de votre formulaire.
- Si les données de la requête contiennent `guid`, le système définit `$vars['entity']` à l'entité correspondante ou à `false` si elle ne peut pas être chargée.

Note : Seules les données de la requête sont transmises à la vue du formulaire demandé (c'est-à-dire en tant que troisième paramètre accepté par `elgg_view_form()`). Si vous devez passer des attributs ou des paramètres de l'élément de formulaire rendu par la vue `input/form` (c'est-à-dire normalement passé comme deuxième paramètre à `elgg_view_form()`), utilisez le hook serveur `view_vars`, `input/form`.

Avertissement : Dans les vues et les formulaires ajax, notez que `$vars` peut être peuplé par les saisies du client. Les données sont filtrées comme `get_input()`, mais peuvent ne pas avoir le type que vous attendez ou peuvent avoir des clefs inattendues.

Envoyer des formulaires

Pour envoyer un formulaire avec Ajax, il suffit de passer le paramètre `ajax` avec les variables de formulaire :

```
echo elgg_view_form('login', ['ajax' => true]);
```

Redirections

Utilisez `ajax.forward()` pour démarrer un spinner et rediriger l'utilisateur vers une nouvelle destination.

```
var Ajax = require('elgg/Ajax');
var ajax = new Ajax();
ajax.forward('/activity');
```

Réagir (piggybacking) lors d'une requête Ajax

Le hook `ajax_request_data` côté client peut être utilisé pour ajouter ou filtrer les données envoyées par une demande `elgg/Ajax`.

Disons que lorsque la vue `foo` est récupérée, nous voulons également envoyer au serveur quelques données :

```
// in your boot module
var Ajax = require('elgg/Ajax');
var elgg = require('elgg');

var ajax = new Ajax();

elgg.register_hook_handler(Ajax.REQUEST_DATA_HOOK, 'view:foo', function (name, type, ↵
↵params, data) {
    // send some data back
    data.bar = 1;
    return data;
});
```

Ces données peuvent être lues côté serveur via `get_input('bar');`.

Note : Si les données ont été données sous forme de chaîne (par ex. `$form.serialize()`), les hooks de requête ne sont pas déclenchés.

Note : Le formulaire sera objectivé sous le nom de `FormData`, et le type de contenu de la requête sera déterminé en conséquence. En effet, cela permet aux plugins de soumettre des données de formulaire multipart sans utiliser le plugin `jquery.form` plugin et autres hacks de type `iframe`.

Réagir (piggybacking) à une réponse Ajax

Le hook `ajax_response` côté serveur peut être utilisé pour ajouter ou filtrer des données de réponse (ou des méta-données).

Disons que lorsque la vue `foo` est récupérée, nous voulons également envoyer au client quelques données supplémentaires :

```
use Elgg\Services\AjaxResponse;

function myplugin_append_ajax($hook, $type, AjaxResponse $response, $params) {

    // alter the value being returned
    $response->getData()->value .= " hello";

    // send some metadata back. Only client-side "ajax_response" hooks can see this!
    $response->getData()->myplugin_alert = 'Listen to me!';

    return $response;
}

// in myplugin_init()
elgg_register_plugin_hook_handler(AjaxResponse::RESPONSE_HOOK, 'view:foo', 'myplugin_
↳append_ajax');
```

Pour capturer les métadonnées envoyées au client, nous utilisons le hook `ajax_response` côté client :

```
// in your boot module
var Ajax = require('elgg/Ajax');
var elgg = require('elgg');

elgg.register_hook_handler(Ajax.RESPONSE_DATA_HOOK, 'view:foo', function (name, type, ␣
↳params, data) {

    // the return value is data.value

    // the rest is metadata

    alert(data.myplugin_alert);

    return data;
});
```

Note : Seule la valeur `data.value` est renvoyée à la fonction `success`, ou disponible via l'interface *Deferred*.

Note : Elgg utilise ces mêmes hooks pour diffuser des messages système via les réponses `elgg/Ajax`.

Gérer les erreurs

Les réponses correspondent à l'une de ces trois catégories :

1. Succès HTTP (200) avec un statut 0. Aucun appel `register_error()` n'a été fait sur le serveur.
2. Succès HTTP (200) avec le statut -1. `register_error()` a été appelé.
3. Erreur HTTP (4xx/5xx). Par ex. appeler une action avec des jetons expirés, ou une exception sur le serveur.
Dans ce cas les fonctions de callbacks `done` et `success` ne sont pas appelées.

Vous devriez ne vous inquiéter que du 2e cas. Ceci peut être fait en regardant à `jqXHR.AjaxData.status` :

```
ajax.action('entity/delete?guid=123').done(function (value, textStatus, jqXHR) {  
    if (jqXHR.AjaxData.status == -1) {  
        // a server error was already displayed  
        return;  
    }  
  
    // remove element from the page  
});
```

Requérir des modules AMD

Chaque réponse d'un service Ajax contiendra une liste de modules AMD requis côté serveur avec `elgg_require_js()`. Lorsque les données de réponse sont déballées, ces modules seront chargés de façon asynchrone - les plugins ne doivent pas s'attendre à ce que ces modules soient chargés dans leurs gestionnaires `$.done()` et `$.then()` et doivent utiliser `require()` pour tous les modules dont ils dépendent. En outre, les modules AMD ne doivent pas s'attendre à ce que le DOM ait été modifié par une demande Ajax lorsqu'ils sont chargés - les événements DOM doivent être délégués et les manipulations sur les éléments DOM doivent être retardées jusqu'à ce que toutes les requêtes Ajax aient été résolues.

APIs elgg.ajax historiques

Elgg 1.8 a introduit `elgg.action`, `elgg.get`, `elgg.getJSON`, et d'autres méthodes qui se comportent de manière moins uniforme côté client et côté serveur.

elgg.action d'origine

Différences :

- vous devez récupérer manuellement `output` à partir du wrapper renvoyé
- le gestionnaire de `success` sera exécuté même si l'action est bloquée
- le gestionnaire `success` recevra un objet `wrapper`. Vous devez rechercher `wrapper.output`
- pas de hook ajax

```
elgg.action('do_math', {  
    data: {  
        arg1: 1,  
        arg2: 2  
    },  
    success: function (wrapper) {  
        if (wrapper.output) {  
            alert(wrapper.output.sum);  
            alert(wrapper.output.product);  
        }  
    }  
});
```

(suite sur la page suivante)

(suite de la page précédente)

```

    } else {
        // the system prevented the action from running, but we really don't
        // know why
        elgg.ajax.handleAjaxError();
    }
}
});

```

notes sur elgg.action

- Il est préférable de renvoyer une chaîne non vide, car cela facilite la validation dans la fonction `success`. Si l'action n'a pas été autorisée à s'exécuter pour une raison quelconque, `wrapper.output` sera une chaîne vide.
- Vous pouvez utiliser le module *elgg/spinner*.
- Elgg n'utilise pas `wrapper.status` pour quoi que ce soit, mais un appel à `register_error()` définit sa valeur sur -1.
- Si l'action renvoie une chaîne non-JSON, `wrapper.output` va emballer cette chaîne.
- `elgg.action` est basé sur `jQuery.ajax` et renvoie un objet `jqXHR` (comme `Promise`), si vous voulez l'utiliser.
- Une fois l'action PHP terminée, d'autres plugins peuvent modifier l'emballage via le hook plugin 'output', 'ajax', qui filtre l'emballage comme un tableau (pas une chaîne JSON).
- Un appel `forward()` force le traitement de l'action et renvoie la sortie immédiatement, avec la valeur `wrapper.forward_url` définie sur l'emplacement normalisé donné.
- Pour vous assurer que les actions Ajax ne peuvent être exécutées que via XHR, utilisez `elgg_ajax_gatekeeper()`.

emballage de réponse JSON de elgg.action

Avertissement : Il est probablement préférable de s'appuyer uniquement sur la clef `output`, et de la valider au cas où l'action PHP ne pourrait pas s'exécuter pour une raison quelconque, par exemple parce que l'utilisateur a été déconnecté ou parce qu'une attaque CSRF n'a pas fourni de jetons.

Avertissement : Si `forward()` est utilisé dans une réponse à une demande ajax héritée (par exemple `elgg.ajax`), Elgg répondra *toujours* avec ce wrapper, **même si ce n'est pas dans une action**.

Récupération d'une vue héritée

Un plugin peut utiliser un script d'affichage pour gérer les demandes XHR GET. Voici un exemple simple d'une vue qui renvoie un lien vers un objet donné par son GUID :

```

// in myplugin_init()
elgg_register_ajax_view('myplugin/get_link');

```

```

// in myplugin/views/default/myplugin/get_link.php

if (empty($vars['entity']) || !$vars['entity'] instanceof ElggObject) {
    return;
}

```

(suite sur la page suivante)

(suite de la page précédente)

```

}

$object = $vars['entity'];
/* @var ElggObject $object */

echo elgg_view('output/url', [
    'text' => $object->getDisplayname(),
    'href' => $object->getUrl(),
    'is_trusted' => true,
]);

```

```

elgg.get('ajax/view/myplugin/get_link', {
    data: {
        guid: 123 // querystring
    },
    success: function (output) {
        $('#myplugin-link').html(output);
    }
});

```

Le système de vues Ajax fonctionne de manière significativement différente du système d'action.

- Il n'y a pas de contrôle d'accès sur la base du statut de session.
- Les requêtes non XHR sont automatiquement rejetées.
- Les variables GET sont injectées dans \$vars dans la vue.
- Si la requête contient \$_GET['guid'], le système définit \$vars['entity'] à l'entité correspondante ou false si celle-ci ne peut pas être chargée.
- Il n'y a pas aucun emballage (wrapper) placé autour de la sortie de la vue.
- Les messages/erreurs système ne doivent pas être utilisés, car ils ne s'affichent pas tant que l'utilisateur n'a pas chargé une autre page.
- Selon le suffixe de la vue (.js, .html, .css, etc.), l'entête Content-Type correspondant est ajouté.

Avertissement :

Dans les vues et les formulaires ajax, notez que \$vars peut être peuplé par les saisies du client. Les données sont filtrées `get_input()`, mais peut ne pas être le type que vous attendez ou peut avoir des clefs inattendues.

Renvoyer du JSON depuis une vue

Si les sorties des vues sont encodées en JSON, vous devez utiliser `elgg.getJSON` pour les récupérer (ou utiliser une autre méthode pour définir l'option ajax `dataType` de jQuery sur `json`). Votre fonction `success` recevra l'objet décodé.

Voici un exemple de récupération d'une vue qui renvoie un tableau d'heures encodé en JSON :

```

elgg.getJSON('ajax/view/myplugin/get_times', {
    success: function (data) {
        alert('The time is ' + data.friendly_time);
    }
});

```

Récupération de formulaire d'origine

Si vous enregistrez une vue de formulaire (nom commençant par `forms/`), vous pouvez la récupérer pré-générée avec `elgg_view_form()`. Il suffit d'utiliser `ajax/form/<action>` (au lieu de `ajax/view/<view_name>`) :

```
// in myplugin_init()
elgg_register_ajax_view('forms/myplugin/add');

elgg.get('ajax/form/myplugin/add', {
    success: function (output) {
        $('#myplugin-form-container').html(output);
    }
});
```

Seules les données de la requête sont transmises à la vue du formulaire demandé (c'est-à-dire en tant que troisième paramètre accepté par `elgg_view_form()`). Si vous devez passer des attributs ou des paramètres de l'élément de formulaire rendu par la vue `input/form` (c'est-à-dire normalement passé comme deuxième paramètre à `elgg_view_form()`), utilisez le hook serveur `view_vars`, `input/form`.

Avertissement :

Dans les vues et les formulaires ajax, notez que `$vars` peut être peuplé par les saisies du client. Les données sont filtrées comme `get_input()`, mais peut ne pas être le type que vous attendez ou peut avoir des clefs inattendues.

Fonctions d'aide d'origine

Ces fonctions étendent les fonctionnalités Ajax natives de jQuery.

`elgg.get()` est un wrapper pour `$.ajax()` de jQuery, mais force GET et normalise l'URL.

```
// normalizes the url to the current <site_url>/activity
elgg.get('/activity', {
    success: function(resultText, success, xhr) {
        console.log(resultText);
    }
});
```

`elgg.post()` est un wrapper pour `$.ajax()` de jQuery, mais force POST et normalise l'URL.

3.3.6 Authentification

Elgg fournit d'emblée tout ce qui est nécessaire pour authentifier les utilisateurs via le nom d'utilisateur/l'e-mail et le mot de passe, y compris :

- cookies pour une connexion persistante
- logique de réinitialisation de mot de passe
- stockage sécurisé des mots de passe
- déconnexion
- UIs pour accomplir tout ce qui précède

Il ne vous reste plus en tant que développeur qu'à utiliser les fonctions d'authentification intégrées pour sécuriser vos pages et vos actions.

Travailler avec l'utilisateur connecté

Vérifiez si l'utilisateur actuel est connecté avec `elgg_is_logged_in()` :

```
if (elgg_is_logged_in()) {  
    // do something just for logged-in users  
}
```

Vérifiez si l'utilisateur actuel est un administrateur avec `elgg_is_admin_logged_in()` :

```
if (elgg_is_admin_logged_in()) {  
    // do something just for admins  
}
```

Récupérez l'utilisateur actuellement connecté avec `elgg_get_logged_in_user_entity()` :

```
$user = elgg_get_logged_in_user_entity();
```

L'objet retourné est un `ElggUser` de sorte que vous pouvez utiliser toutes les méthodes et propriétés de cette classe pour accéder à des informations sur l'utilisateur. Si l'utilisateur n'est pas connecté, cela vous renvoie `null`, alors assurez-vous de vérifier cela en premier.

Gestionnaires d'accès

Les fonctions de gardien Gatekeeper vous permettent de gérer l'exécution du code en appliquant les règles de contrôle d'accès.

Redirige un utilisateur vers la page d'accueil s'il n'est pas connecté avec `elgg_gatekeeper()` :

```
elgg_gatekeeper();  
  
echo "Information for logged-in users only";
```

Redirige un utilisateur vers la page d'accueil à moins qu'il ne soit un administrateur avec `elgg_admin_gatekeeper()` :

```
elgg_admin_gatekeeper();  
  
echo "Information for admins only";
```

Modules d'authentification enfichables (Pluggable Authentication Modules)

Elgg est compatible avec des modules d'authentification enfichables (PAM = Pluggable Authentication Modules), qui vous permettent d'écrire vos propres gestionnaires d'authentification. A chaque fois qu'une requête a besoin d'être authentifiée le système va appeler `elgg_authenticate()` qui va vérifier tous les gestionnaires de PAM enregistrés jusqu'à ce que l'un d'entre eux renvoie un succès.

L'approche préférée est de créer un plugin Elgg séparé qui aura une tâche simple : traiter une demande d'authentification. Il s'agit de configurer un gestionnaire d'authentification dans le fichier *start.php* du plugin, et de l'enregistrer avec le module PAM afin qu'il soit traité chaque fois que le système doit authentifier une requête.

Le gestionnaire d'authentification est une fonction qui prend un seul paramètre. L'enregistrement du gestionnaire se fait par `register_pam_handler()` qui prend le nom du gestionnaire d'authentification, l'importance et la politique de validité comme paramètres. Il est conseillé d'enregistrer le gestionnaire dans la fonction `init` du plugin, par exemple :

```
function your_plugin_init() {
    // Register the authentication handler
    register_pam_handler('your_plugin_auth_handler');
}

function your_plugin_auth_handler($credentials) {
    // do things ...
}

// Add the plugin's init function to the system's init event
elgg_register_elgg_event_handler('init', 'system', 'your_plugin_init');
```

Importance

Par défaut, un module d'authentification est enregistré avec une importance de **sufficient** (suffisant).

Dans une liste de modules d'authentification; si n'importe lequel de ces modules marqué *sufficient* renvoie `true`, `pam_authenticate()` renverra également `true`. L'exception à cela est lorsqu'un module d'authentification est enregistré avec une importance de **required**. Tous les modules requis doivent renvoyer `true` pour que `pam_authenticate()` renvoie `true`, indépendamment du fait que tous les modules suffisants retournent `true`.

Informations d'identification transmises

Le format des informations d'identification transmises au gestionnaire peut varier en fonction de la demande d'origine. Par exemple, une connexion classique via le formulaire de connexion créera un tableau indexé, avec les clefs `username` et `password`. Si une demande a été faite par exemple via XML-RPC, les informations d'identification seront définies dans l'entête HTTP, de sorte que dans ce cas rien ne sera transmis au gestionnaire d'authentification, et que le gestionnaire devra effectuer ses propres étapes pour authentifier la demande.

Valeur de retour

Le gestionnaire d'authentification doit renvoyer un booléen, indiquant si la demande peut être authentifiée ou non. Une mise en garde est que dans le cas d'une connexion utilisateur classique où les informations d'identification sont disponibles (nom d'utilisateur et mot de passe), l'utilisateur sera connecté. Dans le cas de l'exemple XML-RPC, le gestionnaire d'authentification devra effectuer cette étape lui-même puisque le reste du système n'aura aucune idée des formats possibles des informations d'authentification passées ni de leur contenu. La connexion d'un utilisateur est assez simple et se fait par `login()`, qui s'attend à recevoir un objet `ElggUser`.

3.3.7 Contexte

Avertissement : Le contenu de cette page est obsolète. Bien que la fonctionnalité soit toujours en place, l'utilisation du contexte global pour déterminer votre logique de fonctionnement est une mauvaise pratique, et rendra votre code moins testable et plus sujet aux bogues.

Dans le framework Elgg, le contexte peut être utilisé par les fonctions de votre plugin pour déterminer s'ils doivent s'exécuter ou non. Vous enregistrerez les fonctions de rappel à exécuter lorsque des *événements sont déclenchés*. Parfois, les événements sont génériques et vous souhaitez uniquement exécuter votre fonction de rappel quand votre plugin est à l'origine du déclenchement de l'événement. Dans ce cas, vous pouvez utiliser le contexte de la page.

Vous pouvez définir explicitement le contexte avec `set_context()`. Le contexte est une chaîne et généralement vous la définissez sur le nom de votre plugin. Vous pouvez récupérer le contexte avec la fonction `get_context()`. Il est toutefois préférable d'utiliser `elgg_push_context($string)` pour ajouter un contexte à la pile. Vous pouvez vérifier si le contexte que vous souhaitez ajouter est déjà dans la pile actuelle en appelant `elgg_in_context($context)`. N'oubliez pas de dépiler (avec `elgg_pop_context()`) le contexte après l'avoir empilé, si vous n'en avez plus besoin.

Si vous ne le définissez pas, Elgg essaie de deviner le contexte. Si la page a été appelée via le routeur, le contexte est défini sur le premier segment de la route actuelle, par exemple `profile` dans `profile/username`.

Parfois, une vue va renvoyer un code HTML différent selon le contexte. Un plugin peut en profiter en définissant le contexte avant d'appeler `elgg_view()` sur la vue, puis en rétablissant le contexte d'origine. Cela se fait fréquemment avec le contexte de recherche (`search`).

3.3.8 Cron

Si vous configurez cron correctement comme décrit dans [Cron](#) des hooks spéciaux seront déclenchés afin que vous puissiez enregistrer des gestionnaires pour ces hooks à partir de votre propre code.

L'exemple ci-dessous enregistre une fonction pour le cron quotidien.

```
function my_plugin_init() {
    elgg_register_plugin_hook_handler('cron', 'daily', 'my_plugin_cron_handler');
}
```

Si le timing est important dans votre hook cron, notez que les fonctions sont exécutées dans l'ordre d'enregistrement. Cela pourrait signifier que votre fonction peut commencer (beaucoup) plus tard que vous l'avez peut-être prévu. Toutefois, les paramètres fournis dans le hook contiennent l'heure de départ d'origine du cron, de sorte que vous pouvez toujours utiliser ces informations.

```
function my_plugin_cron_handler(\Elgg\Hook $hook) {
    $start_time = $hook->getParam('time');
}
```

Intervalles personnalisés

Les développeurs de plugins peuvent configurer leurs propres intervalles personnalisés.

Avertissement : Il n'est **PAS** recommandé de faire cela, car les utilisateurs de votre plugin peuvent également avoir besoin de configurer votre intervalle personnalisé. Essayez de travailler avec les intervalles par défaut. Si vous n'avez qu'à effectuer une certaine tâche à 16 :30, vous pouvez utiliser l'intervalle `halfhour` et vérifier que `date('G', $start_time) == 16` et `date('i', $start_time) == 30`

```
elgg_register_plugin_hook_handler('cron:intervals', 'system', 'my_custom_cron_interval');

function my_custom_cron_interval(\Elgg\Hook $hook) {
    $cron_intervals = $hook->getValue();

    // add custom interval
    $cron_intervals['my_custom_interval'] = '30 16 * * *'; // every day at 16:30
    ↪ hours
```

(suite sur la page suivante)

(suite de la page précédente)

```

    return $cron_intervals;
}

```

Voir aussi :

- *Événements et Hooks des plugins* a plus d'informations sur les hooks
- Pour plus d'informations sur la définition des intervalles de cron pris en charge, voir la [documentation PHP Scheduler](#)

3.3.9 Base de données

Persistez le contenu et les paramètres générés par l'utilisateur avec l'API de stockage générique d'Elgg.

Contenu

- *Entités*
 - *Créer un objet*
 - *Charger un objet*
 - *Afficher des entités*
 - *Ajouter, lire et supprimer des annotations*
 - *Étendre elggEntity*
 - *Fonctionnalités avancées*
- *Fonctionnalité de base de données personnalisée*
 - *Exemple : exécuter un script SQL lors de l'activation du plugin*
- *Systemlog*
 - *Conservation du journal système*
 - *Créer votre propre journal système*

Entités

Créer un objet

Pour créer un objet dans votre code, vous devez instancier un `ElggObject`. La définition des données est simplement une question d'ajout de variables ou de propriétés d'instance. Les propriétés intégrées sont les suivantes :

- ```guid``` Le GUID de l'entité ; défini automatiquement
- ```owner_guid``` Le GUID de l'utilisateur propriétaire
- ```subtype``` Une chaîne arbitraire en un seul mot qui définit de quel type d'objet il s'agit, par exemple *blog*
- ```access_id``` Un entier qui représente le niveau d'accès de l'objet
- ```title``` Le titre de l'objet
- ```description``` La description de l'objet

Le sous-type d'objet est une propriété spéciale. Il s'agit d'une chaîne arbitraire qui décrit ce qu'est l'objet. Par exemple, si vous écriviez un plugin de blog, votre chaîne de sous-type peut être *blog*. C'est une bonne idée de faire en sorte que cette chaîne soit unique, de sorte que d'autres plugins n'essaient pas accidentellement d'utiliser le même sous-type. Pour les besoins de ce document, supposons que nous construisons un forum simple. Par conséquent, le sous-type sera *forum* :

```

$object = new elggObject();
$object->subtype = "forum";
$object->access_id = 2;
$object->save();

```

`access_id` est une autre propriété importante. Si vous ne la définissez pas, votre objet sera privé, et seul l'utilisateur qui l'a créé sera en mesure de le voir. Elgg définit des constantes pour les valeurs spéciales de `access_id` :

- **ACCESS_PRIVATE** Seul son propriétaire peut le voir
- **ACCESS_LOGGED_IN** Tout utilisateur connecté peut le voir
- **ACCESS_PUBLIC** Même les visiteurs non connectés peuvent le voir

L'enregistrement de l'objet remplira automatiquement la propriété `$object->guid` en cas de succès. Si vous modifiez d'autres propriétés de base, vous pouvez appeler à nouveau `$object->save()`, ce qui va mettre à jour la base de données pour vous.

Vous pouvez définir des métadonnées sur un objet comme une propriété standard. Disons que nous voulons définir le SKU (référence unique pour les stocks) d'un produit :

```
$object->SKU = 62784;
```

Si vous affectez un tableau, toutes les valeurs seront définies pour ces métadonnées. C'est ainsi que, par exemple, vous définissez des balises.

Les métadonnées ne peuvent pas être conservées dans la base de données tant que l'entité n'a pas été enregistrée, mais pour plus de commodité, ElggEntity peut la mettre en cache en interne et l'enregistrer lors de l'enregistrement de l'entité.

Charger un objet

Par GUID

```
$entity = get_entity($guid);
if (!$entity) {
    // The entity does not exist or you're not allowed to access it.
}
```

Mais que faire si vous ne connaissez pas le GUID ? Il y a plusieurs options.

Par utilisateur, sous-type ou site

Si vous connaissez l'ID de l'utilisateur pour lequel vous souhaitez obtenir des objets ou le sous-type, vous disposez de plusieurs options. Le plus simple est probablement d'appeler la fonction procédurale `elgg_get_entities` :

```
$entities = elgg_get_entities(array(
    'type' => $entity_type,
    'subtype' => $subtype,
    'owner_guid' => $owner_guid,
));
```

Ceci renvoie un tableau d'objets ElggEntity que vous pouvez itérer. `elgg_get_entities` a une pagination par défaut, avec une limite de 10, et un décalage (offset) de 0.

Vous pouvez laisser de côté `owner_guid` pour obtenir tous les objets et laisser de côté le sous-type ou le type pour obtenir des objets de tous types/sous-types.

Si vous avez déjà un ElggUser – par exemple `elgg_get_logged_in_user_entity`, qui renvoie toujours l'objet de l'utilisateur actuel lorsque vous êtes connecté – vous pouvez simplement utiliser :

```
$objects = $user->getObjects($subtype, $limit, $offset)
```

Mais qu'en est-il d'obtenir des objets ayant une valeur de métadonnée particulière ?

Par propriétés

Vous pouvez récupérer des entités par leurs propriétés à l'aide de `elgg_get_entities`. En utilisant des paramètres spécifiques transmis au tableau `$options`, vous pouvez récupérer des entités par leurs attributs, métadonnées, annotations, paramètres privés et relations.

Afficher des entités

Pour que les entités s'affichent dans les fonctions de liste, vous devez fournir une vue pour l'entité dans le système de vues.

Pour afficher une entité, créez une vue `EntityType/subtype` où `EntityType` a l'une des valeurs suivantes :

`object` : pour les entités dérivées de `ElggObject`, `user` : pour les entités dérivées de `ElggUser`, `site` : pour les entités dérivées de `ElggSite`, `group` : pour les entités dérivées de `ElggGroup`

Une vue par défaut pour toutes les entités a déjà été créée, elle s'appelle `EntityType/default`.

Icônes des entités

Les icônes d'entité peuvent être enregistrées à partir de fichiers téléchargés, de fichiers locaux existants, ou d'objets `ElggFile` existants. Ces méthodes enregistrent la taille `master` de l'icône définie dans le système. Les autres tailles définies seront générées sur demande.

```
$object = new ElggObject();
$object->title = 'Example entity';
$object->description = 'An example object with an icon.';

// from an uploaded file
$object->saveIconFromUploadedFile('file_upload_input');

// from a local file
$object->saveIconFromLocalFile('/var/data/generic_icon.png');

// from a saved ElggFile object
$file = get_entity(123);
if ($file instanceof ElggFile) {
    $object->saveIconFromElggFile($file);
}

$object->save();
```

Les tailles d'images suivantes existent par défaut :

- `master` - 10240px dans la plus grande longueur (sans agrandissement)
- `large` - 200px dans la plus grande dimension (sans agrandissement)
- `medium` - carré de 100px de côté
- `small` - carré de 40px de côté
- `tiny` - carré de 25px de côté
- `topbar` - carré de 16px de côté

Utilisez `elgg_get_icon_sizes()` pour obtenir toutes les tailles d'icônes possibles pour un type d'entité et un sous-type spécifiques. La fonction déclenche le *hook* `entity:icon:sizes`.

Pour vérifier si une icône est définie, utilisez `$object->hasIcon($size)`.

Vous pouvez récupérer l'URL de l'icône générée avec la méthode `ElggEntity::getIconURL($params)`. Cette méthode accepte comme argument `$params` un tableau qui spécifie la taille, le type et fournit un contexte supplémentaire pour que le hook détermine l'icône à servir. La méthode déclenche le *hook* `entity:icon:url`.

Utilisez `elgg_view_entity_icon($entity, $size, $vars)` pour afficher une icône. Cela analysera les emplacements suivants pour une vue et inclura la première qui correspond.

1. `views/$viewtype/icon/$type/$subtype.php`
2. `views/$viewtype/icon/$type/default.php`
3. `views/$viewtype/icon/default.php`

Où

\$viewtype Type de vue, par ex. `'default'` ou `'json'`.

\$type Type d'entité, par ex. `'group'` ou `'user'`.

\$subtype Sous-type d'entité, par ex. `'blog'` ou `'page'`.

Vous n'avez pas à renvoyer une icône de secours du gestionnaire de hook. Si aucune icône téléchargée n'est trouvée, le système d'affichage analysera les vues (dans cet ordre spécifique) :

1. `views/$viewtype/$icon_type/$entity_type/$entity_subtype.svg`
2. `views/$viewtype/$icon_type/$entity_type/$entity_subtype/$size.gif`
3. `views/$viewtype/$icon_type/$entity_type/$entity_subtype/$size.png`
4. `views/$viewtype/$icon_type/$entity_type/$entity_subtype/$size.jpg`

Où

\$viewtype Type de vue, par ex. `'default'` ou `'json'`.

\$icon_type Type d'icône, par ex. `'icon'` ou `'cover_image'`.

\$entity_type Type d'entité, par ex. `'group'` ou `'user'`.

\$entity_subtype Sous-type d'entité, par ex. `'blog'` ou `'page'` (ou `'default'` si l'entité n'a pas de sous-type).

\$size Taille de l'icône (notez que nous n'utilisons pas la taille avec des icônes svg)

Les méthodes des icônes permettent de passer un type d'icône si une entité possède plus d'une icône. Par exemple, un utilisateur peut avoir un avatar et une icône de photo de couverture. Vous passeriez `'cover_photo'` comme type d'icône :

```
$object->saveIconFromUploadedFile('uploaded_photo', 'cover_photo');

$object->getIconUrl([
    'size' => 'medium',
    'type' => 'cover_photo'
]);
```

Note : Les types d'icônes personnalisés (par ex. les photos de couverture) n'ont qu'un pré-réglage pour la taille *master* ; pour ajouter des tailles personnalisées, utilisez le *hook* `entity:<icon_type>:url` pour les configurer.

Par défaut, les icônes seront stockées dans `/icons/<icon_type>/<size>.jpg` par rapport au répertoire de l'entité dans le répertoire des données. Pour fournir un emplacement alternatif, utilisez le *hook* `entity::file`.

Ajouter, lire et supprimer des annotations

Les annotations peuvent être utilisées, par exemple, pour suivre les évolutions des notations. Pour annoter une entité, vous pouvez utiliser la méthode `annotate()` de l'objet. Par exemple, pour donner à un article de blog une note de 5, vous pouvez utiliser :

```
$blog_post->annotate('rating', 5);
```

Pour récupérer les notations sur l'article de blog, utilisez `$blogpost->getAnnotations('rating')` et si vous souhaitez supprimer une annotation, vous pouvez opérer sur la classe `ElggAnnotation`, par exemple `$annotation->delete()`.

La récupération d'une seule annotation peut se faire avec `get_annotation()` si vous avez l'ID de l'annotation. Si vous supprimez une `ElggEntity` de quelque nature que ce soit, toutes ses métadonnées, annotations et relations seront également supprimées automatiquement.

Étendre ElggEntity

Si vous dérivez de l'une des classes du noyau Elgg, vous devrez dire à Elgg comment instancier correctement le nouveau type d'objet afin que `get_entity()` et assimilés renvoie la classe PHP appropriée. Par exemple, si je personnalise `ElggGroup` dans une classe appelée `Committee`, je dois faire connaître à Elgg le nouveau mappage. Voici un exemple d'extension de classe :

```
// Class source
class Committee extends ElggGroup {

    protected function initializeAttributes() {
        parent::initializeAttributes();
        $this->attributes['subtype'] = 'committee';
    }

    // more customizations here
}
```

Dans votre fichier `elgg-plugin.php` ajoutez la section `entities`.

```
<?php // mod/example/elgg-plugin.php
return [
    // entities registration
    'entities' => [
        [
            'type' => 'group',
            'subtype' => 'committee',
            'class' => 'Committee',
            'searchable' => true,
        ],
    ],
];
```

Les entités seront enregistrées lors de l'activation du plugin.

Maintenant, si vous invoquez `get_entity()` avec le GUID d'un objet de comité, vous récupérerez un objet de type `Comité`.

Fonctionnalités avancées

URLs des entités

Les URLs d'entité sont fournies par l'interface `getURL()` et fournissent à l'infrastructure Elgg un moyen commun de diriger les utilisateurs vers le gestionnaire d'affichage approprié pour n'importe quel objet donné.

Par exemple, une page de profil dans le cas des utilisateurs.

L'URL est définie à l'aide de la fonction `elgg_register_entity_url_handler()`. La fonction que vous enregistrez doit renvoyer l'URL appropriée pour le type donné - qui peut lui-même être une adresse configurée par un gestionnaire de page.

Le gestionnaire par défaut doit utiliser l'interface d'export par défaut.

Performance de chargement des entités

`elgg_get_entities` dispose de quelques options qui peuvent parfois être utiles pour améliorer les performances.

- **preload_owners** : Si les entités récupérées sont affichées dans une liste avec des informations sur le propriétaire, vous pouvez définir cette option sur `true` pour charger efficacement les utilisateurs propriétaires des entités récupérées.
- **preload_containers** : Si les entités récupérées sont affichées dans une liste avec des informations sur leurs conteneurs, vous pouvez définir cette option sur `true` pour les charger efficacement.
- **distinct** : Quand Elgg récupère des entités à l'aide d'une requête SQL, Elgg doit être sûr que chaque ligne d'entité n'apparaît qu'une seule fois dans le jeu de résultats. Par défaut, il inclut un modificateur `DISTINCT` sur la colonne `GUID` pour l'appliquer, mais certaines requêtes renvoient naturellement des entités uniques. La définition de l'option `distinct` sur `false` supprime ce modificateur et s'appuie sur la requête pour s'assurer de l'unicité des résultats.

Le fonctionnement interne des requêtes d'entités Elgg est un sujet complexe et il est recommandé de demander de l'aide sur le site de la communauté Elgg avant d'utiliser l'option `distinct`.

Fonctionnalité de base de données personnalisée

Il est fortement recommandé d'utiliser des entités dans la mesure du possible. Toutefois, Elgg prend en charge les requêtes SQL personnalisées à l'aide de l'API de base de données.

Exemple : exécuter un script SQL lors de l'activation du plugin

Cet exemple montre comment remplir votre base de données lors de l'activation du plugin.

```
if (!elgg_get_plugin_setting('database_version', 'my_plugin')) {
    run_sql_script(__DIR__ . '/sql/activate.sql');
    elgg_set_plugin_setting('database_version', 1, 'my_plugin');
}
```

my_plugin/sql/activate.sql :

```
-- Create some table
CREATE TABLE prefix_custom_table(
    id INTEGER AUTO_INCREMENT,
    name VARCHAR(32),
    description VARCHAR(32),
```

(suite sur la page suivante)

(suite de la page précédente)

```

PRIMARY KEY (id)
);

-- Insert initial values for table
INSERT INTO prefix_custom_table (name, description)
VALUES ('Peter', 'Some guy'), ('Lisa', 'Some girl');
```

Notez que Elgg exécute des instructions par le biais des fonctions intégrées de PHP et qu'il dispose d'une prise en charge limitée des commentaires. C'est-à-dire que seuls les commentaires d'une seule ligne sont pris en charge et qu'ils doivent être préfixés par « `--` » ou « `#` ». Un commentaire doit commencer au tout début d'une ligne.

Systemlog

Note : Cette section requiert une certaine attention et risque de contenir des informations obsolètes

Le journal système de Elgg par défaut est un moyen simple d'enregistrer ce qui se passe dans un système Elgg. Il est visible et consultable directement à partir du panneau d'administration.

Conservation du journal système

Une ligne de journal système est stockée chaque fois qu'un événement concernant un objet dont la classe implémente l'interface `design/loggable` est déclenché. `ElggEntity` et `ElggExtender` implémentent *Loggable*, de sorte qu'une ligne de journal système est créée chaque fois qu'un événement est effectué sur tous les objets, utilisateurs, groupes, sites, métadonnées et annotations.

Les événements courants comprennent :

- create
- update
- delete
- login

Créer votre propre journal système

Il y a quelques raisons pour lesquelles vous pouvez vouloir créer votre propre journal système. Par exemple, vous pourriez avoir besoin de stocker une copie complète des entités lorsqu'elles sont mises à jour ou supprimées, à des fins d'audit. Vous pourriez aussi avoir besoin d'aviser un administrateur lorsque certains types d'événements se produisent.

Pour ce faire, vous pouvez créer une fonction qui écoute tous les événements pour tous les types d'objets :

```
register_elgg_event_handler('all', 'all', 'your_function_name');
```

Votre fonction peut alors être définie comme :

```
function your_function_name($object, $event) {
    if ($object instanceof Loggable) {
        ...
    }
}
```

Vous pouvez ensuite utiliser les méthodes supplémentaires définies par *Loggable* pour extraire les informations dont vous avez besoin.

3.3.10 Gérer les erreurs

Sous le capot, Elgg utilise *Monolog* pour enregistrer les erreurs dans le journal d'erreur du serveur (et le stdout pour les commandes CLI).

Monolog est livré avec un certain nombre d'outils qui peuvent aider les administrateurs à suivre les erreurs et le débogage des informations.

Vous pouvez ajouter des gestionnaires personnalisés (voir la documentation de *Monolog* pour une liste complète des gestionnaires) :

```
// Add a new handler to notify a given email about a critical error
elgg()->logger->pushHandler(
    new \Monolog\Handler\NativeMailerHandler(
        'admin@example.com',
        'Critical error',
        'no-reply@mysite.com',
        \Monolog\Logger::CRITICAL
    )
);
```

3.3.11 Liste des événements dans le cœur

Pour plus d'informations sur le fonctionnement des événements, visitez *Événements et Hooks des plugins*.

Contenu

- Événements système
- Événements utilisateur
- Événements des relations
- Événements des entités
- Événements des métadonnées
- Événements des annotations
- Événements de la rivière
- Événements des fichiers
- Notes

Note : Certains événements sont marqués par (*séquence*) cela signifie que ces événements ont également un événement `:before` et `:after`. Voyez également *Séquence d'événement*

Événements système

plugins_load, system (séquence) Déclenché avant que les plugins ne soient chargés. Rarement utilisé. `init`, `system` est utilisé à la place. Peut être utilisé pour charger des bibliothèques supplémentaires.

plugins_boot, system (séquence) Déclenché juste après le chargement des plugins. Rarement utilisé. `init`, `system` est utilisé à la place.

init, system (séquence) Les plugins ont tendance à utiliser cet événement pour l'initialisation (extension des vues, enregistrement des rappels, etc.)

ready, system (séquence) Déclenché après l'événement `init`, `system`. Tous les plugins sont entièrement chargés et le moteur est prêt à servir les pages.

shutdown, system Déclenché après l'envoi de la page à l'utilisateur. Des opérations coûteuses pourraient être faites ici sans que cela fasse attendre l'utilisateur.

Note : Selon la configuration de votre serveur, la sortie PHP peut ne pas être affichée avant la fin du processus. Cela signifie que tous les processus longs retarderont toujours le chargement de la page.

Note : Cet événement est préféré à l'utilisation de `register_shutdown_function` car vous n'avez peut-être pas accès à tous les services Elgg (par ex. la base de données) dans la fonction d'arrêt, alors que vous l'aurez dans l'événement.

Note : La session Elgg est déjà terminée avant cet événement. La manipulation de la session n'est plus possible.

regenerate_site_secret :before, system Renvoyez `false` pour annuler la régénération du secret du site. Vous devez également fournir un message à l'utilisateur.

regenerate_site_secret :after, system Déclenché après que le secret du site a été régénéré.

log, systemlog Appelé pour tous les événements déclenchés par le plugin `system_log`. Utilisé en interne par `system_log_default_logger()` pour remplir la table `system_log`.

upgrade, system Déclenché une fois la mise à niveau du système terminée. Tous les scripts de mise à niveau sont exécutés, mais les caches ne sont pas effacés.

upgrade :execute, system (séquence) Déclenché lors de l'exécution d'un `ElggUpgrade`. Le `$object` de l'événement est le `ElggUpgrade`.

activate, plugin Renvoyez `false` pour empêcher l'activation du plugin.

deactivate, plugin Renvoyez `false` pour éviter la désactivation du plugin.

init :cookie, <name> Renvoyez `false` pour remplacer la définition d'un cookie.

cache :invalidate, system (séquence) Invalide les caches internes et externes.

cache :clear, system (séquence) Effacer les caches internes et externes, par défaut, y compris `system_cache`, `simplecache` et `memcache`. Il est possible de l'utiliser pour réinitialiser d'autres caches comme `APC`, `OPCache` ou `WinCache`.

cache :purge, system (séquence) Purge les caches internes et externes. Il s'agit de supprimer le contenu ancien/obsolète des caches.

send :before, http_response Déclenché avant l'envoi d'une réponse HTTP. Les gestionnaires recevront une instance de `\Symfony\Component\HttpFoundation\Response` qui doit être envoyée au demandeur. Les gestionnaires peuvent mettre fin à l'événement et empêcher l'envoi de la réponse en renvoyant `false`.

send :after, http_response Déclenché après l'envoi d'une réponse HTTP. Les gestionnaires recevront une instance de la réponse `\Symfony\Component\HttpFoundation\Response` qui a été envoyée au demandeur.

reload :after, translations Déclenché après que les traductions ont été (re)chargées.

Événements utilisateur

login :before, user Déclenché lors de la connexion. Renvoyer false empêche l'utilisateur de se connecter

login :after, user Déclenché après que l'utilisateur se soit identifié.

logout :before, user Déclenché pendant la déconnexion. Renvoyer false empêche l'utilisateur de se déconnecter.

logout :after, user Déclenché après que l'utilisateur se soit déconnecté.

validate, user Lorsqu'un utilisateur s'inscrit, le compte de l'utilisateur est désactivé. Cet événement est déclenché pour permettre à un plugin de déterminer comment l'utilisateur doit être validé (par exemple, par le biais d'un e-mail avec un lien de validation).

validate :after, user Déclenché lorsque le compte utilisateur a été validé.

invalidate :after, user Déclenché lorsque la validation du compte de l'utilisateur a été révoquée.

profileupdate, user L'utilisateur a modifié son profil

profileiconupdate, user L'utilisateur a modifié son icône de profil

ban, user Déclenché avant qu'un utilisateur ne soit banni. Retournez false pour l'éviter.

unban, user Déclenché avant qu'un utilisateur ne soit réintégré. Retournez false pour l'éviter.

make_admin, user Déclenché avant qu'un utilisateur ne soit promu administrateur. Retournez false pour l'éviter.

remove_admin, user Déclenché avant que le rôle d'administrateur ne soit retiré à un utilisateur. Retournez false pour l'éviter.

Événements des relations

create, relationship Déclenché après la création d'une relation. Renvoyez false pour supprimer la relation qui vient d'être créée.

delete, relationship Déclenché avant la suppression d'une relation. Renvoyez false pour éviter qu'elle soit supprimée.

join, group Déclenché après que l'utilisateur `$params['user']` a rejoint le groupe `$params['group']`.

leave, group Déclenché avant que l'utilisateur `$params['user']` n'ait quitté le groupe `$params['group']`.

Événements des entités

create, <entity type> Déclenché pour les entités user, group, object et site après la création. Renvoyez false pour supprimer l'entité.

update, <entity type> Déclenché avant une mise à jour pour les entités user, group, object et site. Renvoyez false pour empêcher la mise à jour. La méthode d'entité `getOriginalAttributes()` peut être utilisée pour identifier les attributs qui ont changé depuis la dernière fois que l'entité a été enregistrée.

update :after, <entity type> Déclenché après une mise à jour pour les entités user, group, object et site. La méthode d'entité `getOriginalAttributes()` peut être utilisée pour identifier les attributs qui ont changé depuis la dernière fois que l'entité a été enregistrée.

delete, <entity type> Déclenché avant la suppression d'une entité. Retournez false pour éviter la suppression.

disable, <entity type> Déclenché avant qu'une entité soit désactivée. Retournez false pour éviter la désactivation.

disable :after, <entity type> Déclenché après qu'une entité a été désactivée.

enable, <entity type> Retournez false pour éviter l'activation.

enable :after, <entity type> Déclenché après qu'une entité a été activée.

Événements des métadonnées

create, metadata Appelé après qu'une métadonnée a été créée. Retournez false pour supprimer la matadonnée qui vient d'être créée.

update, metadata Appelé après qu'une métadonnée a été modifiée. Retournez false pour *supprimer la métadonnée*.

delete, metadata Appelé avant que métadonnée soit supprimée. Retournez false pour éviter la suppression.

enable, metadata Appelé lors de l'activation de la métadonnée. Retournez false pour éviter l'activation.

disable, metadata Appelé lors de la désactivation de la métadonnée. Retournez false pour éviter la désactivation.

Événements des annotations

annotate, <entity type> Appelé avant que l'annotation soit créée. Retournez false pour éviter l'annotation de cette entité.

create, annotation Appelé après que l'annotation ait été créée. Retournez false pour supprimer l'annotation.

update, annotation Appelé après qu'une annotation a été modifiée. Retournez false pour *supprimer l'annotation*.

delete, annotation Appelé avant que l'annotation soit supprimée. Retournez false pour éviter la suppression.

enable, annotation Appelé lors de l'activation d'annotations. Retournez false pour éviter l'activation.

disable, annotations Appelé lors de la désactivation d'annotations. Retournez false pour éviter la désactivation.

Événements de la rivière

created, river Appelé après qu'un élément de la rivière a été créé.

Note : Utilisez le hook plugin `creating, river` pour annuler la création (ou modifier les options).

delete :before, river Déclenché avant qu'un élément de la rivière soit supprimé. Renvoyer false annule la suppression.

delete :after, river Déclenché après qu'un élément de la rivière a été supprimé.

Événements des fichiers

upload :after, file Appelé après qu'un fichier téléchargé a été écrit dans le répertoire de fichiers. Reçoit une instance du `ElggFile` dans lequel le fichier téléchargé a été écrit. Le `ElggFile` peut ou non être une entité avec un GUID.

Notes

En raison des bogues dans le noyau de Elgg, certains événements peuvent être lancés plus d'une fois sur la même action. Par exemple, `update, object` est lancé deux fois.

3.3.12 Système de fichier

Contenu

- *Répertoire de fichiers*
- *Objets Fichiers (File)*
- *Fichiers temporaires*

Répertoire de fichiers

Emplacement

Le répertoire de fichiers de Elgg est situé dans le dataroot du site, configuré pendant l'installation, et qui peut être modifié via les paramètres du site dans l'interface Admin.

Structure du répertoire

La structure du répertoire de fichiers est liée à la propriété des fichiers par les entités Elgg. Chaque fois que le premier fichier appartenant à une entité est écrit dans le répertoire de fichiers, un répertoire correspondant au GUID de l'entité sera créé dans le répertoire bucket parent (les buckets sont de 5000 guids). Par exemple, les fichiers appartenant à l'utilisateur avec le guid 7777 seront situés dans 5000/7777/.

Lorsque des fichiers sont créés, les noms de fichiers peuvent contenir des noms de sous-répertoires (souvent appelés \$prefix dans le code). Par exemple, les avatars de l'utilisateur ci-dessus, peuvent être trouvés dans 5000/7777/profile/.

Objets Fichiers (File)

Écrire des fichiers

Pour écrire un fichier dans le répertoire de fichiers, vous utiliseriez une instance de ElggFile. Même si ElggFile étend ElggObject et peut être stocké comme une entité Elgg réelle, ce n'est pas toujours nécessaire (par exemple pour la création des miniatures d'une image).

```
$file = new ElggFile();
$file->owner_guid = 7777;
$file->setFilename('portfolio/files/sample.txt');
$file->open('write');
$file->write('Contents of the file');
$file->close();

// to upgrade this file to an entity
$file->subtype = 'file';
$file->save();
```

Lire des fichiers

Vous pouvez lire le contenu du fichier en utilisant une instance de `ElggFile`.

```
// from an Elgg entity
$file = get_entity($file_guid);
readfile($file->getFilenameOnFilestore());

// arbitrary file on the filestore
$file = new ElggFile();
$file->owner_guid = 7777;
$file->setFilename('portfolio/files/sample.txt');

// option 1
$file->open('read');
$content = $file->grabFile();
$file->close();

// option 2
$content = file_get_contents($file->getFilenameOnFilestore());
```

Servir des fichiers

Vous pouvez servir des fichiers depuis le répertoire de fichiers en utilisant `elgg_get_inline_url()` et `elgg_get_download_url()`. Les deux fonctions acceptent 3 arguments :

- `file` Une instance de `ElggFile` à servir
- `use_cookie` Si défini à vrai, la validité de l'URL sera limitée à la session actuelle
- `expires` Heure d'expiration de l'URL

Vous pouvez utiliser les arguments `use_cookie` et `expires` comme moyen de contrôle d'accès. Par exemple, dans la plupart des cas, les avatars des utilisateurs ont un long délai d'expiration et n'ont pas besoin d'être limités à la session en cours - ce qui permettra aux navigateurs de mettre en cache les images. Le service de fichiers enverra les en-têtes `Not Modified` appropriés aux requêtes suivantes.

Le comportement par défaut de `use_cookie` peut être contrôlé sur la page des paramètres de sécurité de l'administrateur.

Pour les entités qui utilisent le contrôle d'accès de Elgg, vous pouvez utiliser des cookies pour vous assurer que les paramètres d'accès sont respectés et que les utilisateurs ne partagent pas les URLs de téléchargement avec d'autres personnes.

Vous pouvez également invalider toutes les URLs précédemment générées en mettant à jour l'heure de modification du fichier, par exemple à l'aide de `touch()`.

Embarquer des fichiers

Veuillez noter qu'en raison de leur nature, les URLs en ligne (inline) et de téléchargement ne sont pas adaptées à l'intégration (embed). Les URLs d'intégration doivent être permanentes, tandis que les URL en ligne et de téléchargement sont volatiles (liées à la session utilisateur et à l'heure de modification des fichiers).

Pour intégrer une icône d'entité, utilisez `elgg_get_embed_url()`.

Gérer l'envoi de fichiers

Pour implémenter une action qui enregistre un unique fichier envoyé par un utilisateur, vous pouvez utiliser l'approche suivante :

```
// in your form
echo elgg_view('input/file', [
    'name' => 'upload',
    'label' => 'Select an image to upload',
    'help' => 'Only jpeg, gif and png images are supported',
]);
```

```
// in your action
$uploaded_file = elgg_get_uploaded_file('upload');
if (!$uploaded_file) {
    return elgg_error_response("No file was uploaded");
}

$supported_mimes = [
    'image/jpeg',
    'image/png',
    'image/gif',
];

$mime_type = elgg()->mimetype->getMimeType($uploaded_file->getPathname());
if (!in_array($mime_type, $supported_mimes)) {
    return elgg_error_response("{ $mime_type } is not supported");
}

$file = new ElggFile();
$file->owner_guid = elgg_get_logged_in_user_guid();
if ($file->acceptUploadedFile($uploaded_file)) {
    $file->save();
}
```

Si votre champ de saisie de fichiers supporte plusieurs fichiers, vous pouvez itérer parmi eux dans votre action :

```
// in your form
echo elgg_view('input/file', [
    'name' => 'upload[]',
    'multiple' => true,
    'label' => 'Select images to upload',
]);
```

```
// in your action
foreach (elgg_get_uploaded_files('upload') as $upload) {
    $file = new ElggFile();
    $file->owner_guid = elgg_get_logged_in_user_guid();
    if ($file->acceptUploadedFile($upload)) {
        $file->save();
    }
}
```

Note : Si des images sont chargées une tentative automatique est faite pour corriger l'orientation de l'image.

Fichiers temporaires

Si jamais vous avez besoin d'un fichier temporaire vous pouvez utiliser `elgg_get_temp_file()`. Vous obtiendrez une instance de `ElggTempFile` qui a toutes les fonctions de `ElggFile`, mais écrit ses données dans le répertoire temporaire du système.

Avertissement : Il n'est pas possible d'enregistrer `ElggTempFile` dans la base de données. Vous obtiendriez une `IOException` si vous essayiez.

3.3.13 Outils du groupe

Les groupes Elgg permettent aux administrateurs de groupe d'activer/désactiver les différents outils disponibles au sein d'un groupe. Ces outils sont fournis par d'autres plugins comme blog ou file.

Les plugins peuvent accéder au registre des outils du groupe via `elgg()->group_tools`.

```
elgg()->group_tools->register('my-tool', [
    'default_on' => false, // default is true
    'label' => elgg_echo('my-tool:checkbox:label'),
    'priority' => 300, // display this earlier than other modules/tools
]);
```

Un outil enregistré aura la possibilité d'être (dés)activé sur le formulaire de modification d'un groupe et peut avoir un module d'affichage de profil qui lui est associé. Pour ajouter un module de profil, il suffit d'ajouter une vue correspondante en tant que `groups/profile/module/<tool_name>`. Cette vue ne sera appelée que si l'outil est activé.

Si vous souhaitez simplement lister certains contenus dans le groupe, vous pouvez utiliser la vue `groups/profile/module` avec quelques paramètres supplémentaires.

- `entity_type` : en combinaison avec le `entity_subtype`, il peut générer tout ce dont le module a besoin
- `entity_subtype` : en combinaison avec le `entity_type`, il peut générer tout ce dont le module a besoin
- `no_results` : texte personnalisé si aucun résultat n'est trouvé

Les éléments suivants seront générés automatiquement :

- `title` : basé sur la clef de traduction `collection:<entity_type>:<entity_subtype>:group`
- `content` : `elgg_list_entities()` basé sur un type/sous-type donné
- `all_link` : basé sur le nom de route `collection:<entity_type>:<entity_subtype>:group`
- `add_link` : basé sur le nom de route `add:<entity_type>:<entity_subtype>:group` et avec une vérification des autorisations pour un type/sous-type donné

```
// file: groups/profile/module/my-tool.php

// if you wish to list some content (eg. files) in the group
// you can use the following
$params = [
    'entity_type' => 'object',
    'entity_subtype' => 'file',
    'no_results' => elgg_echo('file:none'),
];
```

(suite sur la page suivante)

(suite de la page précédente)

```
$params = $params + $vars;  
  
echo elgg_view('groups/profile/module', $params);
```

Vous pouvez également générer votre propre titre et contenu

```
// file: groups/profile/module/my-tool.php  
  
echo elgg_view('groups/profile/module', [  
    'title' => elgg_echo('my-tool'),  
    'content' => 'Hello, world!',  
]);
```

Vous pouvez activer et désactiver des outils de manière programmatique pour un groupe donné :

```
$group = get_entity($group_guid);  
  
// enables the file tool for the group  
$group->enableTool('file');  
  
// disables the file tool for the group  
$group->disableTool('file');
```

Si vous voulez autoriser une certaine fonctionnalité dans un groupe seulement si l'option du groupe pour cet outil est activée, vous pouvez vérifier en utilisant `\ElggGroup::isToolEnabled($tool_option)`.

Il est également possible d'utiliser une fonction gardienne (gatekeeper) pour éviter l'accès à une page d'un groupe sur la base de l'activation d'un outil.

```
elgg_group_tool_gatekeeper('file', $group);
```

Voir aussi :

Lisez-en plus sur les gardiens (gatekeepers) ici : *Gestionnaires d'accès*

Si vous avez besoin des options des outils du groupe configurés vous pouvez utiliser la fonction `use the elgg()->group_tools->group($group)`.

3.3.14 Recommandations pour le développement de plugins

En plus des Standards de Développement Elgg (Elgg Coding Standards), voici les recommandations pour créer des plugins. Les plugins du noyau sont mis à jour vers ce format et tous les auteurs de plugins devraient suivre ces recommandations pour leurs propres plugins.

Voir aussi :

Assurez-vous de suivre *Squelette du plugin* pour la structure de votre plugin.

Avertissement : *Ne modifiez pas le cœur*

Contenu

- Utilisez le routage standard avec vos gestionnaires de pages
- Utilisez les gestionnaires de pages standardisés et les scripts
- La vue `object/<subtype>`
- Actions
- Appeler directement un fichier
- Recommandé

Utilisez le routage standard avec vos gestionnaires de pages

- Exemple : plugin Bookmarks (Signets)
- Les gestionnaires de pages devraient accepter le standard d'URLs suivant :

Objectif	URL
Tout	<code>page_handler/all</code>
Utilisateur	<code>page_handler/owner/<username></code>
Contacts de l'utilisateur	<code>page_handler/friends/<username></code>
Entité unique	<code>page_handler/view/<guid>/<title></code>
Ajouter	<code>page_handler/add/<container_guid></code>
Modifier	<code>page_handler/edit/<guid></code>
Liste du groupe	<code>page_handler/group/<guid>/owner</code>

- Incluez les scripts du gestionnaire de page à partir du gestionnaire de page. Presque tous les gestionnaires de page doivent avoir un script de gestionnaire de page. (Par exemple : `bookmarks/all => mod/bookmarks/views/default/resources/bookmarks/all.php`)
- Passez des arguments tels que des guids d'entités à la vue de la ressource via `$vars` dans `elgg_view_resource()`.
- Appelez `elgg_gatekeeper()` et `elgg_admin_gatekeeper()` dans la fonction du gestionnaire de page si nécessaire.
- L'URL de groupe doit utiliser des vues telles que `resources/groups/*.php` pour afficher des pages.
- Les gestionnaires de pages ne devraient pas contenir de HTML.

Utilisez les gestionnaires de pages standardisés et les scripts

- Exemple : plugin Bookmarks (Signets)
- Stockez les fonctionnalités des pages dans `mod/<plugin>/views/default/resources/<page_handler>/<page_name>.php`
- Utilisez `elgg_view_resource('<page_handler>/<page_name>')` pour les afficher.
- Utilisez la disposition de page par défaut dans les scripts du gestionnaire de pages : `$content = elgg_view_layout('default', $options);`
- Les scripts de gestionnaires de pages ne devraient pas contenir de HTML
- Appelez `elgg_push_breadcrumb()` dans les scripts du gestionnaire de pages.
- Nul besoin de s'inquiéter de définir le propriétaire de la page si les URLs sont dans le format standardisé
- Pour le contenu du groupe, vérifiez le conteneur `container_guid` en utilisant `elgg_get_page_owner_entity()`

La vue object/<subtype>

- Exemple : plugin Bookmarks (Signets)
- Assurez-vous qu'il existe bien des vues pour `$vars['full_view'] == true` et `$vars['full_view'] == false`
- Vérifiez l'objet dans `$vars['entity']`. Utilisez `elgg_instance_of()` pour vous assurer qu'il s'agit du type d'entité que vous souhaitez. Renvoyez `true` pour court-circuiter la vue si l'entité est manquante ou erronée.
- Utilisez les vues du nouveau corps de liste et des métadonnées de liste pour aider au formatage. Vous devriez n'utiliser quasiment aucun balisage dans ces vues.
- Mettez à jour la structure pour les actions - Exemple : le plugin Bookmarks.
- Fichiers d'action et noms d'action de l'espace de noms (exemple : `mod/blog/actions/blog/save.php => action/blog/save`)
- **Utilisez les URLs d'action suivantes :**

Objectif	URL
Ajouter	<code>action/plugin/save</code>
Modifier	<code>action/plugin/save</code>
Supprimer	<code>action/plugin/delete</code>

- Faites que l'action de suppression accepte `action/<handler>/delete?guid=<guid>` de sorte que le menu des métadonnées de l'entité ait la bonne URL par défaut

Actions

Les actions sont des états transitoires pour effectuer une action telle que la mise à jour de la base de données ou l'envoi d'une notification à un utilisateur. Utilisées correctement, les actions fournissent un niveau de contrôle d'accès et de prévention contre les attaques CSRF.

Les actions nécessitent que les jetons d'action (CSRF) soient soumis via GET/POST, mais ceux-ci sont ajoutés automatiquement par `elgg_view_form()`, et à l'aide de l'argument `is_action` de la vue `output/url`.

Meilleures pratiques pour les actions

Les fichiers d'action sont inclus dans le système d'action de Elgg ; comme les vues, *ce ne sont pas* des scripts classiques exécutables par les utilisateurs. Ne démarrez pas le noyau de Elgg dans votre fichier et ne renvoyez pas les utilisateurs directement vers ce fichier.

Comme les actions sont sensibles au temps, elles ne conviennent pas aux liens dans les e-mails ou autres notifications retardées. Un exemple de cela serait des invitations à rejoindre un groupe. La façon propre de créer un lien d'invitation est de créer un gestionnaire de page pour les invitations et les e-mails qui serait lié à l'utilisateur. Il incombe alors au gestionnaire de page de créer les liens d'action pour qu'un utilisateur accepte ou ignore la demande d'invitation.

Considérez que les actions peuvent être soumises par le biais de requêtes XHR, pas seulement par des liens ou des envois de formulaires.

Appeler directement un fichier

C'est très simple : **Ne le faites pas**. À l'exception de l'intégration d'applications tierces, il n'y a pas de raison d'appeler directement un fichier dans le répertoire des plugins.

Recommandé

Ces points sont de bonnes idées, mais ne sont pas encore dans les recommandations officielles. Suivre ces suggestions aidera à garder votre plugin compatible avec le noyau de Elgg.

- Mettez à jour les vues du widget (voir les widgets du blog ou des fichiers)
- Mettez à jour le “widget” du profil du groupe en prenant exemple sur les plugins blog ou file
- **Mettre à jour les formulaires**
 - Déplacez le corps des formulaires vers `/forms/<handler>/<action>` pour utiliser le nouveau `elgg_view_form()` d'Evan
 - Utilisez les vues de saisie dans le corps des formulaires, plutôt que du html
 - Ajoutez une fonction qui prépare le formulaire (voir `mod/file/lib/file.php` pour un exemple)
 - Intégrez des formulaires persistants (voir l'action de téléchargement du plugin de fichier et la fonction de préparation de formulaire)
- **Nettoyez le CSS/HTML**
 - Devrait être en mesure de supprimer presque tous les CSS (recherchez des motifs qui peuvent être déplacés dans le noyau si vous avez besoin de CSS)
- Utilisez les traits- d'union plutôt que les traits de soulignement (« underscore ») dans les classes/ids
- N'utilisez pas la catégorie `bundled` avec vos plugins. Elle est réservée aux plugins distribués avec Elgg
- N'utilisez pas `register_shutdown_function` car vous pourriez ne plus avoir accès à certaines parties de Elgg (par ex. la base de données). Utilisez plutôt l'événement `shutdown system`

3.3.15 Fonctions pratiques

Contenu

- *Entrée et sortie*
- *Méthode des entités*
- *Entité et récupération du contexte*
- *Plugins*
- *Interface et annotations*
- *Messages*
- *Formatage des adresses email*

Entrée et sortie

- `get_input($name)` Récupère des informations depuis un champ de formulaire (ou toute variable passée en utilisant GET ou POST). Se charge également d'assainir l'entrée, en retirant le javascript, etc.
- `set_input($name, $value)` Force une valeur pour une variable particulière destinée à être récupérée par la suite par `get_input()`

Méthode des entités

- `$entity->getURL()` Retourne l'URL de n'importe quelle entité dans le système
- `$entity->getGUID()` Retourne le GUID de n'importe quelle entité dans le système
- `$entity->canEdit()` Indique si l'utilisateur courant est autorisé ou non à modifier l'entité
- `$entity->getOwnerEntity()` Retourne le propriétaire `ElggUser` d'une entité particulière

Entité et récupération du contexte

- `elgg_get_logged_in_user_entity()` Retourne le `ElggUser` de l'utilisateur courant
- `elgg_get_logged_in_user_guid()` Retourne le GUID de l'utilisateur courant
- `elgg_is_logged_in()` Est-ce que le visiteur est identifié
- `elgg_is_admin_logged_in()` Est-ce que le visiteur est un admin identifié
- `elgg_gatekeeper()` Raccourci pour vérifier si un utilisateur est identifié. Redirige sur la page d'accueil du site si ce n'est pas le cas
- `elgg_admin_gatekeeper()` Raccourci pour vérifier si un utilisateur est identifié et est admin. Redirige sur la page d'accueil du site si ce n'est pas le cas
- `get_user($user_guid)` A partir d'un GUID, retourne l'entité `ElggUser` complète
- `elgg_get_page_owner_guid()` Retourne le GUID du propriétaire actuel, s'il y en a un
- `elgg_get_page_owner_entity()` Comme `elgg_get_page_owner_guid()` mais retourne l'entité complète
- `elgg_get_context()` Retourne le contexte courant de la page - par ex. « blog » pour le plugin blog, « thewire » pour le Fil, etc. Retourne « main » comme valeur par défaut
- `elgg_set_context($context)` Force le contexte pour une valeur particulière
- `elgg_push_context($context)` Ajoute un contexte sur la pile
- `elgg_pop_context()` Retire le dernier contexte de la pile
- `elgg_in_context($context)` Vérifie si vous êtes dans un contexte (ceci vérifie la pile complète, par ex. “widget” dans “groups”)

Plugins

- `elgg_is_active_plugin($plugin_id)` Vérifie si un plugin est installé et activé

Interface et annotations

- `elgg_view_image_block($icon, $info)` Retourne le résultat dans une liste formatée
- `elgg_view_comments($entity)` Retourne tous les commentaires associés à une entité donnée
- `elgg_get_friendly_time($unix_timestamp)` Retourne une date formatée de manière relative - « il y a 18 minutes », « il y a 2 jours », etc.

Messages

- `system_message($message)` Enregistre un message de succès
- `register_error($message)` Enregistre un message d'erreur
- `elgg_view_message($type, $message)` Affiche un message

Formatage des adresses email

Elgg dispose d'une classe d'aide pour récupérer des adresses email formatées : `\Elgg\Email\Address`.

```
// the constructor takes two variables
// first is the email address, this is REQUIRED
// second is the name, this is optional
$address = new \Elgg\Email\Address('example@elgg.org', 'Example');

// this will result in 'Example <example@elgg.org>'
echo $address->toString();

// to change the name use:
$address->setName('New Example');

// to change the e-mail address use:
$address->setEmail('example2@elgg.org');
```

Il existe plusieurs fonctions d'aide disponibles

- `\Elgg\Email\Address::fromString($string)` Va retourner une classe `\Elgg\Email\Address` avec le nom et l'email définis, à partir d'une chaîne formatée (par ex. `Exemple <exemple@elgg.org>`)
- `\Elgg\Email\Address::getFormattedEmailAddress($email, $name)` Va retourner une chaîne formatée à partir d'une adresse email et d'un nom optionnel

3.3.16 Liste des hooks de plugin du noyau

Pour plus d'informations sur le fonctionnement des hooks, visitez *Événements et Hooks des plugins*.

Contenu :local :

- *Liste des hooks de plugin du noyau*

Hooks système

page_owner, system Filtre le `page_owner` de la page courante. Aucune option n'est passée.

siteid, system

gc, system Autorise les plugins à exécuter la collecte des déchets (« garbage collection ») pour `$params['period']`.

unit_test, system Ajouter un test Simple Test. (Obsolète.)

diagnostics :report, system Filtre la sortie pour le téléchargement du rapport de diagnostic.

cron, <period> Déclenché par le cron pour chaque période.

cron :intervals, system Permet la configuration d'intervalles de cron personnalisés

validate, input Filtre l'entrée GET et POST. Ceci est utilisé par `get_input()` pour assainir les entrées utilisateur.

prepare, html Déclenché par `elgg_format_html()` et utilisé pour préparer du HTML non sûr.

La valeur `$return` est un tableau :

- `html` - La chaîne de caractères HTML en train d'être préparée
- `options` - Options de préparation

diagnostics :report, system Filtre la sortie pour un rapport de diagnostic.

debug, log Déclenché par le Logger. Retourne false pour arrêter la méthode de journalisation par défaut. \$params comprend :

- **level - Le niveau de débogage. Au choix parmi :**
 - Elgg_Logger::OFF
 - Elgg_Logger::ERROR
 - Elgg_Logger::WARNING
 - Elgg_Logger::NOTICE
 - Elgg_Logger::INFO
- msg - Le message
- display - Est-ce que ce message devrait être affiché ?

format, friendly :title Formate un titre « amical » pour les chaînes de caractères. Ceci est utilisé pour générer des URLs.

format, friendly :time Formate une date « amicale » pour le timestamp \$params['time'].

format, strip_tags Filtre une chaîne pour supprimer les balises. La chaîne d'origine est passée sous le nom de \$params['original_string'] et un ensemble facultatif de balises autorisées est passé sous le nom de \$params['allowed_tags'].

output :before, page Dans elgg_view_page(), ceci filtre \$vars avant qu'il ne soit transmis à la vue d'affichage de la coquille de la page (page/<page_shell>). Pour supprimer l'envoi de l'entête X-Frame-Options, dé-enregistrez le gestionnaire _elgg_views_send_header_x_frame_options() de ce hook.

output, page Dans elgg_view_page(), ceci filtre la valeur de retour renvoyée.

parameters, menu :<menu_name> Déclenché par elgg_view_menu(). Utilisé pour modifier les variables de menu (comme l'ordre de tri) avant le rendu.

Le tableau \$params va contenir :

- name - nom du menu
- sort_by - paramètre de tri préféré
- d'autres paramètres passés à elgg_view_menu()

register, menu :<menu_name> Filtre la liste initiale des éléments de menu issus de la configuration, avant que le menu n'ait été divisé en sections. Déclenché par elgg_view_menu() et elgg()->menus->getMenu().

Le tableau \$params contient des paramètres renvoyés par le hook parameters, menu:<menu_name>.

La valeur de retour est une instance d'objets \Elgg\Collections\Collection contenant des objets \ElggMenuItem.

Les gestionnaires de hooks peuvent ajouter/supprimer des éléments à une collection via l'API des collections, ainsi que des opérations sur le tableau des accès.

prepare, menu :<menu_name> Filtre le tableau des sections de menu avant qu'elles ne soient affichées. Chaque section est une clef de chaîne qui correspond à une zone d'éléments de menu. Il s'agit d'un bon hook pour trier, ajouter, supprimer et modifier les éléments de menu. Déclenché par elgg_view_menu() et elgg()->menus->prepareMenu().

Le tableau \$params va contenir :

- selected_item - ElggMenuItem sélectionné dans le menu, le cas échéant

La valeur de retour est une instance de \Elgg\Menu\PreparedMenu. Le menu préparé est une collection d'objets \Elgg\Menu\MenuSection, qui sont à leur tour des collections d'objets \ElggMenuItem.

register, menu :filter :<filter_id> Permet aux plugins de modifier les onglets de filtre pour les dispositions (layouts) qui spécifient le paramètre <filter_id>. Les paramètres et les valeurs de retour sont les mêmes que dans le hook register, menu:<menu_name>.

filter_tabs, <context> Filtre le tableau des ElggMenuItem utilisés pour afficher les onglets Tout/Moi/Contacts.

Le tableau \$params comprend :

- selected : le nom de l'élément de menu sélectionné
- user : le ElggUser identifié ou null
- vars : Les arguments \$vars passés à elgg_get_filter_tabs

creating, river Les options pour elgg_create_river_item sont filtrées à travers ce hook. Vous pouvez modifier des valeurs ou renvoyer false pour annuler la création de l'élément.

simplecache :generate, <view> Filtre la sortie de la vue pour une URL /cache quand simplecache est activé.

cache :generate, <view> Filtre la sortie de la vue pour une URL /cache quand simplecache est désactivé. Notez que cela sera déclenché pour chaque requête à `"/cache"` – aucun entête Expires n'est utilisé quand simplecache est désactivé.

prepare, breadcrumbs Dans `elgg_get_breadcrumbs()`, ceci filtre le fil d'ariane enregistré avant de le renvoyer, ce qui permet à un plugin de modifier la stratégie de fil d'ariane à l'échelle du site. Le tableau `$params` comprend :

- `chapelure` - un tableau des éléments du Fil d'Ariane, chacun avec les clefs touches `title` et `link`
- `identifier` - identifiant de la route de la page courante
- `segments` - segments de la route de la page courante

add, river

elgg.data, site Filtre les données de configuration mises en cache pour les transmettre au client. *Plus d'informations*

elgg.data, page Filtre les données de configuration non mises en cache et spécifiques à la page à transmettre au client. *Plus d'informations*

registration_url, site Filtre l'URL d'inscription sur le site. Peut être utilisé par les plugins pour ajouter des codes d'invitation, des codes de référence, etc. à l'URL d'inscription. Le tableau `$params` contient un tableau des éléments de la requête ajoutés à l'URL d'inscription par le script qui l'invoque. Le hook doit renvoyer une URL absolue de la page d'inscription.

login_url, site Filtre l'URL de connexion du site. Le tableau `$params` contient un tableau des éléments de requête ajoutés à l'URL de connexion par le script qui l'invoque. Le hook doit renvoyer une URL absolue de la page de connexion.

commands, cli Permet aux plugins d'enregistrer leurs propres commandes exécutables via le binaire `elgg-cli`. Les gestionnaires doivent renvoyer un tableau de noms de classes de commande. Les commandes doivent étendre `\Elgg\Cli\Command` pour être exécutables.

seeds, database Permet aux plugins d'enregistrer leurs propres semences de base de données. Les graines remplissent la base de données avec de fausses entités à des fins de test. Les graines doivent étendre la classe `\Elgg\Database\Seeds\Seed` pour qu'elle soit exécutable via `elgg-cli database:seed`.

languages, translations Permet aux plugins d'ajouter/supprimer des langues des langues configurables du système.

generate, password Permet aux plugins de générer de nouveaux mots de passe texte aléatoires.

Hooks utilisateur

usersettings :save, user Déclenché dans l'action globale pour enregistrer les paramètres utilisateur. Le gestionnaire de hook doit renvoyer `false` pour éviter que les formulaires persistants ne soient effacés (c.-à-d. pour indiquer que certaines valeurs n'ont pas été enregistrées). Ne renvoyez pas `true` dans votre gestionnaire de hook car vous remplacerez la sortie d'autres hooks : renvoyez plutôt `null` pour indiquer une opération réussie.

Le tableau `$params` va contenir :

- `user` - `\ElggUser`, dont les paramètres ont été sauvegardés
- `request` - `\Elgg\Request` vers le contrôleur de l'action

change :email, user Déclenché avant que l'email de l'utilisateur soit modifié. Permet aux plugins d'implémenter une logique additionnelle nécessaire pour modifier l'email, par ex. une validation de l'email additionnelle. Le gestionnaire de hook handler doit retourner `false` pour éviter que l'email ne soit modifié immédiatement.

Le tableau `$params` va contenir :

- `user` - `\ElggUser`, dont les paramètres ont été sauvegardés
- `email` - Adresse email qui passe les vérifications sanitaires
- `request` - `\Elgg\Request` vers le contrôleur de l'action

access :collections :write, user Filtre un tableau d'autorisations d'accès pour lesquelles l'utilisateur \$params['user_id'] est autorisé à enregistrer du contenu. Les autorisations renvoyées sont de la forme (id => "Nom lisible par un Humain").

registeruser :validate :username, all Renvoie un booléen si la chaîne dans \$params['username'] est un nom d'utilisateur valide. Le gestionnaire de hook peut lancer \RegistrationException avec un message d'erreur à montrer à l'utilisateur.

registeruser :validate :password, all Renvoie un booléen si la chaîne dans \$params['password'] est un mot de passe valide. Le gestionnaire de hook peut lancer \RegistrationException avec un message d'erreur à montrer à l'utilisateur.

registeruser :validate :email, all Renvoie un booléen si la chaîne dans \$params['email'] est une adresse e-mail valide. Le gestionnaire de hook peut lancer \RegistrationException avec un message d'erreur à montrer à l'utilisateur.

register, user Déclenché par l'action `register` après que l'utilisateur se soit inscrit. Renvoie `false` pour supprimer l'utilisateur. Notez que la fonction `register_user` ne déclenche pas ce hook. Les gestionnaires de hooks peuvent lancer une exception \RegistrationException avec un message d'erreur à afficher à l'utilisateur.

Le tableau \$params va contenir :

- `user` - L'entité utilisateur nouvellement enregistrée
- Tous les paramètres envoyés avec la requête à l'action (comprend `password`, `friend_guid`, `invitecode`, etc.)

login :forward, user Filtre l'URL vers laquelle l'utilisateur sera dirigé après la connexion.

find_active_users, system Renvoie le nombre d'utilisateurs actifs.

status, user Déclenché par The Wire (le Fil) lors de l'ajout d'une publication.

username :character_blacklist, user Filtre la chaîne et supprimer les caractères interdits pour valider le nom d'utilisateur lors de l'inscription. La valeur de retour doit être une chaîne composée des caractères refusés. La chaîne par défaut se trouve dans \$params['blacklist'].

Hooks des objets

comments, <entity_type> Déclenché dans `elgg_view_comments()`. Si vous renvoyez du contenu, cela remplace la vue `page/elements/comments`.

comments :count, <entity_type> Renvoie le nombre de commentaires sur \$params['entity'].

likes :count, <entity_type> Renvoie le nombre de mentions J'aime pour \$params['entity'].

Hooks d'accès

access_collection :url, access_collection Peut être utilisé pour filtrer l'URL de la collection d'accès.

Le tableau \$params va contenir :

- `access_collection` - *ElggAccessCollection*

access_collection :name, access_collection Peut être utilisé pour filtrer le nom d'affichage (niveau d'accès lisible) de la collection d'accès.

Le tableau \$params va contenir :

- `access_collection` - *ElggAccessCollection*

access :collections :read, user Filtre un tableau d'ID d'accès que l'utilisateur \$params['user_id'] peut voir.

Avvertissement : Le gestionnaire doit soit ne pas utiliser les parties de l'API qui utilisent le système d'accès (ce qui déclencherait à nouveau le hook), soit ignorer le deuxième appel. Sinon, une boucle infinie sera créée.

access :collections :write, user Filtre un tableau d'ID d'accès sur lequel l'utilisateur `$params['user_id']` peut écrire. Dans `get_write_access_array()`, ce hook filtre la valeur de retour, de sorte qu'il peut être utilisé pour modifier les options disponibles dans la vue `input/access`. Pour les plugins du noyau, « `input_params` » est passé avec les clefs « `entity` » (`ElggEntity`|false), « `entity_type` » (chaîne), « `entity_subtype` » (chaîne), et « `container_guid` » (int). Une valeur d'entité vide signifie généralement que le formulaire consiste à créer un nouvel objet.

Avertissement : Le gestionnaire doit soit ne pas utiliser les parties de l'API qui utilisent le système d'accès (ce qui déclencherait à nouveau le hook), soit ignorer le deuxième appel. Sinon, une boucle infinie sera créée.

access :collections :write :subtypes, user Renvoie un tableau de sous-types de collection d'accès à utiliser lors de la récupération des collections d'accès appartenant à un utilisateur, dans le cadre de la fonction `get_write_access_array()`.

access :collections :addcollection, collection Déclenché après la création d'une collection d'accès `$params['collection_id']`.

access :collections :deletecollection, collection Déclenché avant qu'une collection d'accès `$params['collection_id']` soit supprimée. Renvoyez false pour éviter la suppression.

access :collections :add_user, collection Déclenché avant d'ajouter l'utilisateur `$params['user_id']` à la collection `$params['collection_id']`. Retournez false pour éviter l'ajout.

access :collections :remove_user, collection Déclenché avant de supprimer l'utilisateur `$params['user_id']` de la collection `$params['collection_id']`. Renvoyez false pour éviter le retrait.

get_sql, access Filtre les clauses SQL limitant/permettant l'accès aux entités et aux annotations.

Le hook est déclenché indépendamment du fait que les accès soient ignorés. Les gestionnaires peuvent avoir besoin de vérifier si l'accès est ignoré et de renvoyer un retour plus tôt, si les clauses ajoutées ne doivent s'appliquer qu'aux contextes d'accès contrôlés.

La valeur `$return` est un tableau imbriqué de ands et de ors.

`$params` comprend :

- `table_alias` - alias de la table principale utilisée dans la clause de sélection
- `ignore_access` - selon que les accès sont ignorés ou activés
- `use_enabled_clause` - selon que les entités désactivées sont affichées/cachées
- `access_column` - colonne de la table principale contenant l'ID de la collection d'accès
- `owner_guid_column` - colonne de la table principale faisant référence au GUID du propriétaire
- `guid_column` - colonne de la table principale faisant référence au GUID de l'entité
- `enabled_column` - colonne de la table principale faisant référence à l'état activé de l'entité
- `query_builder` - une instance du `QueryBuilder`

Hooks d'action

action, <action> Obsolète. Utilisez plutôt le hook `'action:validate', <action>`. Déclenché avant d'exécuter des scripts d'action. Renvoyez false pour interrompre l'action.

action :validate, <action>

Déclenché avant l'exécution du script/contrôleur d'action. Ce hook doit être utilisé pour valider/modifier l'entrée de l'utilisateur, avant de procéder à l'action. Le gestionnaire de hook peut lancer une instance de `\Elgg\ValidationException` ou renvoyer false pour bloquer la poursuite de l'exécution.

le tableau `$params` comprend :

- `request` - instance de `\Elgg\Request`

action_gatekeeper :permissions :check, all Déclenché après la validation d'un jeton CSRF. Renvoyez false pour empêcher la validation.

action_gatekeeper :upload_exceeded_msg, all Déclenché lorsqu'un POST dépasse la taille maximale autorisée par le serveur. Renvoyez un message d'erreur à afficher.

forward, <reason> Filtre l'URL pour rediriger un utilisateur lorsque `forward($url, $reason)` est appelé. Dans certains cas, le tableau `params` va contenir une instance de `HttpException` qui a déclenché l'erreur.

response, action :<action> Filtre une instance de `\Elgg\Http\ResponseBuilder` avant qu'elle soit envoyée au client. Ce hook peut être utilisé pour modifier le contenu de réponse, le code d'état, l'URL de redirection, ou définir des entêtes de réponse supplémentaires. Notez que la valeur `<action>` est analysée à partir de l'URL de requête, par conséquent vous pouvez ne pas être en mesure de filtrer les réponses des appels à `action()` si elles sont imbriquées dans un autre fichier de script d'action.

Ajax

ajax_response, * Quand le module AMD `elgg/Ajax` est utilisé, ce hook donne accès à l'objet de réponse (`\Elgg\Services\AjaxResponse`) afin qu'il puisse être modifié/étendu. Le type de hook dépend de l'appel de méthode :

méthode elgg/Ajax	type de hook de plugin
<code>action()</code>	<code>action :<action_name></code>
<code>path()</code>	<code>path :<url_path></code>
<code>view()</code>	<code>view :<view_name></code>
<code>form()</code>	<code>form :<action_name></code>

output, ajax Ceci filtre l'emballage de sortie JSON renvoyé à l'API ajax héritée (`elgg.ajax`, `elgg.action`, etc.). Les plugins peuvent modifier la sortie, l'URL de redirection, les messages système et les erreurs. Pour le module AMD `elgg/Ajax`, utilisez le hook `ajax_response` documenté ci-dessus.

Hooks des Permissions

container_logic_check, <entity_type> Déclenché par `ElggEntity::canWriteToContainer()` avant de déclencher les hooks `permissions_check` et `container_permissions_check`. Contrairement aux hooks d'autorisations, la vérification logique peut être utilisée pour empêcher que certains types d'entités ne soient contenus par d'autres types d'entités, par exemple les réponses de discussion ne doivent être contenues que par les discussions. Ce hook peut également être utilisé pour appliquer une logique de statut, par exemple pour refuser de nouvelles réponses pour les discussions fermées.

Le gestionnaire devrait renvoyer `false` pour empêcher une entité de contenir une autre entité. La valeur par défaut transmise au hook est `null`, de sorte que le gestionnaire peut vérifier si un autre hook a modifié la valeur en vérifiant si la valeur de retour est définie. Si ce crochet renvoie `false`, les hooks `container_permissions_check` et `permissions_check` ne seront pas déclenchés.

Le tableau `$params` va contenir :

- `container` - Une entité qui sera utilisée comme conteneur
- `user` - L'utilisateur qui sera propriétaire de l'entité à écrire sur le conteneur
- `subtype` - Sous-type de l'entité à écrire dans le conteneur (le type d'entité est identifié à partir du type de hook)

container_permissions_check, <entity_type> Renvoyez un booléen pour indiquer si l'utilisateur `$params['user']` peut utiliser l'entité `$params['container']` comme conteneur pour une entité de `<entity_type>` et un sous-type `$params['subtype']`.

Dans les rares cas où une entité est créée avec ni le `container_guid` ni le `owner_guid` correspondant à l'utilisateur connecté, ce hook est appelé *deux fois*, et dans le premier appel `$params['container']` aura pour valeur le *propriétaire*, et non le véritable conteneur de l'entité.

Le tableau `$params` va contenir :

- `container` - Une entité qui sera utilisée comme conteneur

- **user** - L'utilisateur qui sera propriétaire de l'entité à écrire sur le conteneur
- **subtype** - Sous-type de l'entité à écrire dans le conteneur (le type d'entité est identifié à partir du type de hook)

permissions_check, <entity_type> Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut modifier l'entité `$params['entity']`.

permissions_check:delete, <entity_type> Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut supprimer l'entité `$params['entity']`. Vaut par défaut `$entity->canEdit()`.

permissions_check:delete, river Renvoiez un booléen pour inquier si l'utilisateur `$params['user']` peut supprimer l'élément de la rivière `$params['item']`. Par défaut à `true` pour les administrateurs et `false` pour les autres utilisateurs.

permissions_check:download, file Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut télécharger le fichier dans `$params['entity']`.

Le tableau `$params` va contenir :

- **entity** - Instance de `ElggFile`
- **user** - Utilisateur qui va télécharger le fichier

permissions_check, widget_layout Renvoyez un booléen pour indiquer si `$params['user']` peut modifier les widgets dans le contexte passé par `$params['context']` et avec un propriétaire de page `$params['page_owner']`.

permissions_check:metadata, <entity_type> (Déprécié) Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut modifier les métadonnées `$params['metadata']` sur l'entité `$params['entity']`.

permissions_check:comment, <entity_type> Renvoyer un booléen pour indiquer si l'utilisateur `$params['user']` peut commenter l'entité `$params['entity']`.

permissions_check:annotate:<annotation_name>, <entity_type> Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut créer une annotation `<annotation_name>` sur l'entité `$params['entity']`. Si connecté, la valeur par défaut est `true`.

Note : Ceci est appelé avant le hook plus général `permissions_check:annotated`, et sa valeur de retour est la valeur initiale de ce hook.

permissions_check:annotate, <entity_type> Renvoiez un booléen pour indiquer si l'utilisateur `$params['user']` peut créer une annotation `$params['annotation_name']` sur l'entité `$params['entity']`. Si connecté, la valeur par défaut est `true`.

permissions_check:annotation Renvoiez un booléen pour indiquer si l'utilisateur dans `$params['user']` peut modifier l'annotation `$params['annotation']` sur l'entité `$params['entity']`. L'utilisateur peut être `null`.

fail, auth Renvoiez le message d'échec en cas d'échec de l'authentification. Un tableau des méthodes d'échec PAM précédentes est passé dans `$params`.

api_key, use Déclenché par `api_auth_key()`. Renvoyer `false` empêche d'authentifier la clé.

gatekeeper, <entity_type>:<entity_subtype> Filtre le résultat de `elgg_entity_gatekeeper()` pour empêcher ou permettre l'accès à une entité à laquelle l'utilisateur aurait ou n'aurait pas accès. Un gestionnaire peut renvoyer `false` ou une instance de `HttpException` pour bloquer l'accès à une entité. Un gestionnaire peut retourner `true` pour remplacer le résultat du gardien (gatekeeper). **Important** l'entité reçue par ce hook est récupérée sans vérifier les accès, y compris les entités désactivées, de sorte que vous devez faire attention à ne pas contourner le système d'accès.

le tableau `$params` comprend :

- **entity** - Entité à laquelle on accède
- **user** - L'utilisateur qui accède à l'entité (`null` implique l'utilisateur connecté)

Notifications

Ces hooks sont listés dans l'ordre chronologique de la vie de l'événement de notification. Notez que tous les hooks ne s'appliquent pas aux notifications instantanées.

enqueue, notification Peut être utilisé pour empêcher un événement de notification d'envoyer des notifications **subscription**. Le gestionnaire de hook doit renvoyer `false` pour empêcher le déclenchement d'un événement de notification d'abonnement.

le tableau `$params` comprend :

- `object` - objet de l'événement de notification
- `action` - action qui a déclenché l'événement de notification. Par exemple, correspond à `publish` quand `elgg_trigger_event('publish', 'object', $object)` est appelé

get, subscriptions

Filtre les abonnés de l'événement de notification. S'applique aux notifications **subscriptions** et **instant**.

Lors d'un événement d'abonnement, par défaut, la liste des abonnés se compose des utilisateurs abonnés à l'entité conteneur de l'objet de l'événement. En cas d'événement de notification instantanée, la liste des abonnés se compose des utilisateurs passés en tant que destinataires à `notify_user()`

IMPORTANT Validez toujours les types d'événements, d'objets et/ou d'action de notification avant d'ajouter de nouveaux destinataires pour vous assurer que vous n'envoyez pas accidentellement de notifications à des destinataires non souhaités. Considérez une situation où un plugin de mentions envoie une notification instantanée à un utilisateur mentionné - tout hook agissant sur un sujet ou un objet sans valider un événement ou un type d'action (par exemple, y compris un propriétaire du fil d'origine) peut finir par envoyer des notifications aux mauvais utilisateurs.

le tableau `$params` comprend :

- `event` - instance de `\Elgg\Notifications\NotificationEvent` qui décrit l'événement de notification
- `origin` - `subscriptions_service` ou `instant_notifications`
- `methods_override` - préférence de méthode de remise pour les notifications instantanées

Les gestionnaires doivent renvoyer un tableau de la forme :

```
array(  
    <user_guid> => array('sms'),  
    <user_guid2> => array('email', 'sms', 'ajax')  
);
```

send :before, notifications Déclenché avant le traitement de la file d'attente des événements de notification. Peut être utilisé pour mettre fin à l'événement de notification. S'applique aux notifications **subscriptions** et **instant**.

le tableau `$params` comprend :

- `event` - instance de `\Elgg\Notifications\NotificationEvent` qui décrit l'événement de notification
- `subscriptions` - une liste d'abonnements. Voir le hook `'get', 'subscriptions'` pour plus d'informations

prepare, notification Un hook de haut niveau qui peut être utilisé pour modifier une instance de `\Elgg\Notifications\Notification` avant qu'elle soit envoyée à l'utilisateur. S'applique aux notifications **subscriptions** et **instant**. Ce hook est déclenché avant un `'prepare', 'notification':<action>:<entity_type>:<entity_subtype>` plus granulaire et après `'send:before', 'notifications`. Le gestionnaire de hook doit renvoyer un objet de notification modifié.

`$params` peut varier en fonction du type de notification et peut comprendre :

- `event` - instance de `\Elgg\Notifications\NotificationEvent` qui décrit l'événement de notification
- `object` - objet de la notification event. Peut être null pour les notifications instantanées
- `action` - action qui a déclenché la notification event. Peut avoir pour valeur par défaut `notify_user` pour les notifications instantanées
- `method` - méthode d'envoi (par ex. email, site)

- `sender` - expéditeur
- `recipient` - destinataire
- `language` - langue de la notification (langue du destinataire)
- `origin` - `subscriptions_service` ou `instant_notifications`

prepare, notification :<action> :<entity_type> :<entity_type> Un hook granulaire pouvant être utilisé pour filtrer une notification `\Elgg\Notifications\Notification` avant qu'elle soit envoyée à l'utilisateur. S'applique aux notifications **subscriptions** et **instant**. En cas de notifications instantanées qui n'ont pas reçu d'objet, le hook sera appelé par `'prepare'`, `'notification:<action>'`. En cas de notifications instantanées qui n'ont pas reçu de nom d'action, celui-ci aura pour valeur par défaut `notify_user`.

\$params comprend :

- `event` - instance de `\Elgg\Notifications\NotificationEvent` qui décrit l'événement de notification
- `object` - objet de la notification `event`. Peut être null pour les notifications instantanées
- `action` - action qui a déclenché la notification `event`. Peut avoir pour valeur par défaut `notify_user` pour les notifications instantanées
- `method` - méthode d'envoi (par ex. `email`, `site`)
- `sender` - expéditeur
- `recipient` - destinataire
- `language` - langue de la notification (langue du destinataire)
- `origin` - `subscriptions_service` ou `instant_notifications`

format, notification :<method> Ce hook peut être utilisé pour formater une notification avant qu'elle soit transmise au hook `'send'`, `'notification:<method>'`. S'applique aux notifications **subscriptions** et **instant**. Le gestionnaire de hook doit renvoyer une instance de `\Elgg\Notifications\Notification`. Le hook ne reçoit pas de \$params. Voici quelques cas d'utilisation :

- Supprime les balises du titre et du corps de la notification pour les notifications par email en texte brut
- Styles HTML en ligne (`inline`) pour les emails de notifications HTML
- Envelopper la notification dans un modèle, ajouter une signature, etc.

send, notification :<method> Délivre une notification. S'applique aux notifications **subscriptions** et **instant**. Le gestionnaire doit retourner `true` ou `false` pour indiquer le résultat de la remise.

le tableau \$params comprend :

- `notification` - un objet de notification `\Elgg\Notifications\Notification`

send :after, notifications Déclenché après que toutes les notifications dans la file d'attente pour l'événement notifications ont été traitées. S'applique aux notifications **subscriptions** et **instant**.

le tableau \$params comprend :

- `event` - instance de `\Elgg\Notifications\NotificationEvent` qui décrit l'événement de notification
- `subscriptions` - une liste d'abonnements. Voir le hook `'get'`, `'subscriptions'` pour plus d'informations
- `deliveries` - une matrice des statuts de remise par utilisateur pour chaque méthode de livraison

Emails

prepare, system :email Déclenché par `elgg_send_email()`. S'applique à tous les emails système et de notification sortants. Ce hook vous permet de modifier une instance de `\Elgg\Email` avant qu'il soit transmis à l'agent de transport par email. Ce hook peut être utilisé pour modifier l'expéditeur, le destinataire, le sujet, le corps et/ou les entêtes de l'email.

\$params est vide. La valeur de retour `$return` est une instance de `\Elgg\Email`.

validate, system :email Déclenché par `elgg_send_email()`. S'applique à tous les emails système et de notification sortants. Ce hook vous permet de supprimer ou de mettre en liste blanche les emails sortants, par exemple lorsque le site est en mode de développement. Le gestionnaire doit retourner `false` pour empêcher la remise du courrier électronique.

\$params contient :

- `email` - Une instance de `\Elgg\Email`

transport, system :email Déclenché par `elgg_send_email()`. S'applique à tous les emails système et de notification sortants. Ce hook vous permet d'implémenter un transport de messagerie personnalisé, par exemple en livrant des emails via un service proxy tiers tel que SendGrid ou Mailgun. Le gestionnaire doit retourner `true` pour indiquer que l'email a bien été transporté.

\$params contient :

— email - Une instance de `\Elgg\Email`

zend :message, system :email Déclenché par le gestionnaire de transport de messagerie (MTA) par défaut (Elgg utilise `zendframework/zend-mail`). S'applique à tous les emails de système et de notification sortants qui n'ont pas été transportés à l'aide du hook **transport, system :email**. Ce hook vous permet de modifier une instance de `\Zend\Mail\Message` avant qu'elle soit transmise au transport de messagerie Zend.

\$params contient :

— email - Une instance de `\Elgg\Email`

Routeage

route :config, <route_name> Permet de modifier la configuration de la route avant qu'elle soit enregistrée. Ce hook peut être utilisé pour modifier le chemin d'accès, les valeurs par défaut, les exigences, ainsi que pour définir/supprimer un middleware. Veuillez noter que le gestionnaire de ce hook doit être enregistré à l'extérieur du gestionnaire d'événements `init`, car les routes du noyau sont enregistrées lors de l'événement `plugins_boot`.

route :rewrite, <identifiant> Permet de modifier le chemin d l'URL relative au site pour une requête entrante. Pour plus de détails, consultez [Routeage](#). Veuillez noter que le gestionnaire de ce hook doit être enregistré à l'extérieur du gestionnaire d'événements `init`, car les réécritures de routes ont lieu après la fin de l'événement `plugins_boot`.

response, path :<path> Filtrer une instance de `\Elgg\Http\ResponseBuilder` avant qu'elle soit envoyée au client. Ce type de hook ne sera utilisé que si le chemin n'a pas commencé par `action/` ou `ajax/`. Ce hook peut être utilisé pour modifier le contenu de réponse, le code d'état, l'URL de redirection, ou définir des entêtes de réponse supplémentaires. Notez que la valeur `<path>` est analysée à partir de l'URL de la requête, aussi les plugins qui utilisent le hook `route` doivent utiliser le `<path>` d'origine pour filtrer la réponse, ou utiliser plutôt le hook `route:rewrite`.

ajax_response, path :<path> Filtre les réponses ajax avant qu'elles soient renvoyées au module `elgg/Ajax`. Ce type de hook ne sera utilisé que si le chemin n'a pas commencé par `action/` ou `ajax/`.

Vues

view_vars, <view_name> Filtre le tableau `$vars` passé à la vue

view, <view_name> Filtre le contenu renvoyé par la vue

layout, page Dans `elgg_view_layout()`, filtre le nom de la disposition. Le tableau `$params` comprend :

- `identifiant` - identifiant de la page en cours de rendu
- `segments` - Segments d'URL de la page en cours de rendu
- autres `$vars` reçues par `elgg_view_layout()`

shell, page Dans `elgg_view_page()`, filtre le nom de la coquille (page shell) de page

head, page Dans `elgg_view_page()`, filtre la valeur de retour de `$vars['head']`, qui contient un tableau avec `title`, `metas` et `links` où `metas` est un tableau d'éléments à formater sous la forme de balises `head <meta>`, et `links` est un tableau d'éléments à formater sous la forme de `<link>`. Chaque élément meta et link contient un ensemble de paires de clés/valeurs qui sont formatées en attributs de balise html, par ex.

```
return [  
    'title' => 'Current page title',  
    'metas' => [  

```

(suite sur la page suivante)

(suite de la page précédente)

```

        'viewport' => [
            'name' => 'viewport',
            'content' => 'width=device-width',
        ]
    ],
    'links' => [
        'rss' => [
            'rel' => 'alternative',
            'type' => 'application/rss+xml',
            'title' => 'RSS',
            'href' => elgg_format_url($url),
        ],
        'icon-16' => [
            'rel' => 'icon',
            'sizes' => '16x16',
            'type' => 'image/png',
            'href' => elgg_get_simplecache_url('graphics/favicon-16.png'),
        ],
    ],
];

```

ajax_response, view :<view> Filtre les réponses ajax/view/ avant qu'elles soient renvoyées au module elgg/Ajax.

ajax_response, form :<action> Filtre les réponses ajax/form/ avant qu'elles soient renvoyées au module elgg/Ajax.

response, view :<view_name> Filtre une instance de \Elgg\Http\ResponseBuilder avant qu'elle soit envoyée au client. S'applique à la requête vers /ajax/view/<view_name>. Ce hook peut être utilisé pour modifier le contenu de réponse, le code d'état, l'URL de redirection, ou définir des entêtes de réponse supplémentaires.

response, form :<form_name> Filtre une instance de \Elgg\Http\ResponseBuilder avant qu'elle soit envoyée au client. S'applique à la requête vers /ajax/form/<form_name>. Ce hook peut être utilisé pour modifier le contenu de la réponse, le code d'état, l'URL de redirection, ou définir des entêtes de réponse supplémentaires.

table_columns :call, <name> Quand la méthode elgg()->table_columns->\$name() est appelée, ce hook est appelé pour permettre aux plugins de remplacer ou de fournir une implémentation. Les gestionnaires reçoivent les arguments de la méthode via \$params['arguments'] et devraient renvoyer une instance de Elgg\Views\TableColumn s'ils souhaitent spécifier la colonne directement.

vars :compiler, css Permet aux plugins de modifier les variables CSS transmises à CssCrush pendant la compilation. Voir *Variables CSS* <_guides/theming#css-vars>.

Fichiers

download :url, file

Permet aux plugins de filtrer l'URL de téléchargement du fichier. Par défaut, l'URL de téléchargement est générée par le service de fichiers.

le tableau \$params comprend :

— entity - instance de ElggFile

inline :url, file

Permet aux plugins de filtrer l'URL en ligne du fichier image. Par défaut, l'URL en ligne (inline) est générée par le service de fichiers.

le tableau `$params` comprend :

— `entity` - instance de `ElggFile`

mime_type, file Retournez le mimetype pour le nom de fichier `$params['filename']` avec le nom de fichier original `$params['original_filename']` et avec le mimetype détecté par défaut depuis `$params['default']`.

simple_type, file Le hook fournit `$params['mime_type']` (par ex. `application/pdf` ou `image/jpeg`) et détermine une catégorie globale comme `document` ou `image`. Le plugin `file` intégré et d'autres plugins tiers stockent habituellement les métadonnées `simpletype` pour les entités de fichiers et les utilisent lorsqu'ils servent des icônes et construisent des filtres et des menus.

upload, file Permet aux plugins d'implémenter une logique personnalisée pour déplacer un fichier téléchargé dans une instance d'`ElggFile`. Le gestionnaire doit renvoyer `true` pour indiquer que le fichier téléchargé a été déplacé. Le gestionnaire doit renvoyer `false` pour indiquer que le fichier téléchargé n'a pas pu être déplacé. D'autres valeurs de retour indiqueront que `ElggFile::acceptUploadedFile` devrait poursuivre la logique de téléchargement par défaut.

le tableau `$params` comprend :

— `file` - instance de `ElggFile` dans laquelle écrire

— `upload` - instance de `UploadedFile` de Symfony

Recherche

search :results, <search_type> Déclenché par `elgg_search()`. Reçoit des options normalisées adaptées pour l'appel de `elgg_get_entities()` et doit renvoyer un tableau d'entités correspondant aux options de recherche. Ce hook est conçu pour être utilisé par des plugins intégrant des services d'indexation tiers, tels que Solr et Elasticsearch.

search :params, <search_type> Déclenché par `elgg_search()`. Filtre les paramètres de recherche (requête, tri, champs de recherche, etc.) avant que les clauses de recherche ne soient préparées pour un type de recherche donné. Le noyau de Elgg ne prend en charge que le type de recherche `entities`.

search :fields, <entity_type> Déclenché par `elgg_search()`. Filtre les champs de recherche avant que les clauses de recherche ne soient préparées. La valeur `$return` contient un tableau avec les noms de chaque type de propriété d'entité, qui doit être comparé à la requête de recherche. Le tableau `$params` contient un tableau de paramètres de recherche transmis et filtrés par `elgg_search()`.

```
return [  
    'attributes' => [],  
    'metadata' => ['title', 'description'],  
    'annotations' => ['revision'],  
    'private_settings' => ['internal_notes'],  
];
```

search :fields, <entity_type> :<entity_subtype> Voir **search :fields, <entity_type>**

search :fields, <search_type> Voir **search :fields, <entity_type>**

search :options, <entity_type> Déclenché par `elgg_search()`. Prépare les clauses de recherche (options) à transmettre à `elgg_get_entities()`.

search :options, <entity_type> :<entity_subtype> Voir **search :options, <entity_type>**

search :options, <search_type> Voir **search :options, <entity_type>**

search :config, search_types Implémenté dans le plugin **search**. Filtre un tableau de types de recherche personnalisés. Cela permet aux plugins d'ajouter des types de recherche personnalisés (par ex. une recherche de mots-clés ou par localisation géographique). L'ajout d'un type de recherche personnalisé permettra d'étendre l'interface utilisateur du plugin de recherche avec des liens et des listes appropriés.

search :config, type_subtype_pairs Implémenté dans le plugin **search**. Filtre les paires de types/sous-types d'entités avant l'exécution de la recherche d'entité. Permet aux plugins de supprimer certains types/sous-types d'entités des résultats de recherche, de regrouper plusieurs sous-types ou de réorganiser des sections de recherche.

search :format, entity Implémenté dans le plugin **search**. Permet aux plugins de renseigner les données volatiles de l'entité avant qu'elles ne soient transmises à la vue de recherche. Ceci est utilisé pour mettre en surbrillance les correspondances de la recherche, extraire les sous-chaînes pertinentes dans les champs de texte long, etc.

Autres

config, comments_per_page Filtre le nombre de commentaires affichés par page. Vaut par défaut 25. `$params['entity']` indique l'entité contenante, ou null si elle n'est pas fournie.

config, comments_latest_first Filtre l'ordre des commentaires. Vaut `true` par défaut pour le plus récent en premier. `$params['entity']` indique l'entité contenante, ou null si elle n'est pas fournie.

default, access Dans `get_default_access()`, ce hook filtre la valeur de retour, de sorte qu'il peut être utilisé pour modifier la valeur par défaut dans la vue `input/access`. Pour les plugins du noyau, la valeur « `input_params` » contient les clefs « `entity` » (`ElggEntity`/false), « `entity_type` » (chaîne), `entity_subtype` (chaîne), `container_guid` (int) qui sont fournies. Une valeur d'entité vide signifie généralement que le formulaire consiste à créer un nouvel objet.

classes, icon Peut être utilisé pour filtrer les classes CSS appliquées aux glyphes d'icônes. Par défaut, Elgg utilise FontAwesome. Les plugins peuvent utiliser ce hook pour utiliser une autre famille de polices et mapper les classes d'icônes.

entity :icon :sizes, <entity_type> Déclenché par `elgg_get_icon_sizes()` et définit le type d'entité/le sous-type de tailles d'icônes spécifiques. `entity_subtype` sera transmis à la fonction de rappel avec le tableau `$params`.

entity :<icon_type> :sizes, <entity_type>

Permet de filtrer les tailles pour les types d'icônes personnalisés, voir `entity:icon:sizes, <entity_type>`.

Le hook doit renvoyer un tableau associatif où les clefs sont les noms des tailles d'icônes (par exemple `large`), et les valeurs sont des tableaux avec les clefs suivantes :

- `w` - Largeur de l'image en pixels
- `h` - Hauteur de l'image en pixels
- `square` - Le ratio devrait-il être carré (vrai/faux)
- `upscale` - L'image doit-elle être agrandie si elle est plus petite que la largeur et la hauteur indiquées (vrai/faux)
- `crop` - Est-ce que le recadrage est autorisé sur cette taille d'image (vrai/faux, par défaut : vrai)

Si le tableau de configuration d'une taille d'image est vide, l'image sera enregistrée en tant que copie exacte de la source sans redimensionnement ni recadrage.

Exemple :

```
return [
    'small' => [
        'w' => 60,
        'h' => 60,
        'square' => true,
        'upscale' => true,
    ],
    'large' => [
        'w' => 600,
        'h' => 600,
        'upscale' => false,
```

(suite sur la page suivante)

```
],
    'original' => [],
];
```

entity:icon:url, <entity_type> Déclenché lorsque l'URL de l'icône de l'entité est demandée, voir *entity icons*. La fonction de rappel doit renvoyer l'URL de l'icône de taille `$params['size']` pour l'entité `$params['entity']`. Les paramètres suivants sont disponibles via le tableau `$params` :

entity Entité pour laquelle l'url de l'icône est demandée.

viewtype Le type de *vue* par exemple `default` ou `json`.

size Taille demandée, voir *icônes des entités* pour les valeurs possibles.

Exemple sur comment mettre en place une icône par défaut Gravatar pour les utilisateurs qui n'ont pas encore téléchargé un avatar :

```
// Priority 600 so that handler is triggered after avatar handler
elgg_register_plugin_hook_handler('entity:icon:url', 'user', 'gravatar_icon_handler', 600);

/**
 * Default to icon from gravatar for users without avatar.
 */
function gravatar_icon_handler($hook, $type, $url, $params) {
    // Allow users to upload avatars
    if ($params['entity']->icontime) {
        return $url;
    }

    // Generate gravatar hash for user email
    $hash = md5(strtolower(trim($params['entity']->email)));

    // Default icon size
    $size = '150x150';

    // Use configured size if possible
    $config = elgg_get_icon_sizes('user');
    $key = $params['size'];
    if (isset($config[$key])) {
        $size = $config[$key]['w'] . 'x' . $config[$key]['h'];
    }

    // Produce URL used to retrieve icon
    return "http://www.gravatar.com/avatar/$hash?s=$size";
}
```

entity:<icon_type>:url, <entity_type> Permet de filtrer les URLs pour les types d'icônes personnalisées, voir `entity:icon:url, <entity_type>`

entity:icon:file, <entity_type> Déclenché par `ElggEntity::getIcon()` et permet aux plugins de fournir un objet alternatif `ElggIcon` qui pointe vers un emplacement personnalisé de l'icône dans le répertoire de données. Le gestionnaire doit renvoyer une instance de `ElggIcon` ou une exception sera lancée.

entity:<icon_type>:file, <entity_type> Permet de filtrer l'objet de fichier d'icône pour les types d'icônes personnalisés, voir `entity:icon:file, <entity_type>`

entity :<icon_type> :prepare, <entity_type> Déclenché par les méthodes `ElggEntity::saveIcon*()` et peut être utilisé pour préparer une image à partir d'un fichier téléchargé/lié. Ce hook peut être utilisé pour faire pivoter l'image avant qu'elle ne soit redimensionnée/rognée, ou il peut être utilisé pour extraire une image cadre (frame) si le fichier téléchargé est une vidéo. Le gestionnaire doit renvoyer une instance de `ElggFile` avec un `simpletype` qui se résout en *image*. La valeur `$return` transmise au hook est une instance de `ElggFile` qui pointe vers une copie temporaire du fichier téléchargé/lié.

La tableau `$params` contient :

- `entity` - entité propriétaire des icônes
- `file` - le fichier d'entrée original avant qu'il n'ait été modifié par d'autres hooks

entity :<icon_type> :save, <entity_type> Déclenché par les méthodes `ElggEntity::saveIcon*()` et peut être utilisé pour appliquer une logique de manipulation d'image personnalisée aux icônes de redimensionnement/recadrage. Le gestionnaire doit renvoyer `true` pour empêcher les API de base de redimensionner/rognier les icônes. Le tableau `$params` contient :

- `entity` - entité propriétaire des icônes
- `file` - objet `ElggFile` qui pointe vers le fichier image à utiliser comme source pour les icônes
- `x1, y1, x2, y2` - coordonnées de découpe

entity :<icon_type> :saved, <entity_type> Déclenché par les méthodes `ElggEntity::saveIcon*()` une fois que des icônes ont été créées. Ce hook peut être utilisé par les plugins pour créer des éléments pour la rivière, mettre à jour les coordonnées de recadrage pour les types d'icônes personnalisés, etc. Le gestionnaire peut accéder aux icônes créées à l'aide de `ElggEntity::getIcon()`. Le tableau `$params` contient :

- `entity` - entité propriétaire des icônes
- `x1, y1, x2, y2` - coordonnées de découpe

entity :<icon_type> :delete, <entity_type> Déclenché par la méthode `ElggEntity::deleteIcon()` et peut être utilisé pour les opérations de nettoyage. Ce hook est déclenché avant que les icônes ne soient supprimées. Le gestionnaire peut renvoyer `false` pour empêcher la suppression d'icônes. Le tableau `$params` contient :

- `entity` - entité propriétaire des icônes

entity :url, <entity_type> Renvoie l'URL de l'entité `$params['entity']`. Remarque : En général, il est préférable de remplacer la méthode `getUrl()` d'`ElggEntity`. Ce hook doit être utilisé lorsqu'il n'est pas possible de créer une sous-classe (par ex. si vous voulez étendre un plugin groupé sans remplacer de nombreuses vues).

to :object, <entity_type|metadata|annotation|relationship|river_item> Convertit l'entité `$params['entity']` en objet `StdClass`. Ceci est utilisé principalement pour exporter des propriétés d'entité vers des formats de données portables comme JSON et XML.

extender :url, <annotation|metadata> Retournez l'URL de l'annotation ou de la métadonnée `$params['extender']`.

file :icon :url, override Remplacez une URL d'icône de fichier.

is_member, group Renvoie un booléen qui indique si l'utilisateur `$params['user']` est un membre du groupe `$params['group']`.

usersetting, plugin Filtrer les paramètres utilisateur pour les plugins. `$params` contient :

- `user` - Une instance de `ElggUser`
- `plugin` - Une instance de `ElggPlugin`
- `plugin_id` - L'ID du plugin
- `name` - Le nom du paramètre
- `value` - La valeur à définir

setting, plugin Filtre les paramètres du plugin. `$params` contient :

- `plugin` - Une instance de `ElggPlugin`
- `plugin_id` - L'ID du plugin
- `name` - Le nom du paramètre
- `value` - La valeur à définir

relationship :url, <relationship_name> Filtre l'URL pour l'objet de relation `$params['relationship']`.

profile :fields, group Filtrer un tableau de champs de profil. Le résultat doit être renvoyé sous forme de tableau dans le format `name => nom de la vue`. Par exemple :

```
array(  
    'about' => 'longtext'  
);
```

profile :fields, profile Filtrer un tableau de champs de profil. Le résultat doit être renvoyé sous forme de tableau dans le format `name => nom de la vue`. Par exemple :

```
array(  
    'about' => 'longtext'  
);
```

widget_settings, <widget_handler> Déclenché lors de l'enregistrement des paramètres d'un widget `$params['params']` pour le widget `$params['widget']`. Si vous gérez l'enregistrement des paramètres, le gestionnaire doit renvoyer `true` pour empêcher l'exécution du code par défaut.

handlers, widgets Déclenché lorsqu'une liste de widgets disponibles est nécessaire. Les plugins peuvent ajouter ou supprimer conditionnellement des widgets de cette liste ou modifier des attributs de widgets existants comme `context` ou `multiple`.

get_list, default_widgets Filtre une liste de widgets par défaut à ajouter pour les utilisateurs nouvellement enregistrés. La liste est un tableau de tableaux dans le format :

```
array(  
    'event' => $event,  
    'entity_type' => $entity_type,  
    'entity_subtype' => $entity_subtype,  
    'widget_context' => $widget_context  
)
```

public_pages, walled_garden Filtre une liste d'URL (chemins) qui peuvent être vues par les utilisateurs déconnectés dans un mode walled garden. Les gestionnaires doivent renvoyer un tableau de chaînes regex qui permettront l'accès si l'URL correspond. Veuillez noter que les itinéraires publics du système sont passés comme valeur par défaut pour le hok, et que les plugins doivent prendre soin de ne pas remplacer accidentellement ces valeurs.

La tableau `$params` contient :

— `url` - URL de la page testée pour l'accessibilité publique

volatile, metadata Déclenché lors de l'export d'une entité par l'intermédiaire du gestionnaire d'exportation. C'est rare. Cela permet au gestionnaire de traiter toutes les métadonnées volatiles (non persistantes) sur l'entité. Il est préférable d'utiliser le hook `to:object, <type>`.

maintenance :allow, url

Retourne un booléen si l'URL `$params['current_url']` et le chemin `$params['current_path']` est autorisé avec le mode maintenance.

robots.txt, site Filtrez les valeurs robots.txt pour `$params['site']`.

config, amd Filtre la configuration AMD pour la bibliothèque requirejs.

Plugins

Embed

embed_get_items, <active_section>

embed_get_sections, all

embed_get_upload_sections, all

Groupes

profile_buttons, **group** Les boutons de filtres (instances `ElggMenuItem`) à enregistrer dans le menu titre de la page de profil de groupe

tool_options, **group** Filtre une collection d'outils disponibles au sein d'un groupe spécifique :

Le \$return est `\Elgg\Collections\Collection<\Elgg\Groups\Tool>`, une collection d'outils du groupe.

La tableau \$params contient :

— `entity - \ElggGroup`

HTMLawed

allowed_styles, **htmlawed** Filtre le tableau de style HTMLawed autorisé.

config, **htmlawed** Filtrez le tableau HTMLawed \$config.

spec, **htmlawed** Filtre la chaîne HTMLawed \$spec (vide par défaut).

Likes

likes :is_likable, <type> :<subtype> Ceci est appelé pour définir les autorisations par défaut pour déterminer s'il faut afficher/autoriser les mentions J'aime sur une entité de type <type> et de sous-type <subtype>.

Note : La fonction de rappel '`Elgg\Values::getTrue`' est un gestionnaire utile pour ce hook.

Membres

members :list, <page_segment> Pour gérer la page `/members/$page_segment`, enregistrez une fonction de rappel pour ce hook et renvoyez le HTML de la liste.

members :config, **tabs** Ce hook est utilisé pour assembler un tableau d'onglets à transmettre à la vue navigation/tabs pour les pages des membres.

Contenu Signalé

reportedcontent :add, system Déclenché après l'ajout de l'objet de contenu signalé `$params['report']`. Renvoyez `false` pour supprimer le rapport.

reportedcontent :archive, system Déclenché avant l'archivage de l'objet de contenu signalé `$params['report']`. Renvoyez `false` pour empêcher l'archivage.

reportedcontent :delete, system Déclenché avant de supprimer l'objet de contenu signalé `$params['report']`. Renvoyez `false` pour éviter la suppression.

Web Services

rest, init Déclenché par le gestionnaire de web services rest. Les plugins peuvent configurer leurs propres gestionnaires d'authentification, puis renvoyer `true` pour empêcher les gestionnaires par défaut d'être enregistrés.

rest :output, <method_name> Filtrer le résultat (et par la suite la sortie) de la méthode API

3.3.17 Internationalisation

Rendre votre interface utilisateur traduisible dans de nombreuses langues différentes.

Si vous souhaitez contribuer à des traductions pour Elgg, consultez le guide des contributeurs guide.

La langue par défaut est en pour l'anglais. Elgg utilise un système de secours pour les langues :

1. La langue de l'utilisateur
2. La langue du site
3. Anglais

Aperçu

Les traductions sont stockées dans des fichiers PHP dans le répertoire `/languages` de votre plugin. Chaque fichier correspond à une langue. Le format est `/languages/{language-code}.php` où `{language-code}` est le code court ISO 639-1 pour la langue. Par exemple :

```
<?php // mod/example/languages/en.php

return [
    'example:text' => 'Some example text',
];
```

Pour remplacer une traduction existante, incluez-la dans le fichier de traduction de votre plugin et assurez-vous que votre plugin est situé après sur la page Admin > Plugins :

```
<?php // mod/better_example/languages/en.php

return [
    'example:text' => 'Some better text!',
];
```

Note : A moins que vous ne remplaciez les chaînes linguistiques du noyau ou d'un autre plugin, c'est une bonne pratique pour les clefs de traduction de commencer par le nom de votre plugin. Par exemple : `votreplugin:success`, `votreplugin:title`, etc. Cela permet d'éviter les conflits avec d'autres clefs de traduction.

API côté serveur

`elgg_echo($key, $args, $language)`

Sortie de la traduction de la clef dans la langue actuelle.

Exemple :

```
echo elgg_echo('example:text');
```

Elle prend également en charge le remplacement des variables à l'aide de la syntaxe `` vsprintf `` :

```
// 'welcome' => 'Welcome to %S, %s!'
echo elgg_echo('welcome', [
    elgg_get_config('sitename'),
    elgg_get_logged_in_user_entity()->getDisplayName(),
]);
```

Pour forcer la langue qui doit être utilisée pour la traduction, définissez le troisième paramètre :

```
echo elgg_echo('welcome', [], $user->language);
```

Pour tester d'abord si `elgg_echo()` peut trouver une traduction :

```
$key = 'key:that:might:not:exist';
if (!elgg_language_key_exists($key)) {
    $key = 'fallback:key';
}

echo elgg_echo($key);
```

Note : Certaines API permettent de créer des traductions pour de nouvelles clefs. Les traducteurs doivent toujours inclure une traduction anglaise comme solution de repli. Ceci fait de `elgg_language_key_exists($key)` un moyen fiable de prédire si `elgg_echo($key)` va réussir.

API Javascript

`elgg.echo(key, args)`

Cette fonction est équivalente à `elgg_echo` en PHP.

Les traductions côté client sont chargées de façon asynchrone. Assurez-vous que les traductions sont disponibles en exigeant le module AMD `elgg` :

```
define(function(require) {
    var elgg = require("elgg");

    alert(elgg.echo('my_key'));
});
```

Les traductions sont également disponibles après l'événement JavaScript `init`, `system`.

3.3.18 JavaScript

Contenu

- *AMD*
 - *Exécuter un module dans la page en cours*
 - *Définir le Module*
 - *Rendre les modules dépendants d'autres modules*
 - *Passer des paramètres aux modules*
 - *Définir l'URL d'un module*
 - *Utiliser des bibliothèques JS traditionnelles comme modules*
- *Démarrer votre plugin*
- *Modules fournis avec Elgg*
 - *Modules jquery et jquery-ui*
 - *Module elgg*
 - *Module elgg/Ajax*
 - *Module elgg/init*
 - *Module elgg/Plugin*
 - *Module elgg/ready*
 - *Module elgg/spinner*
 - *Module elgg/popup*
 - *Module elgg/widgets*
 - *Module elgg/lightbox*
 - *Module elgg/ckeditor*
 - *Composant d'onglets en ligne*
- *Scripts traditionnels*
- *Hooks*
 - *Enregistrer des gestionnaires de hook*
 - *La fonction de gestion*
 - *Déclencher des hooks personnalisés*
 - *Hooks disponibles*
- *Actifs tierce-partie (assets)*

AMD

Les développeurs doivent utiliser le standard [AMD \(Définition du module asynchrone\)](#) pour écrire du code JavaScript dans Elgg.

Ici, nous allons décrire la création et l'exécution des modules AMD. La documentation RequireJS pour [définir des modules](#) peut également être utile.

Exécuter un module dans la page en cours

Il est facile de dire à Elgg de charger un module existant dans la page en cours :

```
<?php
elgg_require_js("myplugin/say_hello");
```

Du côté client, cela chargera le module de manière asynchrone, chargera toutes les dépendances, et exécutera la fonction de définition du module si elle existe.

Définir le Module

Ici, nous définissons un module de base qui modifie la page, en passant une « fonction de définition » à `define()` :

```
// in views/default/myplugin/say_hello.js

define(function(require) {
    var elgg = require("elgg");
    var $ = require("jquery");

    $('body').append(elgg.echo('hello_world'));
});
```

Le nom du module est déterminé par le nom de vue, qui est ici `monplugin/say_hello.js`. Nous retirons l'extension `.js`, pour conserver `monplugin/say_hello`.

Avertissement : La fonction de définition **doit** avoir un argument nommé `require`.

Rendre les modules dépendants d'autres modules

Ci-dessous nous refactorisons un peu de sorte que le module dépende d'un nouveau module `monplugin/hello` pour fournir la salutation :

```
// in views/default/myplugin/hello.js

define(function(require) {
    var elgg = require("elgg");

    return elgg.echo('hello_world');
});
```

```
// in views/default/myplugin/say_hello.js

define(function(require) {
    var $ = require("jquery");
    var hello = require("myplugin/hello");

    $('body').append(hello);
});
```

Passer des paramètres aux modules

Les hooks de plugin `elgg.data`

Le module `elgg` fournit un objet `elgg.data` qui est peuplé à partir de deux hooks côté serveur :

- **elgg.data, site** : Ceci filtre un tableau associatif de données spécifiques au site, transmises au client et mises en cache.
- **elgg.data, page** : Ceci filtre un tableau associatif de données non mises en cache, et spécifiques à la page transmise au client.

Passons quelques données au module :

```
<?php

function myplugin_config_site($hook, $type, $value, $params) {
    // this will be cached client-side
    $value['myplugin']['api'] = elgg_get_site_url() . 'myplugin-api';
    $value['myplugin']['key'] = 'none';
    return $value;
}

function myplugin_config_page($hook, $type, $value, $params) {
    $user = elgg_get_logged_in_user_entity();
    if ($user) {
        $value['myplugin']['key'] = $user->myplugin_api_key;
        return $value;
    }
}

elgg_register_plugin_hook_handler('elgg.data', 'site', 'myplugin_config_site');
elgg_register_plugin_hook_handler('elgg.data', 'page', 'myplugin_config_page');
```

```
define(function(require) {
    var elgg = require("elgg");

    var api = elgg.data.myplugin.api;
    var key = elgg.data.myplugin.key; // "none" or a user's key

    // ...
});
```

Note : Dans `elgg.data` les données de la page remplacent les données du site. Notez également que `json_encode()` est utilisé pour copier les données côté client, de sorte que les données doivent être encodables en JSON.

Créer un module de configuration

Vous pouvez utiliser un module PHP pour transmettre des valeurs à partir du serveur. Pour créer le module `monplugin/settings`, créez le fichier de la vue `views/default/monplugin/settings.js.php` (notez la double extension `.js.php`).

```
<?php

// this will be cached client-side
$settings = [
    'api' => elgg_get_site_url() . 'myplugin-api',
    'key' => null,
];
?>
define(<?php echo json_encode($settings); ?>);
```

Vous devez également enregistrer manuellement la vue en tant que ressource externe :

```
<?php
// note the view name does not include ".php"
elgg_register_simplecache_view('myplugin/settings.js');
```

Note : La vue PHP est mise en cache, vous devez donc traiter la sortie comme statique (la même pour tous les utilisateurs) et éviter toute logique spécifique à la session.

Définir l'URL d'un module

Vous pouvez avoir un script AMD en dehors de vos vues que vous souhaitez mettre à disposition en tant que module.

La meilleure façon d'y parvenir est de configurer le chemin d'accès au fichier à l'aide de la section `views` du fichier `elgg-plugin.php` dans la racine de votre plugin :

```
<?php // elgg-plugin.php
return [
    'views' => [
        'default' => [
            'underscore.js' => 'vendor/bower-asset/underscore/underscore.min.js',
        ],
    ],
];
```

Si vous avez copié le script directement dans votre plugin au lieu de le gérer avec Composer, vous pouvez utiliser quelque chose comme ceci à la place :

```
<?php // elgg-plugin.php
return [
    'views' => [
        'default' => [
            'underscore.js' => __DIR__ . '/bower_components/underscore/underscore.
↪min.js',
        ],
    ],
];
```

Voilà ! Elgg chargera maintenant ce fichier chaque fois que le module « underscore » est requis.

Utiliser des bibliothèques JS traditionnelles comme modules

Il est possible de prendre en charge les bibliothèques JavaScript qui ne se déclarent pas comme modules AMD (c'est-à-dire qu'elles déclarent plutôt des variables globales) si vous les intégrez via des shims en définissant `exports` et `deps` dans `elgg_define_js` :

```
// set the path, define its dependencies, and what value it returns
elgg_define_js('jquery.form', [
    'deps' => ['jquery'],
    'exports' => 'jQuery.fn.ajaxForm',
]);
```

Lorsque cela est demandé côté client :

1. Le module jQuery est chargé, car il est marqué comme une dépendance.
2. `https://elgg.example.org/cache/125235034/views/default/jquery.form.js` est chargé et exécuté.
3. La valeur de `window.jQuery.fn.ajaxForm` est renvoyée par le module.

Avertissement : Les appels à `elgg_define_js()` doivent être faits dans un gestionnaire d'événement `init`, `system`.

Quelques points à noter

1. N'utilisez plus `elgg.provide()` ni d'autres moyens pour attacher du code à `elgg` ou à d'autres objets globaux. Utilisez des modules.
2. Renvoyez la valeur du module au lieu de l'ajouter à une variable globale.
3. Les fichiers statiques (.js, .css, etc.) sont automatiquement minifiés et mis en cache par le système simplecache de Elgg.
4. La configuration est également mise en cache dans le simplecache et ne doit pas s'appuyer sur des valeurs spécifiques à l'utilisateur comme `get_current_language()`.

Démarrer votre plugin

Pour ajouter des fonctionnalités à chaque page ou vous assurer que vos gestionnaires de hooks sont enregistrés assez tôt, vous pouvez créer un module de démarrage pour votre plugin, avec le nom `boot/<plugin_id>`.

```
// in views/default/boot/example.js

define(function(require) {
    var elgg = require("elgg");
    var Plugin = require("elgg/Plugin");

    // plugin logic
    function my_init() { ... }

    return new Plugin({
        // executed in order of plugin priority
        init: function () {
            elgg.register_hook_handler("init", "system", my_init, 400);
        }
    });
});
```

Lorsque votre plugin est actif, ce module sera automatiquement chargé sur chaque page. D'autres modules peuvent dépendre de `elgg/init` pour s'assurer que tous les modules de démarrage sont chargés.

Chaque module de démarrage **doit** renvoyer une instance de `elgg/Plugin`. Le constructeur doit recevoir un objet avec une fonction dans la clef `init`. La fonction `init` sera appelée selon l'ordre du plugin dans la zone d'administration d'Elgg.

Note : Bien que cela ne soit pas strictement nécessaire, vous pouvez utiliser l'événement `init`, `system` pour contrôler quand votre code d'initialisation s'exécute par rapport à d'autres modules.

Avvertimento : Un module de démarrage (boot) **ne peut pas** dépendre des modules `elgg/init` ou `elgg/ready`.

Modules fournis avec Elgg

Modules jquery et jquery-ui

Vous devez faire dépendre de ces modules pour utiliser les méthodes `$` ou `$.ui`. À l'avenir, Elgg pourrait cesser de les charger par défaut.

Module elgg

`elgg.echo()`

Traduisez le texte de l'interface

```
elgg.echo('example:text', ['arg1']);
```

`elgg.system_message()`

Affichez un message d'état à l'utilisateur.

```
elgg.system_message(elgg.echo('success'));
```

`elgg.register_error()`

Affiche un message d'erreur à l'utilisateur.

```
elgg.register_error(elgg.echo('error'));
```

`elgg.normalize_url()`

Normalise une URL par rapport à la racine elgg :

```
// "http://localhost/elgg/blog"
elgg.normalize_url('/blog');
```

`elgg.forward()`

Redirige vers une nouvelle page.

```
elgg.forward('/blog');
```

Cette fonction normalise automatiquement l'URL.

`elgg.parse_url()`

Analyse et découpe une URL en ses composants :

```
// returns {
//   fragment: "fragment",
//   host: "community.elgg.org",
//   path: "/file.php",
//   query: "arg=val"
// }
elgg.parse_url('http://community.elgg.org/file.php?arg=val#fragment');
```

`elgg.get_page_owner_guid()`

Récupère le GUID du propriétaire de la page actuelle.

`elgg.register_hook_handler()`

Enregistrez un gestionnaire de hook avec le système d'événements. Pour de meilleurs résultats, faites-le dans un module de démarrage plugin.

```
// boot module: /views/default/boot/example.js
define(function (require) {
    var elgg = require('elgg');
    var Plugin = require('elgg/Plugin');

    elgg.register_hook_handler('foo', 'bar', function () { ... });

    return new Plugin();
});
```

`elgg.trigger_hook()`

Émet un hook d'événement dans le système d'événements. Pour de meilleurs résultats faites-le dépendre du module `elgg/init`.

```
// old
value = elgg.trigger_hook('my_plugin:filter', 'value', {}, value);

define(function (require) {
    require('elgg/init');
    var elgg = require('elgg');

    value = elgg.trigger_hook('my_plugin:filter', 'value', {}, value);
});
```

`elgg.security.refreshToken()`

Forcer le rafraîchissement de tous les jetons XSRF sur la page.

Par défaut, ceci est automatiquement appelé toutes les 5 minutes.

L'utilisateur sera averti si sa session a expiré.

`elgg.security.addToken()`

Ajoute un jeton de sécurité à un objet, une URL ou une chaîne de requête :

```
// returns {
//   __elgg_token: "1468dc44c5b437f34423e2d55acfd87",
//   __elgg_ts: 1328143779,
//   other: "data"
// }
elgg.security.addToken({'other': 'data'});

// returns: "action/add?__elgg_ts=1328144079&__elgg_
↪ token=55fd9c2d7f5075d11e722358afd5fde2"
elgg.security.addToken("action/add");

// returns "?arg=val&__elgg_ts=1328144079&__elgg_token=55fd9c2d7f5075d11e722358afd5fde2"
elgg.security.addToken("?arg=val");
```

`elgg.get_logged_in_user_entity()`

Renvoie l'utilisateur connecté sous le forme d'un objet JS ElggUser.

`elgg.get_logged_in_user_guid()`

Renvoie le GUID de l'utilisateur connecté.

`elgg.is_logged_in()`

True si l'utilisateur est connecté.

`elgg.is_admin_logged_in()`

True si l'utilisateur est connecté et est administrateur.

`elgg.config.get_language()`

Récupère la langue de la page actuelle.

Un certain nombre de valeurs de configuration sont définies dans l'objet `elgg` :

```
// The root of the website.
elgg.config.wwwroot;
// The default site language.
elgg.config.language;
// The current page's viewtype
elgg.config.viewtype;
// The Elgg version (YYYYMMDDXX).
elgg.config.version;
// The Elgg release (X.Y.Z).
elgg.config.release;
```

Module `elgg/Ajax`

Voir la page [Ajax](#) pour plus de détails.

Module `elgg/init`

`elgg/init` charge et initialise tous les modules de démarrage dans l'ordre de priorité et déclenche le hook `[init, system]`.

Ajoutez ce module aux exigences (require) de ce module pour vous assurer que tous les plugins sont prêts.

Module `elgg/Plugin`

Utilisé pour créer un module de démarrage *boot*.

Module elgg/ready

elgg/ready charge et initialise tous les modules de démarrage des plugins dans l'ordre de priorité.

Ajoutez ce module aux exigences (require) de ce module pour vous assurer que tous les plugins sont prêts.

Module elgg/spinner

Le module elgg/spinner peut être utilisé pour créer un indicateur de chargement Ajax positionné en haut de la fenêtre.

```
define(function (require) {
    var spinner = require('elgg/spinner');

    elgg.action('friend/add', {
        beforeSend: spinner.start,
        complete: spinner.stop,
        success: function (json) {
            // ...
        }
    });
});
```

Note : Le module elgg/Ajax utilise le spinner par défaut.

Module elgg/popup

Le module elgg/popup peut être utilisé pour afficher une superposition (overlay) placée relativement à son ancre (déclencheur).

Le module elgg/popup est chargé par défaut, et lier un module popup à une ancre est aussi simple que d'ajouter l'attribut rel="popup" et de définir le module cible avec un attribut href (ou data-href). Le positionnement du module popup peut être défini avec l'attribut data-position de l'élément déclencheur.

```
echo elgg_format_element('div', [
    'class' => 'elgg-module-popup hidden',
    'id' => 'popup-module',
], 'Popup module content');

// Simple anchor
echo elgg_view('output/url', [
    'href' => '#popup-module',
    'text' => 'Show popup',
    'rel' => 'popup',
]);

// Button with custom positioning of the popup
echo elgg_format_element('button', [
    'rel' => 'popup',
    'class' => 'elgg-button elgg-button-submit',
```

(suite sur la page suivante)

(suite de la page précédente)

```

    'text' => 'Show popup',
    'data-href' => '#popup-module',
    'data-position' => json_encode([
        'my' => 'center bottom',
        'at' => 'center top',
    ]),
  ]),
});

```

Le module `elgg/popup` vous permet de créer des éléments d'interface utilisateur/UX plus complexes. Vous pouvez ouvrir et fermer les modules contextuels via le code :

```

define(function(require) {
    var $ = require('jquery');
    $(document).on('click', '.elgg-button-popup', function(e) {

        e.preventDefault();

        var $trigger = $(this);
        var $target = $('#my-target');
        var $close = $target.find('.close');

        require(['elgg/popup'], function(popup) {
            popup.open($trigger, $target, {
                'collision': 'fit none'
            });

            $close.on('click', popup.close);
        });
    });
});

```

Vous pouvez utiliser le hook plugin `getOptions`, `ui.popup` pour manipuler la position du popup avant qu'il ne soit ouvert. Vous pouvez utiliser les événements jQuery `open` et `close` pour manipuler le module popup après son ouverture ou sa fermeture.

```

define(function(require) {

    var elgg = require('elgg');
    var $ = require('jquery');

    $('#my-target').on('open', function() {
        var $module = $(this);
        var $trigger = $module.data('trigger');

        elgg.ajax('ajax/view/my_module', {
            beforeSend: function() {
                $trigger.hide();
                $module.html('').addClass('elgg-ajax-loader');
            },
            success: function(output) {
                $module.removeClass('elgg-ajax-loader').html(output);
            }
        });
    });
});

```

(suite sur la page suivante)

(suite de la page précédente)

```
});
}).on('close', function() {
    var $trigger = $(this).data('trigger');
    $trigger.show();
});
});
```

Les modules popups ouverts contiennent toujours les données suivantes auxquelles on peut accéder via `$.data()` :

- `trigger` - élément jQuery utilisé pour déclencher l'ouverture du module popup
- `position` - Objet définissant la position du module popup qui a été passé à `$.position()`

Par défaut, l'élément `target` sera ajouté à `$('body')`, modifiant ainsi la hiérarchie DOM. Si vous devez conserver la position DOM du module popup, vous pouvez ajouter la classe `.elgg-popup-inline` à votre déclencheur.

Module elgg/widgets

Plugins qui chargent une disposition widget via Ajax devraient s'initialiser via ce module :

```
require(['elgg/widgets'], function (widgets) {
    widgets.init();
});
```

Module elgg/lightbox

Elgg est distribué avec la bibliothèque jQuery Colorbox. Veuillez consulter <http://www.jacklmoore.com/colorbox> pour plus d'informations sur les options de cette lightbox.

Utilisez les classes suivantes pour lier vos éléments d'ancrage à des boîtes surgissantes lightbox :

- `elgg-lightbox` - charge une ressource HTML
- `elgg-lightbox-photo` - charge une ressource de type image (devrait être utilisé pour éviter d'afficher les octets d'image bruts au lieu de la balise `img`)
- `elgg-lightbox-inline` - affiche un élément HTML en ligne dans une boîte surgissante lightbox
- `elgg-lightbox-iframe` - charge une ressource dans une `iframe`

Vous pouvez appliquer des options de colorbox à un élément individuel `elgg-lightbox` en définissant l'attribut `data-colorbox-opts` avec un objet de paramètres JSON.

```
echo elgg_view('output/url', [
    'text' => 'Open lightbox',
    'href' => 'ajax/view/my_view',
    'class' => 'elgg-lightbox',
    'data-colorbox-opts' => json_encode([
        'width' => '300px',
    ])
]);
```

Utilisez le hook plugin `"getOptions"`, `"ui.lightbox"` pour filtrer les options passées à `$.colorbox()` chaque fois qu'une lightbox est ouverte. Notez que le gestionnaire de hook doit dépendre du module AMD `elgg/init`.

Le module AMD `elgg/lightbox` devrait être utilisé pour ouvrir et fermer la lightbox de manière programmatique :

```
define(function(require) {
    var lightbox = require('elgg/lightbox');
```

(suite sur la page suivante)

(suite de la page précédente)

```

var spinner = require('elgg/spinner');

lightbox.open({
  html: '<p>Hello world!</p>',
  onClose: function() {
    lightbox.open({
      onLoad: spinner.start,
      onComplete: spinner.stop,
      photo: true,
      href: 'https://elgg.org/cache/1457904417/default/community_theme/graphics/
↪logo.png',
    });
  }
});
});

```

Pour prendre en charge des ensembles de galerie (via l'attribut `rel`), vous devez lier colorbox directement à un sélecteur spécifique (notez que cela ignorera `data-colorbox-opts` sur tous les éléments d'un ensemble) :

```

require(['elgg/lightbox'], function(lightbox) {
  var options = {
    photo: true,
    width: 500
  };
  lightbox.bind('a[rel="my-gallery"]', options, false); // 3rd attribute ensures
↪binding is done without proxies
});

```

Vous pouvez également redimensionner la lightbox via le code si nécessaire :

```

define(function(require) {
  var lightbox = require('elgg/lightbox');

  lightbox.resize({
    width: '300px'
  });
});

```

Module elgg/ckeditor

Ce module peut être utilisé pour ajouter un éditeur WYSIWYG à un textarea (nécessite l'activation du plugin ckeditor). Notez que le WYSIWYG sera automatiquement attaché à toutes les instances de `.elgg-input-longtext`.

```

require(['elgg/ckeditor'], function (elggCKEditor) {
  elggCKEditor.bind('#my-text-area');

  // Toggle CKEditor
  elggCKEditor.toggle('#my-text-area');

  // Focus on CKEditor input
  elggCKEditor.focus('#my-text-area');

```

(suite sur la page suivante)

(suite de la page précédente)

```
// or
$('#my-text-area').trigger('focus');

// Reset CKEditor input
elggCKEditor.reset('#my-text-area');
// or
$('#my-text-area').trigger('reset');

});
```

Composant d'onglets en ligne

Le composant d'onglets en ligne déclenche un événement `open` chaque fois qu'un onglet est ouvert et, en cas d'onglets ajax, une fois le chargement terminé :

```
// Add custom animation to tab content
require(['jquery', 'elgg/ready'], function($) {
    $(document).on('open', '.theme-sandbox-tab-callback', function() {
        $(this).find('a').text('Clicked!');
        $(this).data('target').hide().show('slide', {
            duration: 2000,
            direction: 'right',
            complete: function() {
                alert('Thank you for clicking. We hope you enjoyed the_
↪show!');
                $(this).css('display', ''); // .show() adds display_
↪property
            }
        });
    });
});
```

Scripts traditionnels

Bien que nous recommandions fortement d'utiliser des modules AMD et qu'il n'existe pas d'API Elgg pour le chargement des scripts, vous pouvez enregistrer des scripts dans un gestionnaire de hook pour ajouter des éléments aux liens de l'entête ;

```
elgg_register_plugin_hook_handler('head', 'page', $callback);
```

Hooks

Le moteur JS dispose d'un système de hooks similaire aux hooks de plugin du moteur PHP : des hooks sont déclenchés et des plugins peuvent enregistrer des fonctions pour réagir ou pour modifier l'information. Il n'y a pas de concept d'événement Elgg dans le moteur JS ; tout dans le moteur JS est implémenté sous forme de hook.

Enregistrer des gestionnaires de hook

Les fonctions de gestionnaire sont enregistrées à l'aide de `elgg.register_hook_handler()`. Plusieurs gestionnaires peuvent être enregistrés pour le même hook.

L'exemple suivant enregistre la fonction `handleFoo` pour le hook `foo`, `bar`.

```
define(function (require) {
    var elgg = require('elgg');
    var Plugin = require('elgg/Plugin');

    function handleFoo(hook, type, params, value) {
        // do something
    }

    elgg.register_hook_handler('foo', 'bar', handleFoo);

    return new Plugin();
});
```

La fonction de gestion

Le gestionnaire va recevoir 4 arguments :

- **hook** - Le nom du hook
- **type** - Le type de hook
- **params** - Un objet ou un ensemble de paramètres propres au hook
- **value** - La valeur actuelle

La valeur `value` passera par chaque hook. Selon le hook, les fonctions de rappel peuvent simplement réagir ou modifier les données.

Déclencher des hooks personnalisés

Les plugins peuvent déclencher leurs propres hooks :

```
define(function(require) {
    require('elgg/init');
    var elgg = require('elgg');

    elgg.trigger_hook('name', 'type', {params}, "value");
});
```

Note : Soyez conscient du timing. Si vous ne faites pas dépendre de `elgg/init`, d'autres plugins n'auront peut-être pas eu la possibilité d'enregistrer leurs gestionnaires.

Hooks disponibles

- init, system** Les plugins devraient enregistrer leurs fonctions d'initialisation via ce hook. Il est déclenché après le chargement des JS de Elgg et l'initialisation de tous les modules de démarrage des plugins. Faites-le dépendre du module `elgg/init` pour être sûr que l'initialisation est bien terminée.
- ready, system** Ce hook est déclenché lorsque le système a complètement démarré (après `init`). Faites-le dépendre du module `elgg/ready` pour être sûr que le démarrage est bien terminé.
- getOptions, ui.popup** Ce hook est déclenché pour les affichages surgissants ("`rel`"="`popup`") et permet des options de placement personnalisées.
- getOptions, ui.lightbox** Ce hook peut être utilisé pour filtrer les options transmises à `$.colorbox()`
- config, ckeditor** Cela filtre l'objet de configuration CKEditor. Enregistrez un gestionnaire pour ce hook dans un module de démarrage du plugin. Les valeurs par défaut sont visibles dans le module `elgg/ckeditor/config`.
- prepare, ckeditor** Ce crochet peut être utilisé pour modifier globalement CKEDITOR. Vous pouvez utiliser ce hook pour enregistrer de nouveaux plugins CKEditor et ajouter des liaisons d'événements (event bindings).
- ajax_request_data, *** Ceci filtre les demandes de données envoyées par le module `elgg/Ajax`. Voir `ajax`` pour plus de détails. Le hook doit vérifier si les données sont un objet simple ou une instance de `FormData` pour récupérer les valeurs à l'aide de la bonne API.
- ajax_response_data, *** Ceci filtre les données de réponse retournées aux utilisateurs du module `elgg/Ajax`. Voir `Ajax` pour plus de détails.
- insert, editor** Ce hook est déclenché par le plugin `embed` et peut être utilisé pour filtrer le contenu avant qu'il ne soit inséré dans le textarea. Ce hook peut également être utilisé par les éditeurs WYSIWYG pour insérer du contenu à l'aide de leur propre API (dans ce cas, le gestionnaire devrait renvoyer `false`). Voir le plugin `ckeditor` pour un exemple.

Actifs tierce-partie (assets)

Nous vous recommandons de gérer des scripts et des styles tiers avec Composer. Le `composer.json` d'Elgg est configuré pour installer des dépendances à partir des référentiels de packages **Bower** ou **Yarn** à l'aide de l'outil de ligne de commande `Composer`. La configuration de base installe les actifs à partir de **Asset Packagist** (<https://asset-packagist.org>) (un référentiel géré par la communauté Yii).

Alternativement, vous pouvez installer `fxp/composer-asset-plugin` globalement pour obtenir les mêmes résultats, mais l'installation et la mise à jour prennent beaucoup plus de temps.

Par exemple, pour inclure jQuery, vous pouvez exécuter les commandes composer suivantes :

```
composer require bower-asset/jquery:~2.0
```

Si vous utilisez un projet de démarrage (`starter-project`) ou que vous utilisez Elgg sous forme de dépendance d'un projet composer personnalisé, mettez à jour votre `composer.json` avec la configuration suivante :

```
{
  "repositories": [
    {
      "type": "composer",
      "url": "https://asset-packagist.org"
    }
  ],
  "config": {
    "fxp-asset": {
      "enabled": false
    }
  }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

    },
  },
}
```

Vous pouvez trouver des informations supplémentaires sur le site de [Asset Packagist](#).

3.3.19 Menus

Elgg propose des fonctions pratiques pour créer des menus sur l'ensemble du site.

Chaque menu a besoin d'un nom, tout comme chaque élément de menu. Ceux-ci sont nécessaires afin de faciliter leur remplacement ou leur modification, ainsi que pour fournir des hooks pour le thème.

Contenu

- *Utilisation simple*
- *Menu admin*
- *Utilisation avancée*
- *Créer un nouveau menu*
- *Sous-menus déroulants*
- *Création de thèmes*
- *Basculer des éléments de menu*
- *JavaScript*

Utilisation simple

Les fonctionnalités de base peuvent être réalisées grâce à ces deux fonctions :

- `elgg_register_menu_item()` pour ajouter un élément à un menu
- `elgg_unregister_menu_item()` pour supprimer un élément d'un menu

Vous voudrez normalement les appeler à partir de la fonction `init` de votre plugin.

Exemples

```
// Add a new menu item to the site main menu
elgg_register_menu_item('site', array(
    'name' => 'itemname',
    'text' => 'This is text of the item',
    'href' => '/item/url',
));
```

```
// Remove the "Elgg" logo from the topbar menu
elgg_unregister_menu_item('topbar', 'elgg_logo');
```

Menu admin

Vous pouvez également inscrire des éléments du menu page au menu de l'interface d'administration. Lorsque vous inscrivez des éléments au menu d'administration, vous pouvez définir le contexte des éléments de menu sur `admin` afin que les éléments du menu ne s'affichent que dans le contexte `admin`. Il existe 3 sections par défaut auxquelles ajouter vos éléments de menu.

- `administer` pour les tâches quotidiennes, la gestion des utilisateurs et d'autres tâches concrètes
- `configure` pour des paramètres, des éléments de configuration et des utilitaires qui configurent diverses choses
- `information` pour des statistiques, une vue d'ensemble d'informations ou de statuts

Utilisation avancée

Vous pouvez obtenir plus de contrôle sur les menus en utilisant les *hooks plugin* et les méthodes publiques fournies par la classe `ElggMenuItem`.

Il existe trois hooks qui peuvent être utilisés pour modifier un menu :

- `'parameters', 'menu:'` pour ajouter ou modifier les paramètres utilisés pour construire le menu (par exemple l'ordre de tri)
- `'register', 'menu:'` pour ajouter ou modifier des éléments (en particulier dans les menus dynamiques)
- `'prepare', 'menu:'` pour modifier la structure du menu avant qu'il ne soit affiché

Lorsque vous enregistrez un gestionnaire de hook, remplacez la partie `<menu name>` par le nom interne du menu.

Le troisième paramètre passé au gestionnaire de menu contient tous les éléments de menu qui ont été enregistrés jusqu'à présent par le noyau Elgg et d'autres plugins activés. Dans le gestionnaire, nous pouvons faire une boucle à travers les éléments de menu et utiliser les méthodes de classe pour interagir avec les propriétés de l'élément de menu.

Exemples

Exemple 1 : Modifier l'URL de l'élément de menu appelé `albums` dans le menu `owner_block` :

```
/**
 * Initialize the plugin
 */
function my_plugin_init() {
    // Register a plugin hook handler for the owner_block menu
    elgg_register_plugin_hook_handler('register', 'menu:owner_block', 'my_owner_
↪block_menu_handler');
}

/**
 * Change the URL of the "Albums" menu item in the owner_block menu
 */
function my_owner_block_menu_handler($hook, $type, $items, $params) {
    $owner = $params['entity'];

    // Owner can be either user or a group, so we
    // need to take both URLs into consideration:
    switch ($owner->getType()) {
        case 'user':
            $url = "album/owner/{$owner->guid}";
            break;
        case 'group':
```

(suite sur la page suivante)

(suite de la page précédente)

```

        $url = "album/group/{$owner->guid}";
        break;
    }

    if ($items->has('albums')) {
        $items->get('albums')->setURL($url);
    }

    return $items;
}

```

Exemple 2 : Modifier le menu entity pour les objets ElggBlog

- Supprimer l'icône du pouce
- Modifier le texte Modifier en une icône personnalisée

```

/**
 * Initialize the plugin
 */
function my_plugin_init() {
    // Register a plugin hook handler for the entity menu
    elgg_register_plugin_hook_handler('register', 'menu:entity', 'my_entity_menu_
    ↪handler');
}

/**
 * Customize the entity menu for ElggBlog objects
 */
function my_entity_menu_handler($hook, $type, $items, $params) {
    // The entity can be found from the $params parameter
    $entity = $params['entity'];

    // We want to modify only the ElggBlog objects, so we
    // return immediately if the entity is something else
    if (!$entity instanceof ElggBlog) {
        return $menu;
    }

    $items->remove('likes');

    if ($items->has('edit')) {
        $items->get('edit')->setText('Modify');
        $items->get('edit')->icon = 'pencil';
    }

    return $items;
}

```

Créer un nouveau menu

Elgg fournit plusieurs menus différents par défaut. Vous pouvez cependant avoir besoin de certains éléments de menu qui ne figurent pas dans l'un des menus existants. Si tel est le cas, vous pouvez créer votre propre menu avec la fonction `elgg_view_menu()`. Vous devez appeler la fonction à partir de la vue dans laquelle vous souhaitez afficher le menu.

Exemple : Affichez un menu appelé « mon_menu » qui affiche ses éléments de menu dans l'ordre alphabétique :

```
// in a resource view
echo elgg_view_menu('my_menu', array('sort_by' => 'text'));
```

Vous pouvez maintenant ajouter de nouveaux éléments au menu comme ceci :

```
// in plugin init
elgg_register_menu_item('my_menu', array(
    'name' => 'my_page',
    'href' => 'path/to/my_page',
    'text' => elgg_echo('my_plugin:my_page'),
));
```

En outre, il est maintenant possible de modifier le menu en utilisant les hooks 'register', 'menu:mon_menu' et 'prepare', 'menu:mon_menu'.

Sous-menus déroulants

Les sous-menus peuvent être configurés à l'aide de l'option `child_menu` sur l'élément parent.

Le tableau d'options `child_menu` accepte le paramètre `display`, qui peut être utilisé pour définir le menu enfant à ouvrir sous forme de dropdown (liste déroulante) ou être affiché via toggle (bascule). Toutes les autres paires de clefs/valeurs seront transmises en tant qu'attributs à l'élément `ul`.

```
// Register a parent menu item that has a dropdown submenu
elgg_register_menu_item('my_menu', array(
    'name' => 'parent_item',
    'href' => '#',
    'text' => 'Show dropdown menu',
    'child_menu' => [
        'display' => 'dropdown',
        'class' => 'elgg-additional-child-menu-class',
        'data-position' => json_encode([
            'at' => 'right bottom',
            'my' => 'right top',
            'collision' => 'fit fit',
        ]),
        'data-foo' => 'bar',
        'id' => 'dropdown-menu-id',
    ],
));

// Register a parent menu item that has a hidden submenu toggled when item is clicked
elgg_register_menu_item('my_menu', array(
    'name' => 'parent_item',
    'href' => '#',
    'text' => 'Show submenu',
```

(suite sur la page suivante)

(suite de la page précédente)

```

        'child_menu' => [
            'display' => 'dropdown',
            'class' => 'elgg-additional-submenu-class',
            'data-toggle-duration' => 'medium',
            'data-foo' => 'bar2',
            'id' => 'submenu-id',
        ],
    ));

```

Création de thèmes

Le nom du menu, les noms de sections et les noms des éléments sont tous intégrés dans le HTML sous forme de classes CSS (normalisées pour ne contenir que des traits d'union, plutôt que des soulignements ou des points). Cela augmente légèrement la taille du balisage, mais procure aux intérateurs un haut degré de contrôle et de flexibilité lors de la mise en forme du site.

Exemple : Voici la sortie du menu `foo` avec les sections `alt` et `default` contenant respectivement les éléments `baz` et `bar`.

```

<ul class="elgg-menu elgg-menu-foo elgg-menu-foo-alt">
  <li class="elgg-menu-item elgg-menu-item-baz"></li>
</ul>
<ul class="elgg-menu elgg-menu-foo elgg-menu-foo-default">
  <li class="elgg-menu-item elgg-menu-item-bar"></li>
</ul>

```

Basculer des éléments de menu

Il y a des situations où vous souhaitez basculer entre des éléments de menu qui sont des actions contraires l'une de l'autre, et les ajaxifier. Par exemple, comme ajouter/retirer un contact, bannir/réintégrer, etc. Elgg dispose d'une prise en charge intégrée pour ce genre d'actions. Lorsque vous enregistrez un élément de menu, vous pouvez fournir le nom de l'élément de menu (dans le même menu) qui doit être basculé. Un appel ajax sera fait en utilisant le href de l'élément de menu.

```

elgg_register_menu_item('my_menu', [
    'name' => 'like',
    'data-toggle' => 'unlike',
    'href' => 'action/like',
    'text' => elgg_echo('like'),
]);

elgg_register_menu_item('my_menu', [
    'name' => 'unlike',
    'data-toggle' => 'like',
    'href' => 'action/unlike',
    'text' => elgg_echo('unlike'),
]);

```

Note : Les éléments de menu sont basculés de manière optimiste. Cela signifie que les éléments de menu sont basculés

avant la fin des actions. Si les actions échouent, les éléments de menu seront rétablis.

JavaScript

Il est courant que les éléments de menu s'appuient sur JavaScript. Vous pouvez lier les événements côté client aux éléments du menu en plaçant votre JavaScript dans le module AMD et en définissant ses exigences lors de son inscription.

```
elgg_register_menu_item('my_menu', array(
    'name' => 'hide_on_click',
    'href' => '#',
    'text' => elgg_echo('hide:on:click'),
    'item_class' => '.hide-on-click',
    'deps' => ['navigation/menu/item/hide_on_click'],
));
```

```
// in navigation/menu/item/hide_on_click.js
define(function(require) {
    var $ = require('jquery');

    $(document).on('click', '.hide-on-click', function(e) {
        e.preventDefault();
        $(this).hide();
    });
});
```

3.3.20 Notifications

Il existe deux manières d'envoyer des notifications dans Elgg :

- Notifications instantanées
- Les notifications basées sur les événements sont envoyées à l'aide d'une file d'attente de notifications

Contenu

- *Notifications instantanées*
- *Notifications en file d'attente*
- *Enregistrement d'une nouvelle méthode de notification*
- *Envoyer des notifications à l'aide de votre propre méthode*
- *Abonnements*
- *Pièces jointes par email*

Notifications instantanées

La méthode générique pour envoyer une notification à un utilisateur utilise la fonction ``notify_user()``. Celle-ci devrait être principalement utilisée lorsque nous voulons notifier un seul utilisateur. Une notification comme celle-ci peut, par exemple, informer que quelqu'un a aimé ou commenté la publication de l'utilisateur.

La fonction est habituellement appelée dans un fichier d'*action*.

Exemple :

Dans cet exemple, un utilisateur (`$user`) déclenche une action pour noter une publication créée par un autre utilisateur (`$owner`). Après avoir enregistré la note (`ElggAnnotation $rating`) dans la base de données, nous pourrions utiliser le code suivant pour envoyer une notification sur la nouvelle note au propriétaire.

```
// Subject of the notification
$subject = elgg_echo('ratings:notification:subject', array(), $owner->language);

// Summary of the notification
$summary = elgg_echo('ratings:notification:summary', array($user->getDisplayName(),
    =>$owner->language);

// Body of the notification message
$body = elgg_echo('ratings:notification:body', array(
    $user->getDisplayName(),
    $owner->getDisplayName(),
    $rating->getValue() // A value between 1-5
), $owner->language);

$params = array(
    'object' => $rating,
    'action' => 'create',
    'summary' => $summary
);

// Send the notification
notify_user($owner->guid, $user->guid, $subject, $body, $params);
```

Note : La langue utilisée par le destinataire n'est pas nécessairement la même que celle de la personne qui déclenche la notification. Par conséquent, vous devez toujours penser à transmettre la langue du destinataire en tant que troisième paramètre à `elgg_echo()`.

Note : Le paramètre `'summary'` est destiné aux plugins de notification qui ne souhaitent afficher qu'un message court au lieu du sujet et du corps. Par conséquent, le résumé doit être laconique, mais toujours contenir toutes les informations nécessaires.

Notifications en file d'attente

Sur les grands sites, il peut y avoir de nombreux utilisateurs qui se sont abonnés pour recevoir des notifications lors d'un événement particulier. L'envoi immédiat de notifications lorsqu'un utilisateur déclenche un tel événement peut ralentir considérablement la vitesse de chargement des pages. C'est pourquoi l'envoi de ces notifications devrait être mis dans la file d'attente de notification de Elgg.

De nouveaux événements de notification peuvent être enregistrés avec la fonction `elgg_register_notification_event()`. Les notifications concernant les événements enregistrés seront envoyées automatiquement à tous les utilisateurs abonnés.

Voici le workflow du système de notifications :

1. **Quelqu'un effectue une action qui déclenche un événement au sein de Elgg**
 - L'action peut être `create`, `update` ou `delete`
 - La cible de l'action peut être n'importe quelle instance de la classe `ElggEntity` (par exemple un article de blog)
2. Le système de notifications enregistre cet événement dans une file d'attente de notifications dans la base de données
3. Lorsque le gestionnaire du hook plugin pour l'intervalle d'une minute est déclenché, l'événement est pris dans la file d'attente et est traité
4. **Les abonnements sont récupérés pour l'utilisateur qui a déclenché l'événement**
 - Par défaut, ceci comprend tous les utilisateurs qui ont activé au moins une méthode de notification pour les utilisateurs via `www.site.com/notifications/personal/<username>`
5. Les plugins peuvent modifier les abonnements avec le hook `[get, subscriptions]`
6. Les plugins peuvent empêcher le traitement de la file d'attente des notifications avec le hook `[send:before, notifications]`
7. Les plugins peuvent modifier les paramètres de notification avec le hook `[prepare, notification]`
8. Les plugins peuvent modifier le sujet, le message et le résumé de la notification avec le hook `[prepare, notification:<action>:<type>:<subtype>]`
9. Les plugins peuvent à mettre en forme le sujet, le message et le résumé de la notification pour chaque méthodes d'envoi individuelle avec le hook `[format, notification:<method>]`
10. **Les notifications sont envoyées à chaque abonné en utilisant les méthodes qu'il a choisies**
 - Les plugins peuvent prendre la main sur chaque notification individuelle ou en bloquer l'envoi avec le hook `[send, notification:<method>]`
11. Le hook `[send:after, notifications]` est déclenché pour l'événement après que toutes les notifications ont été envoyées

Exemple

Dites à Elgg d'envoyer des notifications lorsqu'un nouvel objet de sous-type `photo` est créé :

```
/**
 * Initialize the photos plugin
 */
function photos_init() {
    elgg_register_notification_event('object', 'photo', array('create'));
}
```

Note : Pour envoyer les notifications lors des événements, vous devez configurer l'intervalle *[CRON](#)* d'une minute.

Le contenu du message de notification peut être défini avec le hook 'prepare', 'notification:[action]:[type]:[subtype]'.

Exemple

Dites à Elgg d'utiliser la fonction `photos_prepare_notification()` pour formater le contenu de la notification quand un nouvel objet de sous-type "photo" est créé :

```
/**
 * Initialize the photos plugin
 */
function photos_init() {
    elgg_register_notification_event('object', 'photo', array('create'));
    elgg_register_plugin_hook_handler('prepare', 'notification:create:object:photo',
    ↪ 'photos_prepare_notification');
}

/**
 * Prepare a notification message about a new photo
 *
 * @param string $hook Hook name
 * @param string $type Hook type
 * @param Elgg_Notifications_Notification $notification The notification to prepare
 * @param array $params Hook parameters
 * @return Elgg_Notifications_Notification
 */
function photos_prepare_notification($hook, $type, $notification, $params) {
    $entity = $params['event']->getObject();
    $owner = $params['event']->getActor();
    $recipient = $params['recipient'];
    $language = $params['language'];
    $method = $params['method'];

    // Title for the notification
    $notification->subject = elgg_echo('photos:notify:subject', [$entity->
    ↪ getDisplayName()], $language);

    // Message body for the notification
    $notification->body = elgg_echo('photos:notify:body', array(
        $owner->getDisplayName(),
        $entity->getDisplayName(),
        $entity->getExcerpt(),
        $entity->getURL()
    ), $language);

    // Short summary about the notification
    $notification->summary = elgg_echo('photos:notify:summary', [$entity->
    ↪ getDisplayName()], $language);

    return $notification;
}
```

Note : Assurez-vous que la notification sera dans la bonne langue en indiquant la langue du destinataire à la fonction `elgg_echo()`.

Enregistrement d'une nouvelle méthode de notification

Par défaut, Elgg dispose de deux méthodes de notification : le courrier électronique et le plugin `site_notifications`. Vous pouvez enregistrer une nouvelle méthode de notification avec la fonction `elgg_register_notification_method()`.

Exemple :

Enregistrez un gestionnaire qui enverra les notifications par SMS.

```
/**
 * Initialize the plugin
 */
function sms_notifications_init () {
    elgg_register_notification_method('sms');
}
```

Après avoir enregistré la nouvelle méthode, celle-ci apparaîtra sur la page des paramètres de notification à l'adresse `www.exemple.com/notifications/personal/[username]`.

Envoyer des notifications à l'aide de votre propre méthode

En plus d'enregistrer la méthode de notification, vous devez également enregistrer un gestionnaire qui s'occupe de l'envoi des notifications SMS. Cela se fait avec le hook `'send'`, `'notification:[method]'`.

Exemple :

```
/**
 * Initialize the plugin
 */
function sms_notifications_init () {
    elgg_register_notification_method('sms');
    elgg_register_plugin_hook_handler('send', 'notification:sms', 'sms_notifications_
↪send');
}

/**
 * Send an SMS notification
 *
 * @param string $hook    Hook name
 * @param string $type    Hook type
 * @param bool   $result  Has anyone sent a message yet?
 * @param array  $params  Hook parameters
 * @return bool
 * @internal
 */
```

(suite sur la page suivante)

(suite de la page précédente)

```
function sms_notifications_send($hook, $type, $result, $params) {
    /* @var Elgg_Notifications_Notification $message */
    $message = $params['notification'];

    $recipient = $message->getRecipient();

    if (!$recipient || !$recipient->mobile) {
        return false;
    }

    // (A pseudo SMS API class)
    $sms = new SmsApi();

    return $sms->send($recipient->mobile, $message->body);
}
```

Abonnements

Dans la plupart des cas, Elgg core s'occupe de la gestion des abonnements, de sorte que les plugins de notification n'ont généralement pas besoin de les modifier.

Les abonnements peuvent toutefois être :

- Ajoutées à l'aide de la fonction ``elgg_add_subscription()``
- Supprimées à l'aide de la fonction ``elgg_remove_subscription()``

Il est possible de modifier dynamiquement les destinataires d'une notification avec le hook 'get', 'subscriptions'.

Exemple :

```
/**
 * Initialize the plugin
 */
function discussion_init() {
    elgg_register_plugin_hook_handler('get', 'subscriptions', 'discussion_get_
↳subscriptions');
}

/**
 * Get subscriptions for group notifications
 *
 * @param string $hook      'get'
 * @param string $type      'subscriptions'
 * @param array  $subscriptions Array containing subscriptions in the form
 *                               <user guid> => array('email', 'site', etc.)
 * @param array  $params     Hook parameters
 * @return array
 */
function discussion_get_subscriptions($hook, $type, $subscriptions, $params) {
    $reply = $params['event']->getObject();
}
```

(suite sur la page suivante)

```

    if (!$reply instanceof \ElggDiscussionReply) {
        return $subscriptions;
    }

    $group_guid = $reply->getContainerEntity()->container_guid;
    $group_subscribers = elgg_get_subscriptions_for_container($group_guid);

    return ($subscriptions + $group_subscribers);
}

```

Pièces jointes par email

`notify_user()` ou les notifications mises en file d'attente prennent en charge les pièces jointes pour les notifications par e-mail si elles sont fournies dans `$params`. Pour ajouter une ou plusieurs pièces jointes, ajoutez une clef `attachments` dans `$params`, avec pour valeur un tableau des pièces jointes. Une pièce jointe doit être dans l'un des formats suivants :

- Un `ElggFile` qui pointe vers un fichier existant
- Un tableau avec le contenu du fichier
- Un tableau avec le chemin

```

// this example is for notify_user()
$params['attachments'] = [];

// Example of an ElggFile attachment
$file = new \ElggFile();
$file->owner_guid = <some owner_guid>;
$file->setFilename('<some filename>');

$params['attachments'][] = $file;

// Example of array with content
$params['attachments'][] = [
    'content' => 'The file content',
    'filename' => 'test_file.txt',
    'type' => 'text/plain',
];

// Example of array with filepath
// 'filename' can be provided, if not basename() of filepath will be used
// 'type' can be provided, if not will try a best guess
$params['attachments'][] = [
    'filepath' => '<path to a valid file>',
];

notify_user($to_guid, $from_guid, $subject, $body, $params);

```

3.3.21 Propriété de la page

Une tâche récurrente de tout plugin sera de déterminer qui est propriétaire de la page afin de décider quelles actions sont autorisées ou non. Elgg a un certain nombre de fonctions liées à la propriété de la page et offre également aux développeurs de plugin une flexibilité en laissant le plugin gérer également les demandes de propriété de page. La détermination du propriétaire d'une page peut être déterminée avec `elgg_get_page_owner_guid()`, qui retournera le GUID du propriétaire. Alternativement, `elgg_get_page_owner_entity()` récupérera l'entité complète du propriétaire de la page. Si la page sait déjà qui est le propriétaire de la page, mais pas le système, la page peut définir le propriétaire de la page en passant son GUID à `elgg_set_page_owner_guid($guid)`.

Note : L'entité propriétaire de la page peut être n'importe quelle `ElggEntity`. Si vous souhaitez appliquer des paramètres particuliers selon que c'est un utilisateur ou un groupe, assurez-vous de vérifier que vous avez la l'entité correcte.

Gestionnaires de propriétaire de page personnalisé

Les développeurs de plugins peuvent créer des gestionnaires de propriétaire de page, ce qui peut être nécessaire dans certains cas, par exemple lors de l'intégration de fonctionnalités tierces. Le gestionnaire sera une fonction qui devra être enregistrée via `elgg_register_plugin_hook_handler('page_owner', 'system', 'nom_de_votre_fonction_gestionnaire_de_page')`; Le gestionnaire devra seulement renvoyer une valeur (un GUID entier) lorsqu'il sait avec certitude qui est le propriétaire de la page.

Par défaut, le système utilise `default_page_owner_handler()` pour déterminer le propriétaire de la page (`page_owner`) à partir des éléments suivants :

- Le paramètre d'URL `username`
- Le paramètre d'URL `owner_guid`
- Le chemin de l'URL

Il passe ensuite à tous les gestionnaires de propriétaire de page définis à l'aide du hook plugin. Si aucun propriétaire de page ne peut être déterminé, le propriétaire de la page est défini sur 0, ce qui correspond à l'utilisateur déconnecté.

3.3.22 Vérification des permissions

Avertissement : Comme indiqué dans la page, cette méthode fonctionne **seulement** pour accorder le droit d'accès en **écriture (write)** sur les entités. Vous **ne pouvez pas** utiliser cette méthode pour récupérer ou afficher des entités pour lesquelles l'utilisateur n'a pas d'accès en lecture.

Elgg fournit un mécanisme de vérification des autorisations d'écriture via le hook plugin `permissions_check`. Ceci est utile pour permettre à un plugin d'écrire sur toutes les entités accessibles indépendamment des autorisations d'accès. Les entités qui sont masquées, cependant, ne seront pas accessibles au plugin.

Modifier les permission d'accès avec un hook dans permissions_check

Dans votre plugin, vous devez enregistrer le hook plugin pour permissions_check.

```
elgg_register_plugin_hook_handler('permissions_check', 'all', 'myplugin_permissions_check
→');;
```

La fonction de substitution

Créez maintenant la fonction qui sera appelée par le hook de contrôle des permission. Dans cette fonction, nous déterminons si l'entité (dans les paramètres) a accès à l'écriture. Étant donné qu'il est important de garder Elgg sécurisé, l'accès à l'écriture ne doit être donné qu'après vérification d'une variété de situations, y compris le contexte de la page, l'identification de l'utilisateur, etc. Notez que cette fonction peut renvoyer 3 valeurs : true si l'entité a l'accès à l'écriture, false si l'entité ne le fait pas, et null si ce plugin ne s'en soucie pas et que le système de sécurité doit consulter d'autres plugins.

```
function myplugin_permissions_check($hook_name, $entity_type, $return_value,
→$parameters) {
    $has_access = determine_access_somewhat();

    if ($has_access === true) {
        return true;
    } else if ($has_access === false) {
        return false;
    }

    return null;
}
```

Exemple complet

Voici un exemple complet utilisant le contexte pour déterminer si l'entité a un accès en écriture.

```
<?php

function myaccess_init() {
    // Register cron hook
    if (!elgg_get_plugin_setting('period', 'myaccess')) {
        elgg_set_plugin_setting('period', 'fiveminute', 'myaccess');
    }

    // override permissions for the myaccess context
    elgg_register_plugin_hook_handler('permissions_check', 'all', 'myaccess_permissions_
→check');

    elgg_register_plugin_hook_handler('cron', elgg_get_plugin_setting('period', 'myaccess
→'), 'myaccess_cron');
}

/**
 * Hook for cron event.
```

(suite sur la page suivante)

(suite de la page précédente)

```

*/
function myaccess_cron($event, $object_type, $object) {

    elgg_push_context('myaccess_cron');

    // returns all entities regardless of access permissions.
    // will NOT return hidden entities.
    $entities = get_entities();

    elgg_pop_context();
}

/**
 * Overrides default permissions for the myaccess context
 */
function myaccess_permissions_check($hook_name, $entity_type, $return_value,
↪ $parameters) {
    if (elgg_in_context('myaccess_cron')) {
        return true;
    }

    return null;
}

// Initialise plugin
register_elgg_event_handler('init', 'system', 'myaccess_init');
?>

```

3.3.23 Plugins

Les plugins doivent fournir un fichier `manifest.xml` situé dans la racine du plugin afin d'être reconnu par Elgg.

Contenu

- *elgg-plugin.php*
- *Classe d'initialisation Bootstrap*
- *start.php*
- *elgg-services.php*
- *manifest.xml*
- *composer.json*
- *Tests*
- *Connexe*

elgg-plugin.php

elgg-plugin.php est un fichier statique de configuration du plugin. Il est lu par Elgg pour configurer différents services, et doit retourner un tableau s'il est présent. Il ne doit pas être inclus par les plugins et n'est pas destiné à être exécuté. Outre des constantes magiques comme `__DIR__`, sa valeur de retour ne doit pas changer. Les sections actuellement prises en charge sont :

- `bootstrap` - définit une classe utilisée pour initialiser le plugin
- `entités` - définit les types et les classes d'entités du plugin, et les enregistre éventuellement pour la recherche
- `actions` - élimine la nécessité d'appeler `elgg_register_action()`
- `routes` - élimine la nécessité d'appeler `elgg_register_route()`
- `settings` - élimine la nécessité de définir des valeurs par défaut sur chaque appel à `elgg_get_plugin_setting()`
- `user_settings` - élimine la nécessité de définir des valeurs par défaut sur chaque appel à `elgg_get_plugin_user_setting()`
- `views` - permet aux plugins de créer des alias des actifs vers un chemin d'accès dans le système de vue de Elgg
- `widgets` - élimine la nécessité d'appeler `elgg_register_widget_type()`
- `events` - élimine la nécessité d'appeler `elgg_register_event_handler()`
- `hooks` - élimine la nécessité d'appeler `elgg_register_plugin_hook_handler()`
- `cli_commands` - un tableau de classes `Elgg/Cli/Command` pour étendre les fonctionnalités de `elgg-cli`
- `view_extensions` - élimine la nécessité d'appeler `elgg_extend_view()` ou `elgg_unextend_view()`
- `thème` - un tableau de variables pour les thèmes

```
return [
    // Bootstrap must implement \Elgg\PluginBootstrapInterface
    'bootstrap' => MyPluginBootstrap::class,

    'entities' => [
        // Register a new object subtype and tell Elgg to use a specific
        ↪class to instantiate it
        'type' => 'object',
        'subtype' => 'my_object_subtype',
        'class' => MyObjectClass::class,

        // Register this subtype for search
        'searchable' => true,
    ],

    'actions' => [
        // Registers an action
        // By default, action is registered with 'logged_in' access
        // By default, Elgg will look for file in plugin's actions/ directory:
        ↪actions/my_plugin/action.php
        'my_plugin/action/default' => [],

        'my_plugin/action/custom_access' => [
            'access' => 'public', // supports 'public', 'logged_in', 'admin'
        ],

        // you can use action controllers instead of action files by setting the
        ↪controller parameters
        // controller must be a callable that receives \Elgg\Request as the
        ↪first and only argument
```

(suite sur la page suivante)

(suite de la page précédente)

```

        // in example below, MyActionController::__invoke(\Elgg\Request
↪ $request) will be called
        'my_plugin/action/controller' => [
            'controller' => MyActionController::class,
        ],
    ],

    'routes' => [
        // routes can be associated with resource views or controllers
        'collection:object:my_object_subtype:all' => [
            'path' => '/my_stuff/all',
            'resource' => 'my_stuff/all', // view file is in resources/my_
↪ stuff/all
        ],

        // similar to actions, routes can be associated with a callable.
↪ controller that receives an instance of \Elgg\Request
        'collection:object:my_object_subtype:json' => [
            'path' => '/my_stuff/json',
            'controller' => JsonDumpController::class,
        ],

        // route definitions support other parameters, such as 'middleware',
↪ 'requirements', 'defaults'
        // see elgg_register_route() for all options
    ],

    'widgets' => [
        // register a new widget
        // corresponds to a view in widgets/my_stuff/content
        'my_stuff' => [
            'description' => elgg_echo('widgets:my_stuff'),
            'context' => ['profile', 'dashboard'],
        ],
    ],

    'settings' => [
        'plugin_setting_name' => 'plugin_setting_value',
    ],

    'user_settings' => [
        'user_setting_name' => 'user_setting_value',
    ],

    'views' => [
        'default' => [
            'cool_lib/' => __DIR__ . '/vendors/cool_lib/dist/',
        ],
    ],

    'hooks' => [
        'register' => [

```

(suite sur la page suivante)

```

        'menu:owner_block' => [
            'blog_owner_block_menu' => [
                'priority' => 700,
            ],
        ],
    ],
    'likes:is_likable' => [
        'object:blog' => [
            'Elgg\Values::getTrue' => [],
        ],
    ],
    'usersettings:save' => [
        'user' => [
            '_elgg_save_notification_user_settings' => ['unregister' ↵
↵=> true],
        ],
    ],
],
'events' => [
    'delete' => [
        'object' => [
            'file_handle_object_delete' => [
                'priority' => 999,
            ],
        ],
    ],
    'create' => [
        'relationship' => [
            '_elgg_send_friend_notification' => [],
        ],
    ],
    'log' => [
        'systemlog' => [
            'system_log_default_logger' => ['unregister' => true],
        ],
    ],
],
'cli_commands' => [
    \My\Plugin\CliCommand::class,
    '\My\Plugin\OtherCliCommand',
],
'view_extensions' => [
    'elgg.js' => [
        'bookmarks.js' => [],
    ],
    'page/components/list' => [
        'list/extension' => [
            'priority' => 600,
        ],
    ],
],

```

(suite sur la page suivante)

(suite de la page précédente)

```

        ],
        'forms/usersettings/save' => [
            'core/settings/account/notifications' => [
                'unextend' => true,
            ],
        ],
    ],
    'theme' => [
        'body-background-color' => '#000',
    ],
];

```

Classe d'initialisation Bootstrap

À partir d'Elgg 3.0, la manière recommandée pour initialiser votre plugin est d'utiliser une classe bootstrap. Cette classe doit implémenter l'interface `\Elgg\PluginBootstrapInterface`. Vous pouvez inscrire votre classe bootstrap dans `elgg-plugin.php`.

L'interface d'initialisation bootstrap définit plusieurs fonctions à implémenter qui sont appelées lors de différents événements dans le processus de démarrage du système.

Voir aussi :

Pour plus d'informations sur les différentes fonctions définies dans `\Elgg\PluginBootstrapInterface` veuillez lire *Initialisation (bootstrap) d'un plugin*

start.php

Le fichier `start.php` initialise le plugin en enregistrant les écouteurs d'événements (event listeners) et les hooks plugin.

Il est conseillé que les plugins renvoient une instance de Closure à partir du `start.php`, au lieu de placer les enregistrements dans la racine du fichier. Cela permet une cohérence dans l'application d'initialisation, en particulier à des fins de test.

```

function my_plugin_does_something_else() {
    // Some procedural code that you want to run before any events are fired
}

function my_plugin_init() {
    // Your plugin's initialization logic
}

function my_plugin_rewrite_hook() {
    // Path rewrite hook
}

return function() {
    my_plugin_do_something_else();
    elgg_register_event_handler('init', 'system', 'my_plugin_init');
};

```

(suite sur la page suivante)

(suite de la page précédente)

```
elgg_register_plugin_hook_handler('route:rewrite', 'profile', 'my_plugin_rewrite_hook
↪');
}
```

elgg-services.php

Les plugins peuvent joindre leurs services au conteneur public DI de Elgg en fournissant des définitions PHP-DI dans `elgg-services.php`, situé dans la racine du répertoire de plugin.

Ce fichier doit renvoyer un tableau de définitions PHP-DI. Les services seront disponibles via `elgg()`.

```
return [
    PluginService::class => \DI\object()->constructor(\DI\get(DependencyService::class)),
];
```

Les plugins peuvent ensuite utiliser l'API PHP-DI pour s'auto-câbler et appeler le service :

```
$service = elgg()->get(PluginService::class);
```

Voyez la [documentation PHP-DI](#) pour une liste complète des possibilités de définition et d'invocation.

Syntaxe

Voici un exemple trivial de configuration des emplacements des vues via la clef `views` :

```
return [
    'views' => [
        'default' => [
            'file/icon/' => __DIR__ . '/graphics/icons',
        ],
    ],
];
```

manifest.xml

Les plugins Elgg doivent disposer d'un fichier `manifest.xml` situé dans la racine de chaque plugin.

Le fichier `manifest.xml` contient des informations sur le plugin lui-même, des exigences pour exécuter le plugin et des informations facultatives, y compris l'endroit où afficher les paramètres du plugin dans la zone d'administration et les API fournies par le plugin.

Syntaxe

Le fichier manifeste est un fichier XML standard en UTF-8. Tout est un enfant de l'élément `<plugin_manifest>`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
```

La syntaxe du manifest est la suivante :

```
<name>value</name>
```

De nombreux éléments peuvent contenir des attributs enfants :

```
<parent_name>
  <child_name>value</child_name>
  <child_name_2>value_2</child_name_2>
</parent_name>
```

Éléments requis

Tous les plugins doivent définir les éléments suivants dans leurs fichiers manifest :

- id - Doit être le nom du répertoire du plugin.
- nom - Le nom d'affichage du plugin.
- author - Le nom de l'auteur qui a écrit le plugin.
- version - La version du plugin.
- description - Une description de ce que le plugin fournit, ses caractéristiques, et d'autres informations pertinentes
- requires - Chaque plugin doit spécifier la version de Elgg pour laquelle il a été développé. Pour plus d'informations, consultez la page sur les Dépendances d'un plugin.

Éléments disponibles

En plus des éléments requis ci-dessus, les éléments suivants peuvent être utilisés :

- blurb - Une courte description du plugin.
- category - La catégorie du plugin. Il est recommandé de suivre les :doc : *recommandations* et d'utiliser l'une des catégories définies. Il peut y avoir plusieurs entrées.
- conflicts - Spécifie que le plugin est en conflit avec une certaine configuration système.
- copyright - Les informations sur le droit d'auteur du plugin.
- license - Informations sur la licence du plugin.
- provides - Spécifie que ce plugin fournit la même fonctionnalité qu'un autre plugin Elgg ou une extension PHP.
- suggests - Similaire au système requires, mais n'a pas d'effet si le plugin n'est pas activé. Utilisé pour suggérer d'autres plugins qui interagissent avec ou sont utilisés par le plugin.
- website - Un lien vers l'adresse web du plugin.

Voir aussi :

Dépendances du plugin

Exemple simple

Ce fichier de manifest est le strict minimum qu'un plugin doit avoir.

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>Example Manifest</name>
  <author>Elgg</author>
  <version>1.0</version>
  <description>This is a simple example of a manifest file. In this example, there
  ↪are no dependencies, or additional information about the plugin.</description>

  <requires>
    <type>elgg_release</type>
    <version>1.9</version>
  </requires>
</plugin_manifest>
```

Exemple avancé

Cet exemple utilise tous les éléments disponibles :

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>Example Manifest</name>
  <author>Brett Profitt</author>
  <version>1.0</version>
  <blurb>This is an example manifest file.</blurb>
  <description>This is a simple example of a manifest file. In this example, there
  ↪are many options used, including dependencies, and additional information about the
  ↪plugin.</description>
  <website>http://www.elgg.org/</website>
  <copyright>(C) Brett Profitt 2014</copyright>
  <license>GNU Public License version 2</license>

  <category>3rd_party_integration</category>

  <requires>
    <type>elgg_release</type>
    <version>1.9.1</version>
  </requires>

  <provides>
    <type>plugin</type>
    <name>example_plugin</name>
    <version>1.5</version>
  </provides>

  <suggests>
    <type>plugin</type>
    <name>twitter</name>
    <version>1.0</version>
```

(suite sur la page suivante)

(suite de la page précédente)

```
</suggests>
</plugin_manifest>
```

composer.json

Depuis que Elgg prend en charge l'installation en tant que dépendance [Composer](#), le fait que vos plugins aient également la prise en charge de Composer facilite l'installation par les administrateurs du site. Afin de rendre votre plugin compatible avec Composer, vous devez au moins avoir un fichier `composer.json` dans la racine de votre plugin.

Voici un exemple de fichier `composer.json` :

```
{
    "name": "company/example_plugin",
    "description": "Some description of the plugin",
    "type": "elgg-plugin",
    "keywords": ["elgg", "plugin"],
    "license": "GPL-2.0-only",
    "support": {
        "source": "URL to your code repository",
        "issues": "URL to your issue tracker"
    },
    "require": {
        "composer/installers": "^1.0.8"
    },
    "conflict": {
        "elgg/elgg": "<3.0"
    }
}
```

Apprenez-en plus sur le format de `composer.json` sur le site de [Composer](#).

Les parties importantes du fichier `composer.json` sont :

- **name** : le nom de votre plugin, utilisez le même que le nom de dossier de votre plugin pour assurer une installation correcte
- **type** : indique à Composer où installer votre plugin, laissez TOUJOURS ceci sur `elgg-plugin`
- **require** : l'exigence `composer/installers` vise à s'assurer que Composer sait où installer votre plugin

À titre de suggestion, incluez une règle de **conflict** avec n'importe quelle version de Elgg inférieure à votre version minimale requise, cela aidera à éviter l'installation accidentelle de votre plugin sur une version de Elgg incompatible.

Après avoir ajouté un fichier `composer.json` à votre projet de plugin, vous devez enregistrer votre projet sur [Packagist](#) afin que d'autres personnes puissent installer votre plugin.

Tests

Il est recommandé de créer des tests PHPUnit pour votre plugin. Tous les tests doivent être situés dans les tests d'intégration `tests/phpunit/unit` pour les tests unitaires et `tests/phpunit/integration` pour les tests d'intégration.

Un exemple simple d'ajout de test est le test `ViewStackTest`, qui va vérifier que les vues de votre plugin sont enregistrées correctement et n'ont pas d'erreurs de syntaxe. Pour ajouter ce test, créez un fichier `ViewStackTest.php` dans le dossier `tests/phpunit/unit/<YourNamespace>/<YourPluginName>/` avec le contenu suivant :

```
namespace <YourNameSpace>\<YourPluginName>;

/**
 * @group ViewsService
 */
class ViewStackTest extends \Elgg\Plugins\ViewStackTest {

}
```

Note : Si vous souhaitez voir un meilleur exemple, regardez dans l'un des plugins du noyau de Elgg.

Voir aussi :

Écrire des tests

Connexe

Squelette du plugin

Ce qui suit est le standard pour la structure d'un plugin dans Elgg à partir de Elgg 2.0.

Exemple de structure

Voici un exemple de plugin avec un structure standard. Pour plus d'explications sur cette structure, voyez les informations dans les sections suivantes. Votre plugin peut ne pas avoir besoin de tous les fichiers listés

Les fichiers suivants pour le plugin `example` iraient dans `/mod/example/`

```
actions/
  example/
    action.php
    other_action.php
classes/
  VendorNamespace/
    PluginNamespace/
      ExampleClass.php
languages/
  en.php
vendors/
  example_3rd_party_lib/
views/
  default/
    example/
      component.css
      component.js
      component.png
  forms/
    example/
      action.php
      other_action.php
```

(suite sur la page suivante)

(suite de la page précédente)

```

object/
  example.php
  example/
    context1.php
    context2.php
plugins/
  example/
    settings.php
    usersettings.php
resources/
  example/
    all.css
    all.js
    all.php
    owner.css
    owner.js
    owner.php
widgets/
  example_widget/
    content.php
    edit.php
elgg-plugin.php
CHANGES.txt
COPYRIGHT.txt
INSTALL.txt
LICENSE.txt
manifest.xml
README.txt
start.php
composer.json

```

Fichiers requis

Les plugins **doivent** fournir un fichier `manifest.xml` dans la racine du plugin afin d'être reconnus par Elgg.

De ce fait, voici la structure conforme minimale :

```

mod/example/
  manifest.xml

```

Actions

Les plugins *devraient* placer les scripts pour les actions dans un dossier `actions/`, et *devraient* en outre utiliser le nom de l'action pour déterminer l'emplacement dans ce dossier.

Par exemple, l'action `my/example/action` irait dans `my_plugin/actions/my/example/action.php`. Ceci permet de rendre évident quel script est associé avec quelle action.

De même, le corps du formulaire qui est soumis à cette action doit être situé dans `forms/mon/exemple/action.php`. Non seulement cela rend évident le lien entre le gestionnaire d'action, le code de formulaire et le nom de l'action, mais cela vous permet aussi d'utiliser facilement la fonction `elgg_view_form()`.

Fichiers texte

Les plugins *peuvent* fournir divers fichiers *.txt comme documentation supplémentaire pour le plugin. Ces fichiers **doivent** utiliser la syntaxe Markdown et vont générer des liens vers les sections de gestion du plugins.

README.txt *devrait* fournir des informations supplémentaires de toute nature sur le plugin

COPYRIGHT.txt Si inclus, **doit** fournir une description du droit d'auteur du plugin, en plus de ce qui est inclus dans `manifest.xml`

LICENSE.txt Si inclus, **doit** fournir le texte de la licence sous laquelle le plugin est publié.

INSTALL.txt Si inclus, **doit** fournir des instructions supplémentaires pour l'installation du plugin si le processus est suffisamment compliqué (par exemple, s'il nécessite l'installation de bibliothèques tierces sur l'hôte, ou nécessite l'acquisition d'une clef API provenant d'un tiers).

CHANGES.txt Si inclus, **doit** fournir une liste de modifications pour leur plugin, regroupées par numéro de version, avec la version la plus récente au début.

Les plugins *peuvent* inclure des fichiers supplémentaires *.txt en plus de ceux-ci, mais aucune interface n'est donnée pour les lire.

Pages

Pour afficher des pages complètes, les plugins doivent utiliser les **vues de ressources** (qui ont des noms commençant par `resources/`). Cela permet à d'autres plugins de remplacer facilement certaines fonctionnalités via le système de vue.

Note : La raison pour laquelle nous encourageons cette structure est

- Pour former une relation logique entre urls et scripts, de sorte que les personnes qui examinent le code puissent avoir une idée de ce qu'il fait rien qu'en examinant la structure du code.
 - Pour nettoyer le répertoire racine des plugins, qui avait historiquement été rapidement encombré avec les scripts de gestion des pages.
-

Classes

Elgg fournit un autochargement [PSR-0](#) de toutes les classes du répertoire `classes/` des plugins actifs.

Vous êtes encouragés à suivre les standards [PHP-FIG](#) quand vous écrivez vos classes.

Note : Les fichiers avec une extension « `.class.php` » ne seront **pas** reconnus par Elgg.

Lors de l'organisation de vos classes, Elgg n'impose pas une structure spécifique. Utilisez ce qui fonctionne le mieux pour votre plugin, mais gardez à l'esprit qu'il devrait être facile à lire, et que ses fonctionnalités devraient être faciles à trouver, et qu'avoir des fonctions séparées dans différentes classes permettra d'améliorer la maintenance et la testabilité.

Fournisseurs (vendors)

Les bibliothèques tierce-partie *devraient* être placées dans le dossier `vendors/` de la racine du plugin. Quoique ce dossier n'ait pas de signification particulière pour Elgg, il a été historiquement l'emplacement où le noyau d'Elgg conservait ses bibliothèques tierces-parties, aussi nous encourageons le même format au nom de la cohérence et de la familiarité.

Vues

Afin de surcharger des vues du noyau, les vues d'un plugin peuvent être placées dans `views/`, ou un fichier de configuration `elgg-plugin.php` peut être utilisé pour un mappage fichier/chemin plus détaillé. Voyez [Vues](#).

JavaScript et CSS seront hébergés dans le système de vues. Voyez [JavaScript](#).

Dépendances du plugin

Dans Elgg, le système de dépendances de plugin est présent pour empêcher les plugins d'être utilisés sur des systèmes incompatibles.

Contenu

- *Aperçu*
- *Verbes*
 - *Requiert*
 - *Pré-requis obligatoire : elgg_release*
 - *Suggère*
 - *Est en conflit avec*
 - *Fournit*
- *Types*
 - *elgg_release*
 - *plugin*
 - *priority*
 - *php_extension*
 - *php_ini*
 - *php_version*
- *Opérateurs de comparaison*
- *Exemples rapides*
 - *Requiert Elgg 1.8.2 ou supérieur*
 - *Requiert que le plugin Groups soit activé*
 - *Requiert d'être placé après le plugin Profile si celui-ci est activé*
 - *En conflit avec le plugin The Wire (le Fil)*
 - *Requiert au moins 256 Mo de mémoire dans PHP*
 - *Requiert au moins PHP version 5.3*
 - *Suggère que le plugin TidyPics soit activé*

Aperçu

Le système de dépendances est contrôlé par le fichier `manifest.xml` d'un plugin. Les auteurs de plugins peuvent spécifier qu'un plugin :

- Nécessite certaines versions de Elgg, de plugins Elgg, d'extensions PHP et de paramètres PHP.
- Suggère certaines versions de Elgg, de plugins Elgg, d'extensions PHP et de paramètres PHP.
- Conflits avec certaines versions de Elgg ou de plugins Elgg.
- Fournit l'équivalent d'un autre plugin Elgg ou d'une extension PHP.

Le système de dépendance utilise les quatre verbes ci-dessus (**requires**, **suggests**, **conflicts**, and **provides**) comme éléments parents pour indiquer quel type de dépendance est décrite par ses enfants. Toutes les dépendances ont un format similaire avec des options similaires :

```
<verb>
  <type>type</type>
  <noun>value</noun>
  <noun2>value2</noun2>
</verb>
```

Note : `type` est toujours requis

Verbes

À l'exception des **provides**, tous les verbes utilisent les mêmes six types avec des effets différents, et les options de type sont les mêmes parmi les verbes. **provides** ne prend en charge que `plugin` et `php_extension`.

Requiert

L'utilisation d'une dépendance **requires** signifie que le plugin ne peut pas être activé à moins que la dépendance ne soit exactement remplie.

Pré-requis obligatoire : `elgg_release`

Chaque plugin doit avoir au moins une exigence : la version d'Elgg pour laquelle le plugin est développé. Ceci est spécifié par l'API Elgg `release` (1.8). La comparaison par défaut est `>=`, mais vous pouvez spécifier votre propre comparaison en passant l'élément `<comparison>`.

Utiliser `elgg_release` :

```
<requires>
  <type>elgg_release</type>
  <version>1.8</version>
</requires>
```

Suggère

Les dépendances `suggests` signifient que l’auteur du plugin suggère une configuration système spécifique, mais qui n’est pas indispensable pour utiliser le plugin. Les suggestions peuvent également être un autre plugin qui pourrait interagir, étendre, ou être étendu par ce plugin, mais n’est pas nécessaire pour qu’il fonctionne.

Suggère un autre plugin :

```
<suggests>
  <type>plugin</type>
  <name>profile</name>
  <version>1.0</version>
</suggests>
```

Suggère un paramètres PHP spécifique :

```
<suggests>
  <type>php_ini</type>
  <name>memory_limit</name>
  <value>64M</value>
  <comparison>ge</comparison>
</suggests>
```

Est en conflit avec

Les dépendances `conflicts` signifient que le plugin ne peut pas être utilisé avec une configuration système spécifique.

Conflit avec n’importe quelle version du plugin profile :

```
<conflicts>
  <type>plugin</type>
  <name>profile</name>
</conflicts>
```

Conflit avec une version spécifique de Elgg :

```
<conflicts>
  <type>elgg_release</type>
  <version>1.8</version>
  <comparison>==</comparison>
</conflicts>
```

Fournit

Les dépendances `provides` indiquent à Elgg que ce plugin fournit la fonctionnalité d’un autre plugin ou d’une extension PHP. Contrairement aux autres verbes, il ne prend en charge que deux types : `plugin` et `php_extension`.

Le but de ceci est de fournir des API interchangeables implémentées par différents plugins. Par exemple, le plugin `twitter_services` fournit une API qui permet à d’autres plugins de tweeter au nom de l’utilisateur via curl et OAuth. Un auteur de plugin pourrait écrire un plugin compatible pour les serveurs qui n’utilisent pas curl mais des flux de sockets, et spécifier qu’il fournit `twitter_services`. Tous les plugins qui suggèrent ou requièrent `twitter_services` peuvent ainsi savoir qu’ils peuvent fonctionner.

```
<provides>
  <type>plugin</type>
  <name>twitter_services</name>
  <version>1.8</version>
</provides>
```

Note : Tous les plugins se fournissent eux-même sous la forme de leur ID de plugin (nom d'annuaire), dans la version définie dans leur fichier manifest.

Types

Chaque verbe de dépendance a un élément `<type>` obligatoire qui doit être l'une des six valeurs suivantes :

1. **elgg_release** - La version d'Elgg (3.0)
2. **plugin** - Un plugin Elgg
3. **priority** - Une priorité de chargement de plugin
4. **php_extension** - Une extension PHP
5. **php_ini** - Un paramètre PHP
6. **php_version** - Une version PHP

Note : `provides` ne prend en charge que les types `plugin` et `php_extension`.

Chaque type est défini avec un verbe de dépendance comme élément parent. Les éléments d'option supplémentaires sont au même niveau que l'élément du type :

```
<verb>
  <type>type</type>
  <option_1>value_1</option_1>
  <option_2>value_2</option_2>
</verb>
```

elgg_release

Ceux-ci concernent l'API et les versions de version d'Elgg et nécessitent l'élément d'option suivant :

- **version** - La version de l'API ou de Elgg

L'élément d'option suivant est supporté, mais pas requis :

- **comparison** - L'opérateur de comparaison à utiliser. La valeur par défaut est `>=` si non passée

plugin

Spécifie un plugin Elgg par son ID (nom d'annuaire). Cela nécessite l'élément d'option suivant :

- **name** - L'ID du plugin

Les éléments d'option suivants sont pris en charge, mais non requis :

- **version** - La version du plugin
- **comparison** - L'opérateur de comparaison à utiliser. La valeur par défaut est `>=` si non passée

priority

Cela nécessite que le plugin soit chargé avant ou après un autre plugin, si ce plugin existe. `requires` doit être utilisé pour exiger l'existence d'un plugin. Les éléments d'option suivants sont requis :

- **plugin** - L'ID du plugin sur lequel appuyer l'ordre de chargement
- **priority** - L'ordre de chargement : “before” (avant) ou “after” (après)

php_extension

Ceci vérifie les extensions PHP. L'élément d'option suivant est requis :

- **name** - Le nom de l'extension PHP

Les éléments d'option suivants sont pris en charge, mais non requis :

- **version** - La version de l'extension
- **comparison** - L'opérateur de comparaison à utiliser. La valeur par défaut est `==`

Note : Le format des versions des extensions varie largement entre les extensions PHP et n'est parfois même pas défini. Il est habituellement inutile de la vérifier.

php_ini

Ceci vérifie les paramètres PHP. Les éléments d'option suivants sont requis :

- **name** - Le nom du paramètres à vérifier
- **value** - La valeur du paramètre à laquelle comparer

Les options suivantes sont supportées, mais pas requises :

- **comparison** - L'opérateur de comparaison à utiliser. La valeur par défaut est `==`

php_version

Ceci vérifie la version de PHP. Les éléments d'option suivants sont requis :

- **version** - La version de PHP

L'élément d'option suivant est supporté, mais pas requis :

- **comparison** - L'opérateur de comparaison à utiliser. La valeur par défaut est `>=` si non passée

Opérateurs de comparaison

Les dépendances qui vérifient les versions supportent la passage d'un opérateur personnalisé via l'élément `<comparison>`.

Les opérateurs de comparaison suivants sont valides :

- `<` ou `lt`
- `<=` ou `le`
- `=`, `==`, ou `eq`
- `!=`, `<>`, ou `ne`
- `>` ou `gt`
- `>=` ou `ge`

Si `<comparison>` n'est pas passé, ce qui suit est utilisé comme valeurs par défaut, en fonction du type de dépendance :

- `requires->elgg_release` : `>=`
- `requires->plugin` : `>=`
- `requires->php_extension` : `=`

- requires->php_ini : =
- tous les conflits : =

Note : Vous devez échapper < et > vers > ; et < ; . Pour des comparaisons qui utilisent ces valeurs, il est recommandé que vous utilisiez les chaînes équivalentes à la place !

Exemples rapides

Requiert Elgg 1.8.2 ou supérieur

```
<requires>
  <type>elgg_release</type>
  <version>1.8.2</version>
</requires>
```

Requiert que le plugin Groups soit activé

```
<requires>
  <type>plugin</type>
  <name>groups</name>
</requires>
```

Requiert d'être placé après le plugin Profile si celui-ci est activé

```
<requires>
  <type>priority</type>
  <priority>after</priority>
  <plugin>profile</plugin>
</requires>
```

En conflit avec le plugin The Wire (le Fil)

```
<conflicts>
  <type>plugin</type>
  <name>thewire</name>
</conflicts>
```

Requiert au moins 256 Mo de mémoire dans PHP

```
<requires>
  <type>php_ini</type>
  <name>memory_limit</name>
  <value>256M</value>
  <comparison>ge</comparison>
</requires>
```

Requiert au moins PHP version 5.3

```
<requires>
  <type>php_version</type>
  <version>5.3</version>
</requires>
```

Suggère que le plugin TidyPics soit activé

```
<suggests>
  <type>plugin</type>
  <name>tidypics</name>
</suggests>
```

Initialisation (bootstrap) d'un plugin

Afin d'initialiser votre plugin à partir de Elgg 3.0, vous pouvez utiliser une classe bootstrap. Cette classe doit implémenter l'interface abstraite `\Elgg\PluginBootstrapInterface`, mais il est recommandé d'étendre la classe abstraite `\Elgg\PluginBootstrap`, car certaines préparations ont déjà été faites.

Si vous n'avez besoin que d'un sous-ensemble limité des fonctions d'initialisation, votre classe peut également étendre la classe `\Elgg\DefaultPluginBootstrap`, cette classe a déjà implémenté toutes les fonctions de `\Elgg\PluginBootstrapInterface`. Ainsi, vous pouvez surcharger uniquement les fonctions dont vous avez besoin.

Contenu

- *Enregistrement de la classe « bootstrap » d'initialisation*
- *Fonctions disponibles*
 - `->load()`
 - `->boot()`
 - `->init()`
 - `->ready()`
 - `->shutdown()`
 - `->activate()`
 - `->deactivate()`
 - `->upgrade()`
- *Fonctions d'assistance disponibles*
 - `->elgg()`
 - `->plugin()`

Enregistrement de la classe « bootstrap » d'initialisation

Vous devez enregistrer votre classe d'initialisation dans le fichier `elgg-plugin.php`.

```
return [
    // Bootstrap must implement \Elgg\PluginBootstrapInterface
    'bootstrap' => MyPluginBootstrap::class,
];
```

Fonctions disponibles

->load()

Exécuté lors de l'événement `plugins_load, system`

Permet au plugin de requérir des fichiers supplémentaires, ainsi que de configurer des services avant le démarrage du plugin.

->boot()

Exécuté lors de l'événement `plugins_boot:before, system`

Permet au plugin d'enregistrer des gestionnaires pour les événements `plugins_boot, system` et `init, system`, ainsi que de mettre en œuvre la logique de démarrage.

->init()

Exécuté lors de l'événement `init, system`

Permet au plugin d'implémenter la logique métier et d'enregistrer tous les autres gestionnaires.

->ready()

Exécuté lors de l'événement `ready, system`

Permet au plugin d'implémenter sa logique après que tous les plugins ont été initialisés.

->shutdown()

Exécuté lors de l'événement `shutdown, system`

Permet au plugin d'implémenter sa logique pendant l'arrêt (`shutdown`).

->activate()

Exécuté lorsque le plugin est activé, après l'événement `activate`, `plugin`, et avant que `activate.php` soit inclus.

->deactivate()

Exécuté lorsque le plugin est désactivé, après l'événement `deactivate`, `plugin` et avant que `deactivate.php` soit inclus.

->upgrade()

Enregistré en tant que gestionnaire pour l'événement `upgrade`, `system`

Permet au plugin d'implémenter la logique pendant la mise à niveau du système.

Fonctions d'assistance disponibles

Ceci suppose que votre classe bootstrap étend la classe abstraite `\Elgg\PluginBootstrap` ou la classe `\Elgg\DefaultPluginBootstrap`.

->elgg()

Renvoie le conteneur d'informations public de Elgg. Cela peut être utile si vous souhaitez enregistrer des hooks plugin ou des écouteurs (listeners) d'événements.

```
$hooks = $this->elgg()->hooks;
$hooks->registerHandler('register', 'menu:entity', 'my_custom_menu_callback');

$events = $this->elgg()->events;
$events->registerHandler('create', 'object', MyCustomObjectHandler::class);
```

->plugin()

Renvoie l'entité du plugin à laquelle cette initialisation est liée. Il est ainsi plus facile d'obtenir des paramètres de plugin.

```
$plugin = $this->plugin();
$my_setting = $plugin->getSetting('my_setting');
```

3.3.24 Rivière (Flux d'activité)

Elgg supporte nativement la « rivière » (« river »), un flux d'activité qui contient des descriptions des activités effectuées par les membres du site. Cette page donne un aperçu de comment ajouter des événements à la rivière dans un plugin Elgg.

Pousser des éléments vers la rivière

Des éléments sont poussés vers la rivière d'activité à travers l'appel à une fonction, que vous devez inclure dans vos plugins pour que ces éléments apparaissent.

Ici nous ajoutons un élément à la rivière, qui indique qu'un utilisateur a créé un nouvel article de blog :

```
<?php

elgg_create_river_item([
    'view' => 'river/object/blog/create',
    'action_type' => 'create',
    'subject_guid' => $blog->owner_guid,
    'object_guid' => $blog->getGUID(),
]);
```

Tous les paramètres disponibles :

- **view** => STR La vue qui va gérer l'élément de la rivière (doit exister)
- **action_type** => STR Une chaîne arbitraire pour définir l'action (par ex. "create", "update", "vote", "review", etc.)
- **subject_guid** => INT Le GUID de l'entité qui effectue l'action (par défaut : le guid de l'utilisateur connecté)
- **object_guid** => INT Le GUID de l'entité sur laquelle est effectuée l'action
- **target_guid** => INT Le GUID du conteneur de l'entité objet (optionnel)
- **access_id** => INT L'ID d'accès de l'élément de la rivière (par défaut : identique à celui de l'objet)
- **posted** => INT Le timestamp UNIX de l'élément de la rivière (par défaut : maintenant)
- **annotation_id** => INT L'ID de l'annotation ID associée avec cette entrée de la rivière (optionnel)

Quand un élément est supprimé ou modifié, l'élément de la rivière sera mis à jour automatiquement.

Vues de la rivière

A partir de Elgg 3.0 le paramètre **view** n'est plus requis. Une logique de rechange a été créée pour vérifier une série de vues pour vous :

1. `/river/{ $type }/{ $subtype }/{ $action_type }` : par ex. `river/object/blog/create` seule l'action `create` parviendra à cette vue
2. `river/{ $type }/{ $subtype }/default` : par ex. `river/object/blog/default` toute l'activité de la rivière pour object `blog` arrivera ici
3. `river/{ $type }/{ $action_type }` : par ex. `river/object/create` toutes les actions `create` pour object arriveront ici
4. `river/{ $type }/default` : par ex. `river/object/default` toutes les actions pour tous les object arriveront ici
5. `river/elements/layout` : solution de rechange en dernier recours, qui devrait toujours être appelée dans n'importe laquelle des vues de la rivière pour créer une mise en page cohérente

A la fois **type** et **subtype** sont basés sur le **type** et le **subtype** de l'**object_guid** pour lequel l'élément de la rivière a été créé.

Résumé

Si aucun paramètre `summary` n'est fourni à `river/elements/layout` la vue tentera de le créer pour vous. Le résultat de base sera un texte avec le texte *Untel a fait quelque chose sur Ceci*, où *Untel* est basé sur `subject_guid` et *Ceci* est basé sur `object_guid`. Des liens seront créés pour *Untel* et *Ceci*. Ces liens sont transmis à une série de clefs de traduction afin que vous puissiez générer un résumé informatif.

Les clefs de traduction sont :

1. `river:{type}:{subtype}:{action_type}` : par ex. `river:object:blog:create`
2. `river:{type}:{subtype}:default` : par ex. `river:object:blog:default`
3. `river:{type}:{action_type}` : par ex. `river:object:create`
4. `river:{type}:default` : par ex. `river:object:default`

Vue personnalisée de la rivière

Si vous souhaitez ajouter plus d'informations à la vue de la rivière, comme par ex. une pièce jointe (une image, une vidéo YouTube embarquée, etc), vous devez spécifier la *vue* lors de la création de l'élément de la rivière. Cette vue **DOIT** exister.

Nous recommandons `/river/{type}/{subtype}/{action}`, où :

- `{type}` est le type d'entité du contenu qui nous intéresse (`object` pour les objets, `user` pour les utilisateurs, etc.)
- `{subtype}` est le sous-type d'entité du contenu qui nous intéresse (`blog` pour les articles de blog, `photo_album` pour les albums photo, etc.)
- `{action}` est l'action qui a eu lieu (`create`, `update`, etc)

Les informations relatives à un élément de la rivière seront transmises dans un objet appelé `$vars['item']`, qui contient les paramètres importants suivants :

- `'$vars['item']->subject_guid` Le GUID de l'utilisateur qui exécute l'action
- `$vars['item']->object_guid` Le GUID de l'entité sur laquelle l'action est effectuée

Les horodatages, etc. seront générés pour vous.

Par exemple, le plugin blog utilise le code suivant pour sa vue dans la rivière :

```
$item = elgg_extract('item', $vars);
if (!$item instanceof ElggRiverItem) {
    return;
}

$blog = $item->getObjectEntity();
if (!$blog instanceof ElggBlog) {
    return;
}

$vars['message'] = $blog->getExcerpt();

echo elgg_view('river/elements/layout', $vars);
```

3.3.25 Routage

Elgg dispose de deux mécanismes pour répondre aux requêtes HTTP qui ne passent pas déjà par les systèmes des design/actions et du *Cache simple (Simplecache)*.

Identifiant d'URL et ségments

Après avoir supprimé l'URL du site, Elgg découpe le chemin d'accès de l'URL par / dans un tableau. Le premier élément, l'**identificateur**, est mis de côté, et les éléments restants sont appelés les **segments**. Par exemple, si l'URL du site est `http://exemple.com/elgg/`, l'URL `http://exemple.com/elgg/blog/owner/jane?foo=123` produit :

Identificateur : 'blog'. Segments : ['owner', 'jane']. (les paramètres de la chaîne de requête sont disponibles via `get_input()`)

L'URL du site (page d'accueil) est un cas spécial qui produit un identificateur de chaîne vide et un tableau de segments vides.

Avvertissement : Les identificateurs/segments d'URL doivent être considérés comme des entrées utilisateur potentiellement dangereuses. Elgg utilise dessus `htmlspecialchars` pour échapper les entités HTML.

Gestion de pages

Elgg offre des outils pour gérer les pages de votre plugin via des routes personnalisées, permettant des URL comme `http://votresite/mon_plugin/section`. Vous pouvez enregistrer une nouvelle route en utilisant `elgg_register_route()` ou via la configuration `routes` dans `elgg-plugin.php`. Les routes assurent la correspondance avec les vues de ressources, où vous pouvez générer le contenu de la page.

```
// in your 'init', 'system' handler
elgg_register_route('my_plugin:section' [
    'path' => '/my_plugin/section/{guid}/{subsection?}',
    'resource' => 'my_plugin/section',
    'requirements' => [
        'guid' => '\d+',
        'subsection' => '\w+',
    ],
]);

// in my_plugin/views/default/resources/my_plugin/section.php
$guid = elgg_extract('guid', $vars);
$subsection = elgg_extract('subsection', $vars);

// render content
```

Dans l'exemple ci-dessus, nous avons enregistré une nouvelle route qui est accessible via `http://votresite/mon_plugin/section/<guid>/<subsection>`. Chaque fois que cette route est accédée avec le segment requis `guid` et un segment facultatif `subsection`, le routeur affiche la valeur spécifiée par la ressource `mon_plugin/section` et transmet les paramètres extraits de l'URL à votre vue de ressource via `$vars`.

Nom des routes

Les noms de routes peuvent ensuite être utilisés pour générer une URL :

```
$url = elgg_generate_url('my_plugin:section', [
    'guid' => $entity->guid,
    'subsection' => 'assets',
]);
```

Les noms d'itinéraires sont uniques pour l'ensemble des plugins et le noyau, de sorte qu'un autre plugin peut surcharger une route en enregistrant d'autres paramètres sur le même nom de route.

Les noms des routes respectent un certain formalisme et seront, dans certains cas, utilisés pour résoudre automatiquement les URLs, par exemple pour afficher une entité.

Les conventions suivantes sont utilisées dans le noyau et recommandées pour les plugins :

- view :<entity_type> :<entity_subtype>** Correspond à la page de profil de l'entité, par exemple `view:user:user` ou `view:object:blog`. Le chemin d'accès doit contenir un `guid`, ou un `username` pour les utilisateurs
- edit :<entity_type> :<entity_subtype>** Correspond au formulaire pour modifier l'entité, par exemple `edit:user:user` ou `edit:object:blog`. Le chemin d'accès doit contenir un `guid`, ou un `username` pour les utilisateurs. Si vous avez besoin d'ajouter des sous-ressources, utilisez des suffixes, par exemple `edit:object:blog:images`, en gardant au moins une sous-ressource sans suffixe par défaut.
- add :<entity_type> :<entity_subtype>** Correspond au formulaire pour ajouter une nouvelle entité d'un type donné, par exemple `add:object:blog`. Le chemin d'accès, en règle générale, contient le paramètre `container_guid`
- collection :<entity_type> :<entity_subtype> :<collection_type>** Correspond aux pages de listes. Les noms de routes couramment utilisés dans le noyau sont les suivants :
 - `collection:object:blog:all` : liste tous les articles de blog
 - `collection:object:blog:owner` : liste les articles de blog appartenant à un utilisateur avec un identifiant donné
 - `collection:object:blog:friends` : liste les articles de blog appartenant à des contacts de l'utilisateur connecté (ou d'un utilisateur avec un identifiant donné)
 - `collection:object:blog:group` : liste les articles de blog dans un groupe
- default :<entity_type> :<entity_subtype>** Fait correspondre à la page par défaut d'une ressource, par exemple le chemin d'accès `/blog`. Elgg utilise la collection `all` pour toutes ces routes.
 - `default:object:blog` : gère le chemin générique `/blog`.

<entity_subtype> peut être omis des noms de routes pour enregistrer les routes globales applicables à toutes les entités d'un type donné. Le générateur d'URL essaie d'abord de générer une URL à l'aide du sous-type, puis se replie sur un nom de route sans sous-type. Par exemple, les profils utilisateur sont acheminés vers la même vue de ressource quel que soit le sous-type utilisateur.

```
elgg_register_route('view:object:attachments', [
    'path' => '/attachments/{guid}',
    'resource' => 'attachments',
]);

elgg_register_route('view:object:blog:attachments', [
    'path' => '/blog/view/{guid}/attachments',
    'resource' => 'blog/attachments',
]);

$blog = get_entity($blog_guid);
$url = elgg_generate_entity_url($blog, 'view', 'attachments'); // /blog/view/$blog_guid/
attachments
```

(suite sur la page suivante)

(suite de la page précédente)

```
$other = get_entity($other_guid);
$url = elgg_generate_entity_url($other, 'view', 'attachments'); // /attachments/$other_
↳guid
```

Configuration du routage

Les segments peuvent être définis à l'aide de jokers, par exemple `profile/{username}`, qui correspond à toutes les URL qui contiennent `profile/` suivies d'un nom d'utilisateur arbitraire.

Pour rendre un segment facultatif, vous pouvez ajouter un `?` (point d'interrogation) au nom du joker, par exemple `profile/{username}/{section?}`. Dans ce cas, l'URL sera appariée même si le segment `section` n'est pas fourni.

Vous pouvez contraindre davantage les segments en utilisant les exigences regex :

```
// elgg-plugin.php
return [
    'routes' => [
        'profile' => [
            'path' => '/profile/{username}/{section?}',
            'resource' => 'profile',
            'requirements' => [
                'username' => '[\p{L}\p{Nd}._-]+', // only allow valid
↳usernames
                'section' => '\w+', // can only contain alphanumeric
↳characters
            ],
            'defaults' => [
                'section' => 'index',
            ],
        ],
    ],
];
```

Par défaut, Elgg va définir les exigences suivantes pour les segments d'URL nommés :

```
$patterns = [
    'guid' => '\d+', // only digits
    'group_guid' => '\d+', // only digits
    'container_guid' => '\d+', // only digits
    'owner_guid' => '\d+', // only digits
    'username' => '[\p{L}\p{Nd}._-]+', // letters, digits, underscores, dashes
];
```

Routes dépendant du plugin

Si une route a besoin qu'un plugin spécifique soit actif, cela peut être configuré dans la configuration de la route.

```
// elgg-plugin.php
return [
    'routes' => [
        'collection:object:blog:friends' => [
            'path' => '/blog/friends/{username?}/{lower?}/{upper?}',
            'resource' => 'blog/friends',
            'required_plugins' => [
                'friends', // route only allowed when friends plugin is_
↪ active
            ],
        ],
    ],
];
```

Route middleware

Les middlewares de gestion du routage peuvent être utilisés pour empêcher l'accès à une certaine route, ou pour exécuter une certaine logique avant que la route ne soit rendue. Un middleware peut être utilisé, par exemple pour implémenter un système d'inscription payante, ou pour définir des métadonnées open graph.

Le noyau de Elgg implémente plusieurs gestionnaires de middleware. Le middleware suivant se trouve dans l'espace de noms `\Elgg\Router\Middleware` :

Gatekeeper

Ce gardien (gatekeeper) empêchera l'accès des utilisateurs non authentifiés.

AdminGatekeeper

Ce gardien (gatekeeper) empêchera l'accès des utilisateurs non-admin.

LoggedOutGatekeeper

Ce gardien (gatekeeper) empêchera l'accès des utilisateurs authentifiés.

AjaxGatekeeper

Ce gardien (gatekeeper) empêchera l'accès avec des demandes non-XHR.

PageOwnerCanEditGatekeeper

Ce gardien empêchera l'accès s'il y a un propriétaire de la page détecté et que ce propriétaire ne peut pas être modifié.

GroupPageOwnerCanEditGatekeeper

Ce gardien (gatekeeper) étend `PageOwnerCanEditGatekeeper` mais exige également que le propriétaire de la page soit une entité `ElggGroup`.

UserPageOwnerCanEditGatekeeper

Ce gardien (gatekeeper) étend le `PageOwnerCanEditGatekeeper` mais exige également que le propriétaire soit une entité `ElggUser`.

CsrfFirewall

Ce middleware empêchera l'accès sans les jetons CSRF corrects. Ce middleware sera automatiquement appliqué aux actions.

ActionMiddleware

Ce middleware fournira une logique liée à l'action. Ce middleware sera automatiquement appliqué aux actions.

SignedRequestGatekeeper

Ce gardien (gatekeeper) empêchera l'accès si l'URL a été trafiquée. Une URL sécurisée peut être générée à l'aide de la fonction `elgg_http_get_signed_url`.

UpgradeGatekeeper

Ce gardien (gatekeeper) empêchera l'accès si l'URL de mise à niveau est sécurisée et que l'URL n'est pas valide.

WalledGarden

Ce middleware empêchera l'accès à une route si le site est configuré uniquement pour les utilisateurs authentifiés et qu'aucun utilisateur authentifié ne s'est connecté. Ce middleware est automatiquement activé pour toutes les routes. Vous pouvez désactiver le gardien du walled garden avec une option de *configuration de route*.

Custom Middleware

Les gestionnaires middleware peuvent être configurés sur n'importe quel fonction callable (callable) qui reçoit une instance de `\Elgg\Request` : le gestionnaire doit lancer une instance de `HttpException` pour empêcher l'accès à la route. Le gestionnaire peut renvoyer une instance de `\Elgg\Http\ResponseBuilder` pour empêcher une implémentation ultérieure de la séquence de routage (une réponse de redirection peut être retournée pour réorienter la requête).

```
class MyMiddleware {

    public function __invoke(\Elgg\Request $request) {
        $entity = $request->getEntityParam();
        if ($entity) {
            // do stuff
        } else {
            throw new EntityNotFoundException();
        }
    }
}

elgg_register_route('myroute', [
    'path' => '/myroute/{guid?}',
    'resource' => 'myroute',
    'middleware' => [
        \Elgg\Router\Middleware\Gatekeeper::class,
        MyMiddleware::class,
    ]
]);
```

Contrôleurs de routage

Dans certains cas, l'utilisation des vues « ressource » n'est pas appropriée. Dans ces cas, vous pouvez utiliser un contrôleur - toute fonction callable qui reçoit une instance de `\Elgg\Request` :

```
class MyController {

    public function handleFoo(\Elgg\Request $request) {
        elgg_set_http_header('Content-Type: application/json');
        $data = [
            'entity' => $request->getEntityParam(),
        ];
        return elgg_ok_response($data);
    }
}

elgg_register_route('myroute', [
    'path' => '/myroute/{guid?}',
    'controller' => [MyController::class, 'handleFoo'],
]);
```

Le hook de plugin `route:rewrite`

Pour la réécriture de l'URL, le hook `route:rewrite` (avec des arguments similaires à `route`) est déclenché très tôt, et permet de modifier le chemin d'accès de l'URL de requête (par rapport au site Elgg).

Ici, nous réécrivons les requêtes de `news/*` vers `blog/*` :

```
function myplugin_rewrite_handler($hook, $type, $value, $params) {
    $value['identifiant'] = 'blog';
    return $value;
}

elgg_register_plugin_hook_handler('route:rewrite', 'news', 'myplugin_rewrite_handler');
```

Avvertissement : Le hook doit être enregistré directement dans le `start.php` de votre plugin (l'événement `[init, system]` est trop tardif).

Aperçu du routage

Pour les pages standard, le flux de programme d'Elgg est quelque chose comme ceci :

1. Un utilisateur demande `http://exemple.com/news/owner/jane`.
2. Les plugins sont initialisés.
3. Elgg analyse l'URL pour identificateur `news` et les segments `['owner', 'jane']`.
4. Elgg déclenche le hook plugin `route:rewrite`, `news` (voir ci-dessus).
5. Elgg déclenche le hook plugin `route`, `blog` (a été réécrit dans le hook de réécriture).
6. Elgg trouve une route enregistrée qui correspond au chemin d'accès final de la route, et génère une vue de ressource qui lui est associée. Il appelle `elgg_view_resource('blog/owner', $vars)` où `$vars` contient le nom d'utilisateur.
7. La vue `resources/blog/owner` obtient le nom d'utilisateur via `$vars['username']` et utilise de nombreuses autres vues et fonctions de mise en forme comme `elgg_view_layout()` et `elgg_view_page()` pour créer la page HTML entière.
8. PHP invoque la séquence d'arrêt d'Elgg.
9. L'utilisateur reçoit une page complète.

Les normes de codage d'Elgg suggèrent une disposition d'URL particulière, mais il n'y a pas de syntaxe obligatoire.

3.3.26 Recherche

Contenu

- *Recherche d'entité*
- *Champs de recherche*
- *Entités recherchables*
- *Types de recherches personnalisés*
- *Points de terminaison pour l'autocomplétion et la recherche en direct*

Recherche d'entité

Le noyau d'Elgg fournit des `elgg_search()` flexibles, qui préparent des clauses de recherche personnalisées et utilisent `elgg_get_entities()` pour obtenir les résultats.

Outre tous les paramètres acceptés par `elgg_get_entities()`, `elgg_search()` accepte les paramètres suivants :

- `query` Requête de recherche
- `fields` Un tableau de noms par type de propriété à rechercher (voir l'exemple ci-dessous)
- `sort` Un tableau contenant des options de tri, comprenant *property*, *property_type* et *direction*
- `type` Type d'entité à rechercher
- `subtype` Sous-type d'entité à rechercher (facultatif)
- `search_type` Type de recherche personnalisé (requis si aucun `type` n'est fourni)
- **partial_match** Autoriser les correspondances partielles Par défaut, les correspondances partielles sont autorisées, ce qui signifie que `elgg` sera trouvé lors de la recherche de `el`. Les correspondances exactes peuvent être utiles lorsque vous souhaitez trouver les correspondances pour une métadonnée précise, par exemple lorsque vous voulez trouver tous les objets dont la couleur est `red` et non `darkred`
- **tokenize** Décomposer la requête de recherche en jetons Par défaut, les requêtes de recherche utilisent des jetons, ce qui signifie que nous allons trouver `elgg` a été publié lors d'une recherche sur `elgg` publié

```
// List all users who list United States as their address or mention it in their_
↳description
$options = [
    'type' => 'user',
    'query' => 'us',
    'fields' => [
        'metadata' => ['description'],
        'annotations' => ['location'],
    ],
    'sort' => [
        'property' => 'zipcode',
        'property_type' => 'annotation',
        'direction' => 'asc',
    ]
];

echo elgg_list_entities($options, 'elgg_search');
```

Champs de recherche

Vous pouvez personnaliser les champs de recherche de chaque type/sous-type d'entité, à l'aide du hook `search:fields` :

```
// Let's remove search in location and add address field instead
elgg_register_plugin_hook_handler('search:fields', 'user', 'my_plugin_search_user_fields
↳');

function my_plugin_search_user_fields(\Elgg\Hook $hook) {
    $fields = $hook->getValue();
    $location_key = array_search('location', $fields['annotations']);
    if ($location_key) {
        unset($fields[$location_key]['annotations']);
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

    $fields['metadata'][] = 'address';

    return $fields;
}

```

Entités recherchables

Pour enregistrer un type d'entité pour la recherche, utilisez `elgg_register_entity_type()`, ou faites-le lors de la définition d'un type d'entité dans `elgg-plugin.php`. Pour combiner des résultats de recherche ou filtrer la façon dont les résultats de recherche sont présentés dans le plugin de recherche, utilisez le hook `'search:config'`, `'type_subtype_pairs'`.

```

// Let's add places and place reviews as public facing entities
elgg_register_entity_type('object', 'place');
elgg_register_entity_type('object', 'place_review');

// Now let's include place reviews in the search results for places
elgg_register_plugin_hook_handler('search:options', 'object:place', 'my_plugin_place_
↳search_options');
elgg_register_plugin_hook_handler('search:config', 'type_subtype_pairs', 'my_plugin_
↳place_search_config');

// Add place review to search options as a subtype
function my_plugin_place_search_options($hook, $type, $value, $params) {

    if (empty($params) || !is_array($params)) {
        return;
    }

    if (isset($params['subtypes'])) {
        $subtypes = (array) $params['subtypes'];
    } else {
        $subtypes = (array) elgg_extract('subtype', $params);
    }

    if (!in_array('place', $subtypes)) {
        return;
    }

    unset($params["subtype"]);

    $subtypes[] = 'place_review';
    $params['subtypes'] = $subtypes;

    return $params;
}

// Remove place reviews as a separate entry in search sections
function my_plugin_place_search_config(\Elgg\Hook $hook) {

    $types = $hook->getValue();

```

(suite sur la page suivante)

(suite de la page précédente)

```

if (empty($types['object'])) {
    return;
}

foreach ($types['object'] as $key => $subtype) {
    if ($subtype == 'place_review') {
        unset($types['object'][$key]);
    }
}

return $types;
}

```

Types de recherches personnalisés

Le noyau de Elgg prend en charge uniquement la recherche d'entités. Vous pouvez implémenter des recherches personnalisées, par exemple en utilisant une requête de recherche comme un emplacement pour lister les entités à proximité de cet emplacement.

```

// Let's added proximity search type
elgg_register_plugin_hook_handler('search:config', 'search_types', function (\Elgg\Hook
↳ $hook) {
    $search_types = $hook->getValue();
    $search_types[] = 'promimity';

    return $search_types;
});

// Let's add search options that will look for entities that have geo coordinates and
↳ order them by proximity to the query location
elgg_register_plugin_hook_handler('search:options', 'proximity', function (\Elgg\Hook
↳ $hook) {

    $query = $hook->getParam('query');
    $options = $hook->getValue();

    // Let's presume we have a geocoding API
    $coords = geocode($query);

    // We are not using standard 'selects' options here, because counting queries do not
↳ use custom selects
    $options['wheres']['proximity'] = function (QueryBuilder $qb, $alias) use ($lat,
↳ $long) {
        $dblat = $qb->joinMetadataTable($alias, 'guid', 'geo:lat');
        $dblong = $qb->joinMetadataTable($alias, 'guid', 'geo:long');

        $qb->addSelect("(((acos(sin(($lat*pi()/180))
            *sin(($dblat.value*pi()/180)) + cos(($lat*pi()/180))
            *cos(($dblat.value*pi()/180))
            *cos((( $long-$dblong.value)*pi()/180)))))*180/pi()

```

(suite sur la page suivante)

(suite de la page précédente)

```

        *60*1.1515*1.60934
        AS proximity");

    $qb->orderBy('proximity', 'asc');

    return $qb->merge([
        $qb->compare("$dblat.value", 'is not null'),
        $qb->compare("$dblong.value", 'is not null'),
    ]);
};

return $options;
});

```

Points de terminaison pour l'autocomplétion et la recherche en direct

Le noyau fournit un point de terminaison JSON pour la recherche d'utilisateurs et de groupes. Ces points de terminaison sont utilisés par les vues `input/autocomplete` et `input/userpicker`.

```

// Get JSON results of a group search for 'class'
$json = file_get_contents('http://example.com/livesearch/groups?view=json&q=class');

```

Vous pouvez ajouter des types de recherche personnalisés en ajoutant une vue de ressource correspondante :

```

// Let's add an endpoint that will search for users that are not members of a group
// and render a userpicker for our invite form
echo elgg_view('input/userpicker', [
    'handler' => 'livesearch/non_members',
    'options' => [
        // this will be sent as URL query elements
        'group_guid' => $group_guid,
    ],
]);

// To enable /livesearch/non_members endpoint, we need to add a view
// in /views/json/resources/livesearch/non_members.php

$limit = get_input('limit', elgg_get_config('default_limit'));
$query = get_input('term', get_input('q'));
$input_name = get_input('name');

// We have passed this value to our input view, and we want to make sure
// external scripts are not using it to mine data on group members
// so let's validate the HMAC that was generated by the userpicker input
$group_guid = (int) get_input('group_guid');

$data = [
    'group_guid' => $group_guid,
];

// let's sort by key, in case we have more elements

```

(suite sur la page suivante)

(suite de la page précédente)

```

ksort($data);

$hmac = elgg_build_hmac($data);
if (!$hmac->matchesToken(get_input('mac'))) {
    // request does not originate from our input view
    forward('', '403');
}

elgg_set_http_header("Content-Type: application/json;charset=utf-8");

$options = [
    'query' => $query,
    'type' => 'user',
    'limit' => $limit,
    'sort' => 'name',
    'order' => 'ASC',
    'fields' => [
        'metadata' => ['name', 'username'],
    ],
    'item_view' => 'search/entity',
    'input_name' => $input_name,
    'wheres' => function (QueryBuilder $qb) use ($group_guid) {
        $subquery = $qb->subquery('entity_relationships', 'er');
        $subquery->select('1')
            ->where($qb->compare('er.guid_one', '=', 'e.guid'))
            ->andWhere($qb->compare('er.relationship', '=', 'member', ELGG_VALUE_STRING))
            ->andWhere($qb->compare('er.guid_two', '=', $group_guid, ELGG_VALUE_
↳ INTEGER));

        return "NOT EXISTS ({ $subquery->getSQL() })";
    }
];

echo elgg_list_entities($options, 'elgg_search');
```

3.3.27 Services

Elgg utilise la classe `Elgg\Application` pour charger et initialiser Elgg. Dans les versions futures cette classe offrira un ensemble d'objets de service à l'usage des plugins.

Note : Si vous avez une idée utile, vous pouvez ajouter un nouveau service !

Menus

`elgg()` -> `menus` fournit des méthodes de bas niveau pour la construction de menus. En général, les menus doivent être transmis à `elgg_view_menu` pour le rendu, plutôt qu'un rendu manuel.

3.3.28 Paramètres du plugin

Vous devez effectuer quelques étapes supplémentaires si votre plugin a besoin de paramètres qui doivent être enregistrés et contrôlés via le panneau d'administration :

- Créez un fichier dans le dossier des vues par défaut de votre plugin appelé `plugins/votre_plugin/settings.php`, où `votre_plugin` est le nom du répertoire de votre plugin dans la hiérarchie `mod`
- Remplissez ce fichier avec les éléments de formulaire que vous souhaitez afficher avec les *traductions des* étiquettes de texte
- Définissez l'attribut de nom dans vos composants de formulaire sur `params['varname']` où `varname` est le nom de la variable. Celles-ci seront enregistrées en tant que paramètres privés attachés à une entité plugin. Ainsi, si votre variable est appelée `params[monparametre]` votre plugin (qui est également passé à cette vue comme `$vars['entity']`) sera appelé `$vars['entity']->monparametre`

Un exemple de `settings.php` ressemblerait à :

```
<p>
  <?php echo elgg_echo('myplugin:settings:limit'); ?>

  <select name="params[limit]">
    <option value="5" <?php if ($vars['entity']->limit == 5) echo " selected=\"yes\" ";
    <?>5</option>
    <option value="8" <?php if ((!$vars['entity']->limit) || ($vars['entity']->limit_
    <?>8)) echo " selected=\"yes\" "; ?>8</option>
    <option value="12" <?php if ($vars['entity']->limit == 12) echo " selected=\"yes\"
    <?>12</option>
    <option value="15" <?php if ($vars['entity']->limit == 15) echo " selected=\"yes\"
    <?>15</option>
  </select>
</p>
```

Note : Vous n'avez pas besoin d'ajouter un bouton d'enregistrement ou le formulaire, cela sera géré par le framework.

Note : Vous ne pouvez pas utiliser des composants de formulaire qui n'envoient aucune valeur lorsqu'ils sont « off ». Il s'agit notamment des entrées radio et des cases à cocher.

Si vos paramètres de plugin ont besoin d'un vidage des caches, vous pouvez ajouter une entrée (cachée) sur le formulaire avec le nom "flush_cache" et la valeur "1"

```
elgg_view_field([
    '#type' => 'hidden',
    'name' => 'flush_cache',
    'value' => 1,
]);
```


Paramètres de l'utilisateur

Votre plugin peut également avoir besoin de stocker des paramètres pour chaque utilisateur, et vous voudrez voir les options de votre plugin apparaître dans la page de paramètres de l'utilisateur. C'est également facile à faire et suit le même modèle que la configuration globale du plugin expliquée plus tôt. La seule différence est qu'au lieu d'utiliser un fichier `settings`, vous utiliserez `usersettings`. Ainsi, le chemin d'accès à la vue de modification des paramètres utilisateur de votre plugin serait `plugins/votre_plugin/usersettings.php`.

Note : Le titre du formulaire des paramètres utilisateur sera par défaut au nom du plugin. Si vous souhaitez modifier cela, ajoutez une traduction pour `plugin_id:usersettings:title`.

Récupérer des paramètres dans votre code

Pour récupérer les paramètres depuis votre code utilisez :

```
$setting = elgg_get_plugin_setting($name, $plugin_id);
```

ou pour les paramètres utilisateur

```
$user_setting = elgg_get_plugin_user_setting($name, $user_guid, $plugin_id);
```

où :

- `$name` Est la valeur que vous souhaitez récupérer
- `$user_guid` Est l'utilisateur pour lequel vous souhaitez récupérer ces valeurs (par défaut l'utilisateur actuellement connecté)
- `$plugin_name` Est le nom du plugin (détection s'il s'exécute à partir d'un plugin)

Définir des valeurs via le code

Les valeurs peuvent également être définies à partir du code de votre plugin, pour cela utilisez l'une des fonctions suivantes :

```
elgg_set_plugin_setting($name, $value, $plugin_id);
```

ou

```
elgg_set_plugin_user_setting($name, $value, $user_guid, $plugin_id);
```

Avertissement : Le `$plugin_id` doit être fourni lors de la définition des paramètres du plugin (ou de l'utilisateur).

Paramètres par défaut du plugin (ou de l'utilisateur)

Si un plugin ou un utilisateur n'a pas de paramètre stocké dans la base de données, vous avez parfois besoin d'une certaine valeur par défaut. Vous pouvez passer cela lors de l'utilisation des fonctions de récupération (getter).

```
$user_setting = elgg_get_plugin_user_setting($name, $user_guid, $plugin_id, $default);

$plugin_setting = elgg_get_plugin_setting($name, $plugin_id, $default);
```

Vous pouvez également fournir des paramètres de plugin et d'utilisateur par défaut dans le fichier `elgg-plugin.php`.

```
<?php

return [
    'settings' => [
        'key' => 'value',
    ],
    'user_settings' => [
        'key' => 'value',
    ],
];
```

3.3.29 Thèmes

Personnaliser l'apparence et le comportement d'Elgg.

Un thème est un type de *plugin* qui surcharge des aspects d'affichage d'Elgg.

Ce guide suppose que vous connaissez :

- *Plugins*
- *Vues*

Contenu

- *Principes de création d'un thème et Bonnes Pratiques*
- *Créez votre plugin*
- *Personnalisez les CSS*
 - *Variables CSS*
 - *Extension d'une vue*
 - *Surcharge des vues*
 - *Icônes*
- *Outils*
- *Personnalisation de la page d'accueil*

Principes de création d'un thème et Bonnes Pratiques

Pas de framework CSS tierce-partie Elgg n'utilise pas de framework CSS, car ces frameworks enferment les utilisateurs dans un balisage HTML spécifique, ce qui rend finalement beaucoup plus difficile pour les plugins de collaborer sur l'apparence. Ce qui est *is-primary* dans un thème pourrait être quelque chose d'autre dans un autre. L'absence de framework permet aux plugins de modifier l'apparence à l'aide de css pur, sans avoir à remplacer les vues et ajouter des sélecteurs spécifiques à un framework donné aux éléments de balisage HTML.

```
/* BAD */
<div class="box has-shadow is-inline">
    This is bad, because if the plugin wants to change the styling, it will have to
    ↪ either write really specific css
    ↪ clearing all the attached styles, or replace the view entirely just to modify
    ↪ the markup
</div>

/* GOOD */
<div class="box-role">
    This is good, because a plugin can just simply add .box-role rule
</div>
<style>
    .box-role {
        padding: 1rem;
        display: inline-block;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    }
</style>
```

système de grille 8-point Elgg utilise un *système de grille à 8 points* <<https://builttoadapt.io/intro-to-the-8-point-grid-system-d2573cde8632>>, aussi le dimensionnement des éléments, leurs marges internes et externes, etc se font par incréments et fractions de 8px. Étant donné que notre taille de police par défaut est 16px, nous utilisons des fractions de *rem*, de sorte que $0.5\text{rem} = 8\text{px}$. Le système de grille à 8 points permet aux développeurs de collaborer beaucoup plus facilement sur des éléments de style : cela évite de se demander la marge interne doit être de 5px ou 6px.

```
/* BAD */
.menu > li {
    margin: 2px 2px 2px 0;
}

.menu > li > a {
    padding: 3px 5px;
}

/* GOOD */
.menu > li > a {
    padding: 0.25rem 0.5rem;
}
```

Mobile d'abord Nous écrivons un CSS mobile-first. Nous utilisons deux points de rupture : 50rem et 80rem (800px et 1280px à 16px/rem).

```
/* BAD: mobile defined in media blocks, different display types */

.menu > li {
    display: inline-block;
}
@media screen and (max-width: 820px) {
    .menu > li {
        display: block;
        width: 100%;
    }
}

/* GOOD: mobile by default. Media blocks style larger viewports. */

.menu {
    display: flex;
    flex-direction: column;
}
@media screen and (min-width: 50rem) {
    .menu {
        flex-direction: row;
    }
}
```

conduit par Flexbox Flexbox offre une simplicité dans l’empilement des éléments dans les grilles. Flexbox est utilisé pour tout, des menus aux éléments de mise en page. Nous évitons les `float` et les `clearfix` car ils ont du mal à collaborer et ouvrent beaucoup de place pour les erreurs et les distorsions.

```
/* BAD */
.heading:after {
    visibility: hidden;
    height: 0;
    clear: both;
    content: " ";
}
.heading > h2 {
    float: left;
}
.heading > .controls {
    float: right;
}

/* GOOD */
.heading {
    display: flex;
    justify-content: flex-end;
}
.heading > h2 {
    order: 1;
    margin-right: auto;
}
.heading > .controls {
```

(suite sur la page suivante)

(suite de la page précédente)

```

    order: 2;
}

```

Symétrique Nous maintenons la symétrie.

```

/* BAD */
.row .column:first-child {
    margin-right: 10px;
}

/* GOOD */
.row {
    margin: 0 -0.5rem;
}
.row .column {
    margin: 0.5rem;
}

```

Des transitions de couleur simples Nous maintenons 4 ensembles de couleurs pour le texte, l'arrière-plan et la bordure : **soft** (doux), **mild** (modéré), **strong** (fort) et **highlight** (en exergue). Lors de la transition vers l'état **hover** ou **active**, nous passons d'un niveau vers le haut, par exemple de **soft** à **mild**, ou utilisons **highlight**. Lorsque nous passons à un état inactif ou désactivé, nous descendons d'un niveau.

Augmentez la zone de clic En travaillant avec des ancres imbriquées, nous augmentons la zone de clic de l'ancre, plutôt que le parent

```

/* BAD */
.menu > li {
    margin: 5px;
    padding: 5px 10px;
}

/* GOOD */
.menu > li {
    margin: 0.5rem;
}
.menu > li > a {
    padding: 0.5rem 1rem;
}

```

Pas de z-index 999999 les z-index sont incrémentés par pas de 1.

Envelopper les frères et sœurs HTML Nous nous assurons qu'il n'y a pas de chaîne orpheline au sein d'un parent et que les frères et sœurs sont enveloppés d'une manière qui permet de les cibler par CSS.

```

/* BAD */
<label>
    Orphan
    <span>Sibling</span>
</label>

```

(suite sur la page suivante)

(suite de la page précédente)

```
/* GOOD */
<label>
    <span>Sibling</span>
    <span>Sibling</span>
</label>
```

```
/* BAD */
<div>
    <h3>Title</h3>
    <p>Subtitle</p>
    <div class="right">This goes to the right</div>
</div>

/* GOOD */
<div>
    <div class="left">
        <h3>Title</h3>
        <p>Subtitle</p>
    </div>
    <div class="right">This goes to the right</div>
</div>
```

Créez votre plugin

Créez votre plugin tel que décrit dans le guide de développement *developer guide*.

- Créez un nouveau dossier dans mod/
- Créez un nouveau fichier start.php
- Créez un fichier manifest.xml décrivant votre thème.

Personnalisez les CSS

Le css est découpé en plusieurs fichiers en fonction des aspects du site que vous êtes en train de styler. Cela vous permet de les aborder un à la fois, vous donnant une chance de faire de réelles améliorations sans être submergé.

Voici la liste des vues CSS existantes :

- elements/buttons.css : fournit un moyen de styler tous les différents types de boutons que votre site utilise. Il existe 5 types de boutons que les plugins s'attendent à être disponibles : action, cancel (annuler), delete (supprimer), submit (soumettre), et special.
- elements/chrome.css : ce fichier comporte des classes de présentation diverses.
- elements/components.css : ce fichier contient de nombreux « objets css » qui sont utilisés sur tout le site : media block (bloc multimédia), list, gallery, table (tableau), owner block (propriétaire), system, messages, river, tags, photo et comments.
- elements/forms.css : ce fichier détermine à quoi ressembleront vos formulaires et éléments de saisie.
- elements/icons.css : contient des styles pour les icônes et avatars utilisés sur votre site.
- elements/layout.css : détermine à quoi ressemblera votre page : barres latérales, emballage de la page, corps principal, en-tête, pied de page, etc.
- elements/modules.css : Beaucoup de contenu dans Elgg s'affiche dans des boîtes avec un titre et un corps de contenu. Nous avons appelé ces boîtes des modules. Il en existe de quelques sortes : info, aside (sur le côté), featured (en vedette), dropdown (déroulante), popup, widget. Les styles des widgets sont également inclus dans ce fichier, car ils sont un sous-ensemble des modules.
- elements/navigation.css : ce fichier détermine à quoi ressembleront tous vos menus.

- `elements/typography.css` : ce fichier détermine à quoi ressemblera le contenu et les titres de votre site.
- `rtl.css` : des règles personnalisées pour les utilisateurs qui consultent votre site dans une langue de droite à gauche.
- `admin.css` : un thème totalement distinct pour la zone d'administration (généralement pas remplacé).
- `elgg.css` : compile tous les fichiers des éléments de base/* dans un seul fichier (NE PAS REMPLACER).
- `éléments/core.css` : contient des styles de base pour les « objets css » les plus compliqués. Si vous vous trouvez en situation de vouloir remplacer cela, vous avez probablement besoin de signaler un bogue du noyau Elgg à la place (NE PAS REMPLACER).
- `éléments/reset.css` : contient une feuille de style de réinitialisation qui force les éléments à avoir le même rendu par défaut

Variables CSS

Elgg utilise CssCrush pour prétraiter les fichiers CSS. Cela nous donne la flexibilité d'utiliser des variables CSS globales. Les plugins devraient, dans la mesure du possible, utiliser des variables CSS globales et étendre le thème de base avec leurs variables plugin, afin qu'ils puissent être simplement modifiés par d'autres plugins.

Pour ajouter ou modifier des variables, utilisez le hook `vars:compiler, css`. Notez que vous devrez peut-être vider le cache pour voir vos modifications en action.

Pour obtenir une liste des variables par défaut du noyau, voir `engine/theme.php`.

Extension d'une vue

Il existe deux manières de modifier les vues :

La première façon est d'ajouter des choses à une vue existante est via la fonction d'extension de vue, à partir de la fonction d'initialisation de votre `start.php`.

Par exemple, le `start.php` suivant ajoutera `mytheme/css` au css de base d'Elgg :

```
<?php

function mytheme_init() {
    elgg_extend_view('elgg.css', 'mytheme/css');
}

elgg_register_event_handler('init', 'system', 'mytheme_init');

?>
```

Surcharge des vues

Les plugins peuvent avoir une hiérarchie d'affichage, n'importe quel fichier qui existe ici remplacera tous les fichiers dans la hiérarchie de vue de base existante... par exemple, si mon plugin a un fichier :

`/mod/myplugin/views/default/elements/typography.css`

ceci va remplacer :

`/views/default/elements/typography.css`

Mais seulement si le plugin est actif.

Cela vous donne un contrôle total sur l'apparence et le comportement de Elgg. Cela vous offre la possibilité de modifier légèrement ou de remplacer totalement les vues existantes.

Icônes

À partir d’Elgg 2.0, les icônes Elgg par défaut proviennent de la bibliothèque [FontAwesome](#). Vous pouvez utiliser l’une de ces icônes en appelant :

```
elgg_view_icon('icon-name');
```

icon-name peut être n’importe laquelle des [icônes de FontAwesome](#) sans le préfixe “fa-”.

Par défaut, vous obtiendrez la variante de style solid des icônes. En ajoutant au nom de l’icône l’un des suffixes -solid, -regular ou -light, vous pouvez cibler un style spécifique. Soyez avisé ; la variante de style light n’est disponible qu’en tant qu’icône sous licence FontAwesome Pro.

Outils

Nous vous avons fourni quelques outils de développement pour vous aider à mettre en place des thèmes : activez le plugin “Developers” et accédez à la page “Theme Preview” (aperçu du thème) pour commencer à suivre les progrès de votre thème.

Personnalisation de la page d’accueil

La page principale de Elgg exécute un hook de plugin appelé “index,system”. Si ce hook renvoie true, il suppose qu’une autre page gérant la page d’accueil a déjà été générée, et n’affiche pas la page par défaut.

Par conséquent, vous pouvez remplacer la page d’accueil en enregistrant une fonction sur le hook de plugin “index,system” et en renvoyant true à partir de cette fonction.

Voici un aperçu rapide :

- Créez votre nouveau plugin
- Dans le start.php, vous aurez besoin de quelque chose comme ce qui suit :

```
<?php

function pluginname_init() {
    // Replace the default index page
    elgg_register_plugin_hook_handler('index', 'system', 'new_index');
}

function new_index() {
    if (!include_once(dirname(dirname(__FILE__)) . "/pluginname/pages/index.php"))
        return false;

    return true;
}

// register for the init, system event when our plugin start.php is loaded
elgg_register_event_handler('init', 'system', 'pluginname_init');
?>
```

- Puis créez une page d’index (/nom_plugin/pages/index.php) et utilisez-la pour intégrer du contenu comme vous le souhaitez sur la page d’accueil de votre site Elgg.

3.3.30 Écrire une mise à niveau de plugin

De temps en temps, il arrive un moment où un plugin doit modifier le contenu ou la structure des données qu'il a stockées soit dans la base de données, soit dans le répertoire de données.

La raison de ceci peut être que la structure des données doit être convertie en une structure plus efficace ou plus flexible. Ou peut-être en raison d'un bogue les éléments de données ont été enregistrés d'une manière non valide, et ils doivent être convertis dans un format correct.

Les migrations et conversions comme celle-ci peuvent prendre beaucoup de temps s'il y a beaucoup de données à traiter. C'est pourquoi Elgg fournit l'interface `Elgg\Upgrade\Batch` qui peut être utilisée pour implémenter des mises à niveau de longue durée.

Déclarer la mise à niveau d'un plugin

Un plugin peut signaler la nécessité d'une mise à niveau via la clef `upgrades` dans le fichier `elgg-plugin.php`. Chaque valeur du tableau doit être le nom entièrement qualifié d'une classe de mise à niveau qui implémente l'interface `Elgg\Upgrade\Batch`.

Exemple tiré du fichier `mod/blog/elgg-plugin.php` :

```
return [
    'upgrades' => [
        Blog\Upgrades\AccessLevelFix::class,
        Blog\Upgrades\DraftStatusUpgrade::class,
    ]
];
```

Les noms de classe dans l'exemple se réfèrent aux classes :

- `mod/blog/classes/Blog/Upgrades/AccessLevelFix`
- `mod/blog/classes/Blog/Upgrades/DraftStatusUpgrade`

Note : Les classes de mise à niveau du noyau Elgg peuvent être déclarées dans `engine/lib/upgrades/async-upgrades.php`.

La classe de mise à niveau

Une classe implémentant l'interface `Elgg\Upgrade\Batch` a beaucoup de liberté sur la façon dont elle veut gérer le traitement réel des données. Elle doit cependant déclarer certaines variables constantes et également prendre soin de marquer si chaque élément traité a bien été mis à niveau ou non.

La structure de base de la classe est la suivante :

```
<?php

namespace Blog\Upgrades;

use Elgg\Upgrade\Batch;
use Elgg\Upgrade\Result;

/**
 * Fixes invalid blog access values
```

(suite sur la page suivante)

```
*/
class AccessLevelFix implements Batch {

    /**
     * Version of the upgrade
     *
     * @return int
     */
    public function getVersion() {
        return 2016120300;
    }

    /**
     * Should the run() method receive an offset representing all processed items?
     *
     * @return bool
     */
    public function needsIncrementOffset() {
        return true;
    }

    /**
     * Should this upgrade be skipped?
     *
     * @return bool
     */
    public function shouldBeSkipped() {
        return false;
    }

    /**
     * The total number of items to process in the upgrade
     *
     * @return int
     */
    public function countItems() {
        // return count of all blogs
    }

    /**
     * Runs upgrade on a single batch of items
     *
     * @param Result $result Result of the batch (this must be returned)
     * @param int $offset Number to skip when processing
     *
     * @return Result Instance of \Elgg\Upgrade\Result
     */
    public function run(Result $result, $offset) {
        // fix 50 blogs skipping the first $offset
    }
}
```

Avertissement : Ne présumez pas quand votre classe sera instanciée ou quand/combien de fois ses méthodes publiques seront appelées.

Méthodes de la classe

getVersion()

Cela doit renvoyer un entier représentant la date à laquelle la mise à niveau a été ajoutée. La valeur se compose de huit chiffres au format `yyyymmddnn` où :

- `yyyy` représente l'année
- `mm` représente le mois (avec le zéro initial)
- `dd` représente le jour (avec le zéro initial)
- `nn` est un numéro d'incrément (à partir de `00`) qui est utilisé au cas où deux mises à niveau distinctes auraient été ajoutées au cours de la même journée

shouldBeSkipped()

Cela devrait renvoyer `false` à moins que la mise à niveau ne sera pas nécessaire.

Avertissement : Si `true` est renvoyé, la mise à niveau ne peut pas être exécutée plus tard.

needsIncrementOffset()

Si `true`, votre méthode `run()` recevra comme `$offset` le nombre d'éléments traités. Ceci est utile si vous ne modifiez que des données et que vous devez utiliser le `$offset` dans une fonction comme `elgg_get_entities()` pour savoir combien d'éléments ont déjà été traités.

Si `false`, votre méthode `run()` recevra comme `$offset` le nombre total d'échecs. `false` doit être utilisé si votre processus supprime ou déplace des données en cours de processus. Par exemple, si vous supprimez 50 objets sur chaque `run()`, vous n'avez pas vraiment besoin de `$offset`.

countItems()

Récupère le nombre total d'éléments à traiter pendant la mise à niveau. Si ce nombre est inconnu, `Batch::UNKNOWN_COUNT` peut être retourné, mais `run()` doit alors marquer manuellement la mise à niveau terminée.

run()

Cette fonction doit effectuer une partie de la mise à niveau réelle. Et selon le temps qu'il faut, elle peut être appelée plusieurs fois au cours d'une seule demande.

Il reçoit deux arguments :

- `$result` : Une instance de l'objet `Elgg\Upgrade\Result`
- `$offset` : Le décalage à partir duquel la prochaine partie de mise à niveau doit commencer (ou le nombre total de défaillances)

Pour chaque élément que la méthode traite, elle doit appeler selon les cas :

- `$result->addSuccesses()` : Si l'élément a bien été mis à niveau
- `$result->addFailures()` : Si l'élément n'a pas pu être mis à niveau

Les deux méthodes ont par défaut un élément, mais vous pouvez éventuellement passer le nombre d'éléments.

En outre, elle peut définir autant de messages d'erreur que nécessaire au cas où quelque chose ne va pas :

- `$result->addError("Le message d'erreur est ici")`

Si `countItems()` renvoie `Batch::UNKNOWN_COUNT`, alors à un moment donné `run()` doit appeler `$result->markComplete()` pour terminer la mise à niveau.

Dans la plupart des cas, votre méthode `run()` voudra passer le paramètre `$offset` à l'une des fonctions `elgg_get_entities()` :

```
/**
 * Process blog posts
 *
 * @param Result $result The batch result (will be modified and returned)
 * @param int $offset Starting point of the batch
 * @return Result Instance of \Elgg\Upgrade\Result;
 */
public function run(Result $result, $offset) {
    $blogs = elgg_get_entities([
        'type' => 'object'
        'subtype' => 'blog'
        'offset' => $offset,
    ]);

    foreach ($blogs as $blog) {
        if ($this->fixBlogPost($blog)) {
            $result->addSuccesses();
        } else {
            $result->addFailures();
            $result->addError("Failed to fix the blog {$blog->guid}.");
        }
    }

    return $result;
}
```

Interface d'administration

Chaque mise à niveau implémentant l'interface Elgg\Upgrade\Batch est répertoriée dans le panneau d'administration après avoir déclenché la mise à niveau du site à partir du tableau de bord Administration.

Lors de l'exécution des mises à niveau Elgg fournit :

- Durée estimée de la mise à niveau
- Nombre d'éléments traités
- Nombre d'erreurs
- Messages d'erreur possibles

3.3.31 Vues

Contenu

- *Introduction*
- *Utiliser les vues*
- *Les vues comme modèles (templates)*
- *Les vues en tant qu'actifs pouvant être mis en cache*
- *Vues et actifs tierce-partie (assets)*
 - *En spécifiant des répertoires supplémentaires pour les vues*
- *Types de vues (viewtypes)*
- *Modification des vues via des plugins*
 - *Surcharger les vues (override)*
 - *Étendre les vues*
 - *Modification de l'entrée d'une vue*
 - *Modification de la sortie d'une vue*
 - *Remplacement complet du résultat de la vue*
- *Afficher des entités*
 - *Vues des entités partielles et complètes*
- *Lister des entités*
 - *Rendu d'une liste avec une autre vue*
 - *Rendu d'une liste en tant que tableau*
- *Icônes*
 - *Icônes génériques*
 - *Icônes des entités*
- *Connexe*

Introduction

Les vues sont responsables de la création des sorties. Elles gèrent tout depuis :

- la disposition des pages
- des morceaux de sortie d'affichage (comme un pied de page ou une barre d'outils)
- des liens individuels et des entrées de formulaire.
- des images, js, et css nécessaires pour votre page web

Utiliser les vues

À leur niveau le plus élémentaire, les vues par défaut ne sont que des fichiers PHP avec des extraits de html :

```
<h1>Hello, World!</h1>
```

En supposant que cette vue se trouve dans `/views/default/hello.php`, nous pourrions l'afficher comme ceci :

```
echo elgg_view('hello');
```

Pour votre commodité, Elgg est livré avec beaucoup de vues par défaut. Afin de garder les choses gérables, elles sont organisées en sous-répertoires. Elgg gère très bien cette organisation. Par exemple, notre vue simple peut être située dans `/views/default/hello/world.php`, auquel cas elle sera appelée comme ceci :

```
echo elgg_view('hello/world');
```

Le nom de la vue reflète simplement l'emplacement de la vue dans le répertoire des vues.

Les vues comme modèles (templates)

Vous pouvez transmettre des données arbitraires à une vue via le tableau `$vars`. Notre vue `hello/world` pourrait être modifié pour accepter une variable comme ceci :

```
<h1>Hello, <?= $vars['name']; ?>!</h1>
```

Dans ce cas, nous pouvons passer un paramètre de nom arbitraire à la vue comme ceci :

```
echo elgg_view('hello/world', ['name' => 'World']);
```

qui produirait la sortie suivante :

```
<h1>Hello, World!</h1>
```

Avertissement : Les vues ne font aucune sorte d'assainissement automatique de la sortie par défaut. Vous êtes responsable de faire l'assainissement correct vous-même pour prévenir les attaques XSS et similaires.

Les vues en tant qu'actifs pouvant être mis en cache

Comme mentionné précédemment, les vues peuvent contenir du JS, des CSS, ou même des images.

Les actifs doivent correspondre à certaines exigences :

- Elles *ne doivent pas* utiliser de paramètre de `$vars`
- Elles *ne doivent pas* modifier leur sortie en fonction de l'état global comme
 - qui est connecté
 - l'heure de la journée
- Elles *doivent* avoir une extension de fichier valide
 - Mauvais : `my/cool/template`
 - Bon : `my/cool/template.html`

Par exemple, supposons que vous vouliez charger certains CSS sur une page. Vous pouvez définir une vue `messtyles.css`, qui ressemblerait à ceci :

```
/* /views/default/mystyles.css */
.mystyles-foo {
    background: red;
}
```

Note : Omettez l'extension « .php » du nom de fichier et Elgg reconnaîtra automatiquement la vue comme pouvant être mise en cache.

Pour obtenir une URL de ce fichier, vous utiliseriez `elgg_get_simplecache_url` :

```
// Returns "https://mysite.com/.../289124335/default/mystyles.css"
elgg_get_simplecache_url('mystyles.css');
```

Elgg ajoute automatiquement les numéros magiques que vous voyez là pour le cache-busting et définit les entêtes Expires à long terme sur le fichier renvoyé.

Avvertissement : Elgg pourrait décider de modifier l'emplacement ou la structure de l'URL retournée dans une version ultérieure pour une raison quelconque, et les numéros d'invalidation du cache (cache-busting) changent à chaque fois que vous videz les caches d'Elgg, de sorte que l'URL exacte n'est pas stable par design.

Avec cela à l'esprit, voici quelques anti-modèles à éviter :

- Ne vous fiez pas à la structure/emplacement exacts de cette URL
- N'essayez pas de générer les URL vous-même
- Ne stockez pas les URL renvoyées dans une base de données

Sur la page où vous souhaitez charger le css, appelez :

```
elgg_require_css('mystyles');
```

Vues et actifs tierce-partie (assets)

La meilleure manière de servir des actifs tiers est par le biais des vues. Toutefois, au lieu de copier/coller manuellement les actifs au bon endroit dans `/views/*`, vous pouvez mapper les actifs dans le système de vues via la clef "views" dans le fichier de configuration `elgg-plugin.php` de votre plugin.

La valeur des vues doit être un tableau à 2 dimensions. Le premier niveau mappe un type de vue à une liste de mappage de vues. Le deuxième niveau associe les noms des vues à des chemins de fichiers, absolus ou relatifs au répertoire racine de Elgg.

Si vous ajoutez vos actifs au suivi de version, pointez-les comme ceci :

```
<?php // mod/example/elgg-plugin.php
return [
    // view mappings
    'views' => [
        // viewtype
        'default' => [
            // view => /path/from/filesystem/root
            'js/jquery-ui.js' => __DIR__ . '/bower_components/jquery-ui/jquery-ui.min.js'
        ],
    ],
];
```

(suite sur la page suivante)

(suite de la page précédente)

```
    ],  
];
```

Pour pointer vers les actifs installés avec composer, utilisez les chemins install-root-relative en omettant le slash initial :

```
<?php // mod/example/elgg-plugin.php  
return [  
    'views' => [  
        'default' => [  
            // view => path/from/install/root  
            'js/jquery-ui.js' => 'vendor/bower-asset/jquery-ui/jquery-ui.min.js',  
        ],  
    ],  
];
```

Le noyau de Elgg utilise cette fonctionnalité largement, bien que la valeur soit retournée directement à partir de `/engine/views.php`.

Note : Vous n'avez pas besoin d'utiliser Bower, Composer Asset Plugin, ou tout autre script pour gérer les actifs de votre plugin, mais nous vous recommandons fortement d'utiliser un gestionnaire de package, car cela rend la mise à niveau tellement plus facile.

En spécifiant des répertoires supplémentaires pour les vues

Dans `elgg-plugin.php` vous pouvez également spécifier des répertoires à scanner pour les vues. Il suffit de fournir un préfixe de nom de vue se terminant par `/` et un chemin de répertoire (comme ci-dessus).

```
<?php // mod/file/elgg-plugin.php  
return [  
    'views' => [  
        'default' => [  
            'file/icon/' => __DIR__ . '/graphics/icons',  
        ],  
    ],  
];
```

Avec ce qui précède, les fichiers trouvés dans le dossier `icons` seront interprétés comme des vues. Par exemple, l'affichage `file/icon/general.gif` sera créé et mappé à `mod/file/graphics/icons/general.gif`.

Note : Il s'agit d'un scan entièrement récursif. Tous les fichiers trouvés seront introduits dans le système de vues.

Plusieurs chemins peuvent partager le même préfixe, il suffit de donner un tableau de chemins :

```
<?php // mod/file/elgg-plugin.php  
return [  
    'views' => [  
        'default' => [  
            'file/icon/' => [  
                __DIR__ . '/graphics/icons',  
            ],  
        ],  
    ],  
];
```

(suite sur la page suivante)

(suite de la page précédente)

```

        __DIR__ . '/more_icons', // processed 2nd (may override)
    ],
],
];

```

Types de vues (viewtypes)

Vous pouvez vous demander : « Pourquoi `/views/default/hello/world.php` au lieu de simplement `/views/hello/world.php` ? ».

Le sous-répertoire dans `/views` détermine le *viewtype* des vues qu'il contient. Un type de vue correspond généralement au format de sortie des vues.

Le type de vue par défaut est supposé être HTML et d'autres actifs statiques nécessaires pour rendre une page Web réactive dans un navigateur de bureau ou mobile, mais il peut également être :

- RSS
- ATOM
- JSON
- HTML optimisé pour mobile
- HTML optimisé pour TV
- N'importe quel nombre d'autres formats de données

Vous pouvez forcer Elgg à utiliser un type de vue particulier pour afficher la page en définissant la variable d'entrée `view` comme ceci : `https://monsite.com/?view=rss`.

Vous pourriez également écrire un plugin pour définir ceci automatiquement à l'aide de la fonction `elgg_set_viewtype()`. Par exemple, votre plugin peut détecter que la page a été consultée avec la chaîne de navigateur d'un iPhone, et définir le type de vue sur `iphone` en appelant :

```
elgg_set_viewtype('iphone');
```

Le plugin fournirait probablement également un ensemble de vues optimisées pour ces appareils.

Modification des vues via des plugins

Sans modifier le noyau d'Elgg, Elgg offre plusieurs manières de personnaliser quasiment toutes les sorties :

- Vous pouvez *remplacer une vue*, en modifiant complètement le fichier utilisé pour l'afficher.
- Vous pouvez *étendre une vue* en y insérant la sortie d'une autre vue avant ou après.
- Vous pouvez *modifier les entrées d'une vue* avec un hook plugin.
- Vous pouvez *modifier la sortie d'une vue* avec un hook plugin.

Surcharger les vues (override)

Les vues dans les répertoires des plugins remplacent toujours les vues du répertoire du noyau ; toutefois, quand les plugins remplacent les vues d'autres plugins, les vues des derniers plugins remplacent celles de précédents.

Par exemple, si nous voulions personnaliser la vue `hello/world` pour utiliser un `h2` au lieu d'un `h1`, nous pourrions créer un fichier à `mod/exemple/views/default/hello/world.php` comme ceci :

```
<h2>Hello, <?= $vars['name']; ?></h2>
```

Note : Quand vous considérez la maintenance à long terme, la surcharge des vues du noyau et des plugins groupés a un coût : les mises à niveau peuvent apporter des changements dans les vues, et si vous les avez remplacés, vous n'obtiendrez pas ces changements ou devrez les réintégrer dans vos vues.

Vous pouvez plutôt souhaiter modifier l'*entrée* ou :ref:{la sortie <guides/views#altering-view-output>` de la vue via des hooks plugin.

Note : Elgg met en cache les chemins des vues. Cela signifie que vous devez désactiver le cache système quand vous développez avec des vues. Lorsque vous installez les modifications apportées dans un environnement de production, vous devez vider les caches.

Étendre les vues

Il peut y avoir d'autres situations dans lesquelles vous ne voulez pas remplacer l'ensemble de la vue, et plutôt y ajouter un peu plus de contenu au début ou à la fin. Dans Elgg, c'est ce qu'on appelle *étendre une vue*.

Par exemple, au lieu de remplacer la vue `hello/world`, nous pourrions l'étendre comme ceci :

```
elgg_extend_view('hello/world', 'hello/greeting');
```

Si le contenu de `/views/default/hello/greeting.php` est :

```
<h2>How are you today?</h2>
```

Ensuite, chaque fois que nous appelons `elgg_view('hello/world');`, nous obtiendrons :

```
<h1>Hello, World!</h1>
<h2>How are you today?</h2>
```

Vous pouvez ajouter des choses avant la vue en transmettant au 3ème paramètre une valeur inférieure à 500 :

```
// appends 'hello/greeting' to every occurrence of 'hello/world'
elgg_extend_view('hello/world', 'hello/greeting');

// prepends 'hello/greeting' to every occurrence of 'hello/world'
elgg_extend_view('hello/world', 'hello/greeting', 450);
```

Toutes les extensions de vues doivent être enregistrées dans le gestionnaire d'événements `init`, `system` de votre plugin dans `start.php`.

Modification de l'entrée d'une vue

Il peut être utile de modifier le tableau `$vars` d'une vue avant que la vue ne soit rendue.

Avant chaque affichage de la vue, le tableau `$vars` est filtré par le *hook de plugin* `["view_vars", $view_name]`. Chaque fonction de gestionnaire enregistrée dispose de ces arguments :

- `$hook` - la chaîne "view_vars"
- `$view_name` - le nom de la vue (le premier argument est passé à `elgg_view()`)
- `$returnvalue` - le tableau modifié `$vars`
- `$params` - un tableau contenant :
 - `vars` - le tableau original `$vars`, inchangé

- `view` - le nom de la vue
- `viewtype` - Le *type de vue* en train d'être affiché

Exemple de modification des entrées d'une vue

Ici, nous allons modifier la limite de pagination par défaut pour la vue des commentaires :

```
elgg_register_plugin_hook_handler('view_vars', 'page/elements/comments', 'myplugin_alter_
↳comments_limit');

function myplugin_alter_comments_limit($hook, $type, $vars, $params) {
    // only 10 comments per page
    $vars['limit'] = elgg_extract('limit', $vars, 10);
    return $vars;
}
```

Modification de la sortie d'une vue

Parfois, il est préférable de modifier la sortie d'une vue au lieu de la remplacer.

La sortie de chaque vue passe par le *hook plugin* ["view", \$view_name] avant d'être renvoyé par `elgg_view()`. Chaque fonction de gestionnaire enregistrée dispose de ces arguments :

- `$hook` - la chaîne "view"
- `$view_name` - le nom de la vue (le premier argument est passé à `elgg_view()`)
- `$result` - la sortie modifiée de la vue
- `$params` - un tableau contenant :
 - `viewtype` - Le *type de vue* en train d'être affiché

Pour modifier le résultat de la vue, le gestionnaire n'a qu'à modifier `$returnvalue` et à renvoyer une nouvelle chaîne.

Exemple de modification d'affichage d'une vue

Ici, nous allons éliminer le fil d'Ariane lorsqu'il ne contient aucun lien.

```
elgg_register_plugin_hook_handler('view', 'navigation/breadcrumbs', 'myplugin_alter_
↳breadcrumb');

function myplugin_alter_breadcrumb($hook, $type, $returnvalue, $params) {
    // we only want to alter when viewtype is "default"
    if ($params['viewtype'] !== 'default') {
        return $returnvalue;
    }

    // output nothing if the content doesn't have a single link
    if (false === elgg_strpos($returnvalue, '<a ')) {
        return '';
    }

    // returning nothing means "don't alter the returnvalue"
}
```

Remplacement complet du résultat de la vue

Vous pouvez pré-définir la sortie de la vue en définissant `$vars['__view_output']`. La valeur sera retournée sous forme de chaîne. Les extensions de vue ne seront pas utilisées et le hook `view` ne sera pas déclenché.

```
elgg_register_plugin_hook_handler('view_vars', 'navigation/breadcrumbs', 'myplugin_no_
↳page_breadcrumbs');

function myplugin_no_page_breadcrumbs($hook, $type, $vars, $params) {
    if (elgg_in_context('pages')) {
        return ['__view_output' => ""];
    }
}
```

Note : Pour une plus grande facilité d'utilisation, vous pouvez également utiliser une fonction de rappel par défaut pour bloquer tout rendu : `\Elgg\Values::preventViewOutput`

Afficher des entités

Si vous ne savez pas ce qu'est une entité, *commencez par lire cette page*.

Le code suivant affiche automatiquement l'entité dans `$entity` :

```
echo elgg_view_entity($entity);
```

Comme vous le savez après la lecture de l'introduction au modèle de données, toutes les entités ont un *type* (object, site, user ou group), et éventuellement un sous-type (qui pourrait être n'importe quoi - blog, forumpost, banane).

'`elgg_view_entity`' va automatiquement rechercher une vue appelée `type/subtype`; s'il n'y a pas de sous-type, elle recherchera `type/type`. À défaut, elle essaiera `type/default` avant de renoncer.

Les flux RSS d'Elgg fonctionnent généralement en affichant la vue `object/default` dans le type de vue "rss".

Par exemple, la vue pour afficher un article de blog peut être `object/blog`. La vue pour afficher un utilisateur est `user/default`.

Vues des entités partielles et complètes

`elgg_view_entity` dispose en fait d'un certain nombre de paramètres, bien que seul le tout premier soit nécessaire. Les trois premiers sont :

- `$entity` - L'entité à afficher
- `'$viewtype` - Le type de vue à afficher (par défaut c'est celui actuellement utilisé, mais il peut être forcé - par exemple pour afficher un extrait RSS dans une page HTML)
- `'$full_view` - Affichage d'une version *complète* de l'entité. (Par défaut à `true`.)

Ce dernier paramètre est transmis à la vue sous le nom de `$vars['full_view']`. C'est à vous de déterminer comment vous souhaitez l'utiliser; le comportement habituel consiste à afficher uniquement des commentaires et des informations de ce type si cela est défini sur `true`.

Lister des entités

Ceci est ensuite utilisé dans les fonctions de liste fournies. Pour afficher automatiquement une liste d'articles de blog (voir le tutoriel complet), vous pouvez appeler :

```
echo elgg_list_entities([
    'type' => 'object',
    'subtype' => 'blog',
]);
```

Cette fonction vérifie s'il y a des entités ; s'il y en a, elle affiche d'abord la vue `navigation/pagination` afin d'afficher un moyen de passer d'une page à l'autre. Elle appelle ensuite à plusieurs reprises `elgg_view_entity` sur chaque entité avant de renvoyer le résultat.

Notez que `elgg_list_entities` permet à l'URL de définir ses options `limit` et `offset`, aussi définissez-les explicitement si vous avez besoin de valeurs particulières (par exemple si vous ne les utilisez pas pour la pagination).

Elgg sait qu'il peut automatiquement fournir un flux RSS sur les pages qui utilisent `elgg_list_entities`. Il initialise le hook de plugin `["head", "page"]` (qui est utilisé par l'entête) afin de fournir la découverte automatique RSS, c'est pourquoi vous pouvez voir l'icône RSS orange sur ces pages dans certains navigateurs.

Les listes d'entités tenteront par défaut de charger les propriétaires des entités et les propriétaires de leur conteneur. Si vous voulez éviter cela, vous pouvez le désactiver.

```
echo elgg_list_entities([
    'type' => 'object',
    'subtype' => 'blog',

    // disable owner preloading
    'preload_owners' => false,
]);
```

Voyez aussi *ces informations de contexte sur la base de données d'Elgg*.

Si vous souhaitez afficher un message lorsque la liste ne contient pas d'élément à répertorier, vous pouvez transmettre un message avec `no_results`, ou `true` pour le message par défaut. Si vous voulez encore plus de contrôle sur le message le message `no_results` peut également passer une Closure (une fonction anonyme).

```
echo elgg_list_entities([
    'type' => 'object',
    'subtype' => 'blog',

    'no_results' => elgg_echo('notfound'),
]);
```

Rendu d'une liste avec une autre vue

Vous pouvez définir une autre vue pour afficher des éléments de liste à l'aide du paramètre `item_view`.

Dans certains cas, les vues d'entité par défaut peuvent ne pas convenir à vos besoins. L'utilisation de `item_view` vous permet de personnaliser l'apparence, tout en préservant la pagination, le balisage HTML de la liste, etc.

Prenons les deux exemples suivants :

```
echo elgg_list_entities([
    'type' => 'group',
    'relationship' => 'member',
    'relationship_guid' => elgg_get_logged_in_user_guid(),
    'inverse_relationship' => false,
    'full_view' => false,
]);
```

```
echo elgg_list_entities([
    'type' => 'group',
    'relationship' => 'invited',
    'relationship_guid' => (int) $user_guid,
    'inverse_relationship' => true,
    'item_view' => 'group/format/invitationrequest',
]);
```

Dans le premier exemple, nous affichons une liste de groupes dont un utilisateur est membre à l'aide de l'affichage de groupe par défaut. Dans le deuxième exemple, nous voulons afficher une liste de groupes auxquels l'utilisateur a été invité.

Étant donné que les invitations ne sont pas des entités, elles n'ont pas leurs propres vues et ne peuvent pas être répertoriées à l'aide de `elgg_list_*`. Nous fournissons une vue d'élément alternative, qui utilisera l'entité de groupe pour afficher une invitation qui contient un nom de groupe et des boutons pour accéder ou rejeter l'invitation.

Rendu d'une liste en tant que tableau

Depuis 2.3, vous pouvez afficher des listes sous forme de tableaux. Définissez `$options['list_type'] = 'table'` et fournissez un tableau d'objets `TableColumn` comme `$options['columns']`. Le service `elgg()->table_columns` fournit plusieurs méthodes pour créer des objets de colonnes basés sur des vues (comme `page/components/column/*`), des propriétés, ou des méthodes existantes.

Dans cet exemple, nous répertorions les derniers objets `mon_plugin` dans un tableau de 3 colonnes : icône de l'entité, nom d'affichage, et date dans un format convivial.

```
echo elgg_list_entities([
    'type' => 'object',
    'subtype' => 'my_plugin',

    'list_type' => 'table',
    'columns' => [
        elgg()->table_columns->icon(),
        elgg()->table_columns->getDisplayName(),
        elgg()->table_columns->time_created(null, [
            'format' => 'friendly',
        ]),
    ],
]);
```

Pour plus d'informations sur la façon dont les colonnes sont spécifiées et rendues, consultez la classe `Elgg\Views\TableColumn\ColumnFactory`. Vous pouvez ajouter ou remplacer des méthodes de `elgg()->table_columns` de diverses façons, en fonction des vues, des propriétés/méthodes sur les éléments ou des fonctions données.

Icônes

Elgg supporte deux types d'icônes : les icônes génériques pour aider avec le style des pages (par exemple, afficher l'icône de suppression) et les icônes des Entity (par exemple l'avatar utilisateur).

Icônes génériques

À partir d'Elgg 2.0, les icônes génériques sont basées sur la bibliothèque [FontAwesome](#). Vous pouvez obtenir n'importe laquelle des icônes prises en charge en appelant `elgg_view_icon($icon_name, $vars)`; où :

- `$icon_name` est le nom de l'icône FontAwesome (sans `fa-`) par exemple `user`
- `$vars` est facultatif, par exemple vous pouvez définir une classe supplémentaire

`elgg_view_icon()` appelle la vue `output/icon` avec le nom de l'icône donné et ajoute toutes les classes pour afficher l'icône FontAwesome. Si vous souhaitez remplacer une icône par une autre icône, vous pouvez écrire un hook `view_vars, output/icon` pour remplacer le nom de l'icône par votre remplacement.

A des fins de rétro-compatibilité, certains anciens noms d'icônes Elgg sont traduits en une icône FontAwesome correspondante.

Icônes des entités

Pour afficher un icône appartenant à une Entité appelez `elgg_view_entity_icon($entity, $size, $vars)`; où :

- `$entity` est l'`ElggEntity` dont vous voulez afficher l'icône
- `$size` est la dimension demandée. Par défaut Elgg supporte `large`, `medium`, `small`, `tiny` et `topbar` (`master` est également disponible, mais ne l'utilisez pas)
- `$vars` pour passer des informations additionnelles à la vue de l'icône

`elgg_view_entity_icon()` appelle une vue dans l'ordre :

- `icon/<type>/<subtype>`
- `icon/<type>/default`
- `icon/default`

Ainsi si vous souhaitez personnaliser la mise en page de l'icône, vous pouvez surcharger la vue correspondante.

Un exemple d'affichage d'un avatar utilisateur est

```
// get the user
$user = elgg_get_logged_in_user_entity();

// show the small icon
echo elgg_view_entity_icon($user, 'small');

// don't add the user_hover menu to the icon
echo elgg_view_entity_icon($user, 'small', [
    'use_hover' => false,
]);
```

Connexe

Recommandations pour la structure des pages

Les pages Elgg utilisent un pageshell (coquille HTML) global, une mise en page principale et plusieurs éléments de page. Il est recommandé d'utiliser toujours la disposition `default` car tous les éléments de page peuvent être contrôlés à l'aide de cette disposition.

Note : Plusieurs des anciennes dispositions (dépréciées) sont automatiquement réécrites pour utiliser la disposition `default` :

- `one_column`
 - `one_sidebar`
 - `two_sidebar`
 - `contenu`
-

If you're not using the `default` layout you can call

```
$layout_area = elgg_view_layout($layout_name, [  
    'content' => $content,  
    'section' => $section,  
]);
```

Les différents éléments de page sont passés sous la forme d'un array dans le deuxième paramètre. Les touches de tableau correspondent à des éléments de la disposition. Les valeurs de tableau contiennent le html qui devrait être affiché dans ces zones :

```
$layout_area = elgg_view_layout('default', [  
    'content' => $content,  
]);
```

```
$layout_area = elgg_view_layout('default', [  
    'content' => $content,  
    'sidebar' => $sidebar,  
]);
```

Vous pouvez ensuite, finalement, passer ceci dans la fonction `elgg_view_page` :

```
echo elgg_view_page($title, $layout_area);
```

Si vous utilisez la disposition `default`, vous pouvez également passer le tableau avec les éléments de page directement à `elgg_view_page` :

```
echo elgg_view_page($title, [  
    'content' => $content,  
    'sidebar' => $sidebar,  
]);
```

Vous pouvez contrôler de nombreux élément de la page :

```
echo elgg_view_page('This is the browser title', [  
    'title' => 'This is the page title',  
    'content' => $content,
```

(suite sur la page suivante)

(suite de la page précédente)

```
'sidebar' => false, // no default sidebar
'sidebar_alt' => $sidebar_alt, // show an alternate sidebar
]);
```

Voir aussi :

Regardez la vue `page/layouts/default` pour obtenir plus d'informations sur les éléments de page pris en charge

Cache simple (Simplecache)**Voir aussi :**

- [Performance](#)
- [Vues](#)

Le Simplecache est un mécanisme conçu pour atténuer la nécessité de régénérer dynamiquement certaines vues. Au lieu de cela, elles sont générées une seule fois, enregistrées comme un fichier statique, et servies d'une manière qui contourne entièrement le moteur Elgg.

Si Simplecache est désactivé (ce qui peut être fait à partir du panneau d'administration), ces vues seront servies comme d'habitude, à l'exception des CSS du site.

Les critères pour savoir si une vue convient pour Simplecache sont les suivants :

- La vue ne doit pas changer selon par qui ou quand elle est affichée
- La vue ne doit pas dépendre des variables qui lui sont transmises (à l'exception des variables globales comme l'URL du site qui ne changent jamais)

Régénérer le Simplecache

Vous pouvez régénérer la Simplecache à tout moment en :

- Chargement de `/upgrade.php`, même si vous n'avez rien à mettre à niveau
- Dans le panneau d'administration, cliquez sur "Vider les caches"
- Activation ou désactivation d'un plugin
- Réorganiser vos plugins

Utilisation du Simplecache dans vos plugins**Enregistrement des vues avec Simplecache**

Vous pouvez enregistrer une vue avec le Simplecache avec la fonction suivante lors de la phase d'initialisation :

```
elgg_register_simplecache_view($viewname);
```

Accès à la vue mise en cache

Si vous avez enregistré un fichier JavaScript ou CSS avec Simplecache et que vous l'avez mis dans le dossier des vues sous la forme `votre_vue.js` ou `votre_vue.css`, vous pouvez très facilement obtenir l'URL de cette vue mise en cache en appelant `elgg_get_simplecache_url($votre_vue)`. Par exemple :

```
$js = elgg_get_simplecache_url('your_view.js');
$css = elgg_get_simplecache_url('your_view.css');
```

Page/elements/footer vs footer

page/elements/footer est le contenu qui s’affiche dans cette partie de la page :

```
<div class="elgg-page-footer">
  <div class="elgg-inner">
    <!-- page/elements/footer goes here -->
  </div>
</div>
```

Son contenu est visible des utilisateurs finaux, et c’est habituellement l’endroit où vous mettriez un plan du site ou un autre système de navigation secondaire, des informations sur les mentions légales, l’indication propulsé par elgg, etc.

page/elements/footer est inséré juste avant la balise de fin `</body>` et est principalement prévu pour insérer des scripts qui ne fonctionnent pas déjà avec `elgg_register_js(array('location' => 'footer'))`; ou `elgg_require_js('amd/module')`; . En d’autres termes, vous ne devriez jamais surcharger cette vue et ne devriez sans doute pas l’étendre non plus. Utilisez plutôt les fonctions `elgg*_js` à la place

3.3.32 Walled Garden (Réseau privé)

Elgg prend en charge un mode Walled Garden (jardin clos). Dans ce mode, presque toutes les pages ne sont accessible qu’aux seuls utilisateurs connectés. Ceci est utile pour les sites qui n’autorisent pas l’inscription publique.

Activer le mode Walled Garden

Pour activer le mode Walled Garden dans Elgg, accédez à la section Administration. Dans le menu de la barre latérale droite, sous la section « Configurer », développez « Paramètres » puis cliquez sur « Avancé ».

Dans la page Paramètres avancés, recherchez l’option étiquetée « Restreindre les pages aux utilisateurs connectés ». Activez cette option, puis cliquez sur « Enregistrer » pour passer votre site en mode Walled Garden.

Exposer des pages à travers le Walled Garden

De nombreux plugins étendent Elgg en ajoutant des pages. Le mode Walled Garden empêchera ces pages d’être vues par les utilisateurs déconnectés. Elgg utilise un *hook de plugin* pour gérer les pages visibles à travers le walled garden.

Les auteurs de plugins doivent enregistrer les pages en tant que publiques si elles doivent être visibles à travers le walled garden :

- en définissant `'walled' => false` dans la configuration des routes
- en répondant au hook de plugin `public_pages, walled_garden`. La valeur retournée est un tableau d’expressions regex pour les pages publiques.

Le code suivant montre comment exposer http://exemple.org/mon_plugin/public_page à travers la walled garden. Cela suppose que le plugin a enregistré une route pour `mon_plugin/public_page`.

```
// Preferred way
elgg_register_route('my_plugin:public_page', [
  'path' => '/my_plugin/public_page',
  'resource' => 'my_plugin/public_page',
  'walled' => false,
]);

// Legacy approach
```

(suite sur la page suivante)

(suite de la page précédente)

```

elgg_register_plugin_hook_handler('public_pages', 'walled_garden', 'my_plugin_walled_
↪garden_public_pages');

function my_plugin_walled_garden_public_pages($hook, $type, $pages) {
    $pages[] = 'my_plugin/public_page';
    return $pages;
}

```

3.3.33 Web services

Construisez une API HTTP pour votre site.

Elgg fournit un framework puissant pour la création de services web. Cela permet aux développeurs d'exposer des fonctionnalités à d'autres sites Web et applications de bureau, ainsi que des intégrations avec des applications web tierces. Bien que nous appelions l'API RESTful, il s'agit en fait d'un hybride REST/RPC similaire aux API fournies par des sites comme Flickr et Twitter.

Pour créer une API pour votre site Elgg, vous devez effectuer 4 choses :

- activer le plugin web services
- exposer des méthodes
- mettre en place l'API d'authentification
- mettre en place l'authentification utilisateur

De plus, vous pouvez contrôler quels types d'authentification sont disponibles sur votre site. Cela sera également traité.

Contenu

- *Sécurité*
- *Exposer des méthodes*
 - *Formats de réponse*
 - *Paramètres*
 - *Recevoir des paramètres en tant que tableau associatif*
- *API d'authentification*
 - *Authentification par clef*
 - *Authentification par signature*
- *Authentification utilisateur*
- *Créer votre propre API*
- *Déterminer l'authentification disponible*
- *Connexe*

Sécurité

Il est essentiel que les services web soient utilisés via des protocoles sécurisés. N'activez pas les services web si votre site n'est pas servi par HTTPS. Ceci est particulièrement important si vous autorisez l'authentification par clef d'API uniquement.

Si vous utilisez des outils tiers qui exposent des méthodes d'API, assurez-vous d'effectuer un audit de sécurité approfondi. Vous pouvez vouloir vous assurer que l'authentification API est requise pour TOUTES les méthodes, même si elles nécessitent l'authentification de l'utilisateur. Les méthodes qui ne nécessitent pas d'authentification API peuvent être facilement utilisées pour spammer votre site.

Assurez-vous que la validité des clefs d'API est limitée et fournissez à vos clients API des mécanismes pour renouveler leurs clefs.

Exposer des méthodes

La fonction à utiliser pour exposer une méthode est `elgg_ws_expose_function()`. Par exemple, supposons que vous souhaitez exposer une fonction qui renvoie du texte à l'application appelante. La fonction pourrait ressembler à ceci

```
function my_echo($string) {  
    return $string;  
}
```

Puisque nous fournissons cette fonction pour permettre aux développeurs de tester leurs clients de l'API, nous n'aurons besoin ni d'authentification API ni d'authentification utilisateur. Cet appel enregistre la fonction avec le framework API des services web :

```
elgg_ws_expose_function(  
    "test.echo",  
    "my_echo",  
    [  
        "string" => [  
            'type' => 'string',  
        ],  
    ],  
    'A testing method which echos back a string',  
    'GET',  
    false,  
    false  
);
```

Si vous ajoutez ce code à un plugin, puis accédez à <http://votresite.com/services/api/rest/json/?method=system.api.list>, vous devez maintenant voir votre méthode `test.echo` répertoriée comme un appel API. En outre, pour tester la méthode exposée à partir d'un navigateur web, vous pouvez accéder à l'URL : <http://votresite.com/services/api/rest/json/?method=test.echo&string=testing> et vous devriez voir les données JSON comme ceci :

```
{  
    "status":0,  
    "result":"testing"  
}
```

Les plugins peuvent filtrer la sortie des méthodes d'API individuelles en enregistrant un gestionnaire pour le hook plugin `'rest:output', $method`.

Formats de réponse

JSON est le format par défaut, mais il est possible de récupérer du XML et du PHP sérialisé en activant le plugin `data_views` et en remplaçant `json` par `xml` ou `php` dans les URL ci-dessus.

Vous pouvez également ajouter des formats de réponse supplémentaires en définissant de nouveaux types de vues.

Paramètres

Les paramètres attendus par chaque méthode doivent être répertoriés comme un tableau associatif, où la clef représente le nom du paramètre, et la valeur contient un tableau avec des champs `type`, `default` et `required`.

Les valeurs soumises avec la requête auprès de d'API pour chaque paramètre doivent correspondre au type déclaré. L'API renverra une exception en cas d'échec de la validation.

Les types de paramètres reconnus sont :

- `integer` (ou `int`)
- `boolean` (ou `bool`) `'false'`, `0` et `'0'` seront évalués comme `false`, le reste sera évalué comme `true`
- `string`
- `float`
- `array`

Les types non reconnus lancent une exception d'API.

Vous pouvez utiliser des champs supplémentaires pour décrire votre paramètre, par exemple `description`.

```
elgg_ws_expose_function(
    'test.greet',
    'my_greeting',
    [
        'name' => [
            'type' => 'string',
            'required' => true,
            'description' => 'Name of the person to be greeted by the API',
        ],
        'greeting' => [
            'type' => 'string',
            'required' => false,
            'default' => 'Hello',
            'description' => 'Greeting to be used, e.g. "Good day" or "Hi"',
        ],
    ],
    'A testing method which greets the user with a custom greeting',
    'GET',
    false,
    false
);
```

Note : Si un paramètre manquant n'a pas de valeur par défaut, l'argument sera `null`. Avant Elgg v2.1, un bogue provoquait le décalage vers la gauche des arguments ultérieurs dans ce cas.

Recevoir des paramètres en tant que tableau associatif

Si vous avez d'un grand nombre de paramètres de méthode, vous pouvez forcer le script d'exécution à invoquer la fonction de rappel avec un seul argument qui contient un tableau associatif de paires de paramètre => valeur (au lieu que chaque paramètre soit un argument distinct). Pour ce faire, définissez `$assoc` sur `true` dans `elgg_ws_expose_function()`.

```
function greet_me($values) {
    $name = elgg_extract('name', $values);
    $greeting = elgg_extract('greeting', $values, 'Hello');
    return "$greeting, $name";
}

elgg_ws_expose_function(
    "test.greet",
    "greet_me",
    [
        "name" => [
            'type' => 'string',
        ],
        "greeting" => [
            'type' => 'string',
            'default' => 'Hello',
            'required' => false,
        ],
    ],
    'A testing method which echos a greeting',
    'GET',
    false,
    false,
    true // $assoc makes the callback receive an associative array
);
```

Note : Si un paramètre manquant n'a pas de valeur par défaut, `null` sera utilisé.

API d'authentification

Vous pouvez souhaiter contrôler l'accès à certaines des fonctions que vous exposez. Peut-être que vous exposez des fonctions afin d'intégrer Elgg avec une autre plate-forme opensource sur le même serveur. Dans ce cas, vous souhaitez seulement permettre à cette autre application d'accéder à ces méthodes. Une autre possibilité est que vous souhaitez limiter ce à quoi des développeurs externes ont accès via votre API. Ou peut-être voulez-vous limiter le nombre d'appels qu'un développeur peut passer auprès de votre API en une seule journée.

Dans tous ces cas, vous pouvez utiliser les fonctions d'authentification de l'API d'Elgg pour contrôler l'accès. Elgg fournit deux méthodes intégrées pour effectuer l'authentification auprès de l'API : la première basée sur la clef, et la signature HMAC. Vous pouvez également ajouter vos propres méthodes d'authentification. L'approche basée sur les clefs est très similaire à ce que font Google, Flickr, ou Twitter. Les développeurs peuvent demander une clef (une chaîne aléatoire) et transmettre cette clef avec tous les appels nécessitant l'authentification auprès de l'API. Les clefs sont stockées dans la base de données et si un appel API est effectué sans clef ou avec une mauvaise clef, l'appel est refusé et un message d'erreur est renvoyé.

Authentification par clef

À titre d'exemple, écrivons une fonction qui renvoie le nombre d'utilisateurs qui ont consulté le site au cours des dernières x minutes.

```
function count_active_users($minutes=10) {
    $seconds = 60 * $minutes;
    $count = count(find_active_users($seconds, 9999));
    return $count;
}
```

Maintenant, nous allons l'exposer et faire du nombre de minutes un paramètre facultatif :

```
elgg_ws_expose_function(
    "users.active",
    "count_active_users",
    [
        "minutes" => [
            'type' => 'int',
            'required' => false,
        ],
    ],
    'Number of users who have used the site in the past x minutes',
    'GET',
    true,
    false
);
```

Cette fonction est maintenant disponible, et si vous cochez `system.api.list`, vous verrez qu'elle nécessite l'authentification API. Si vous testez la méthode avec un navigateur Web, cela renvoie un message d'erreur concernant l'échec de l'authentification API. Pour tester cette méthode, vous avez besoin d'une clef API. À partir d'Elgg 3.2 les clefs API peuvent être générées par le plugin webservises. Il permet de créer une clef publique et privée, et vous devrez utiliser la clef publique pour ce type d'authentification API. Prenez une clef, puis effectuez une requête GET avec votre navigateur sur cette méthode d'API en passant dans la chaîne de paramètre la clef `api_key`. Cela pourrait ressembler à quelque chose comme ceci : http://votresite.com/services/api/rest/xml/?method=users.active&api_key=1140321cb56c71710c38feefdf72bc462938f59f.

Authentification par signature

Le *Authentification HMAC* est similaire à ce qui est utilisé avec OAuth ou le service S3 d'Amazon. Il s'agit d'utiliser à la fois clef publique et clef privée. Si vous voulez vous assurer que les appels API proviennent du développeur dont vous pensez qu'ils viennent et que vous voulez vous assurer que les données ne sont pas altérées pendant la transmission, vous devriez utiliser cette méthode d'authentification. Soyez conscient que cela demande beaucoup plus de travail et que cela pourrait décourager des développeurs quand il existe d'autres qui n'utilisent qu'une simple authentification basée sur les clefs.

Authentification utilisateur

Jusqu'à présent, vous avez permis aux développeurs d'extraire des données de votre site Elgg. Maintenant, nous allons passer à pousser des données dans Elgg. Dans ce cas, cela va être fait par un utilisateur. Par exemple vous pouvez avoir créé une application de bureau qui permet à vos utilisateurs de poster sur le fil sans aller sur le site. Vous devez exposer une méthode pour afficher sur le fil et vous devez vous assurer qu'un utilisateur ne peut pas publier à l'aide du compte de quelqu'un d'autre. Elgg fournit une approche basée sur des jetons pour l'authentification utilisateur. Cela permet à un utilisateur de soumettre son nom d'utilisateur et son mot de passe en échange d'un jeton utilisant la méthode `auth.gettoken`. Ce jeton peut ensuite être utilisé pendant un certain temps comme paramètre `auth_token` pour authentifier tous les appels à l'API, jusqu'à ce qu'il expire. Si vous ne voulez pas que vos utilisateurs fassent confiance à leurs mots de passe pour des applications tierces, vous pouvez également étendre la capacité actuelle d'utiliser une approche comme OAuth.

Écrivons notre fonction d'affichage du fil :

```
function my_post_to_wire($text) {  
  
    $text = elgg_substr($text, 0, 140);  
  
    $access = ACCESS_PUBLIC;  
  
    // returns guid of wire post  
    return thewire_save_post($text, $access, "api");  
}
```

Exposer cette fonction est la même chose qu'auparavant, sauf que nous avons besoin de l'authentification de l'utilisateur et nous allons faire cela via des requêtes POST plutôt que des requêtes HTTP.

```
elgg_ws_expose_function(  
    "thewire.post",  
    "my_post_to_wire",  
    [  
        "text" => [  
            'type' => 'string',  
        ],  
    ],  
    'Post to the wire. 140 characters or less',  
    'POST',  
    true,  
    true  
);
```

Veuillez noter que vous ne pourrez pas tester cela à l'aide d'un navigateur Web comme vous l'avez fait avec les autres méthodes. Vous devez écrire du code client pour ce faire.

Créer votre propre API

Dès que vous vous sentirez à l'aise avec l'API du framework de web services Elgg, vous voudrez prendre du recul et concevoir votre API. Quel genre de données essayez-vous d'exposer ? Qui ou qu'est-ce qui seront les utilisateurs d'API ? Comment voulez-vous qu'ils aient accès aux clés d'authentification ? Comment allez-vous documenter votre API ? N'oubliez pas de jeter un œil aux API créées par les sites Web populaires 2.0 pour l'inspiration. Si vous recherchez des développeurs tiers pour créer des applications à l'aide de votre API, vous voudrez probablement fournir un ou plusieurs clients dans un langage spécifique.

Déterminer l'authentification disponible

L'API des services Web d'Elgg utilise un type d'architecture de [pluggable authentication module \(PAM\)](#) (module d'authentification insérable) pour gérer comment les utilisateurs et les développeurs sont authentifiés. Cela vous donne la flexibilité d'ajouter et de supprimer des modules d'authentification. Vous ne voulez pas utiliser l'authentification utilisateur par défaut PAM, mais préférez utiliser OAuth ? Vous pouvez le faire.

La première étape consiste à enregistrer une fonction de rappel pour le hook de plugin *rest, init* :

```
register_plugin_hook('rest', 'init', 'rest_plugin_setup_pams');
```

Ensuite, dans la fonction de rappel, vous enregistrez les PAMs que vous souhaitez utiliser :

```
function rest_plugin_setup_pams() {
    // user token can also be used for user authentication
    register_pam_handler('pam_auth_usertoken');

    // simple API key check
    register_pam_handler('api_auth_key', "sufficient", "api");

    // override the default pams
    return true;
}
```

Lors des tests, vous trouverez peut-être utile d'enregistrer le PAM `pam_auth_session` afin que vous puissiez facilement tester vos méthodes à partir du navigateur. Veillez à ne pas utiliser ce PAM sur un site de production, car il pourrait ouvrir vos utilisateurs à une [attaque CSRF](#).

Connexe

Authentification HMAC

Le framework d'API RESTful d'Elgg fournit des fonctions pour prendre en charge un schéma de signature [HMAC](#) pour l'authentification API. Le client doit envoyer la signature HMAC ainsi qu'un ensemble d'en-têtes HTTP spéciaux lors de l'exécution d'un appel qui nécessite l'authentification API. Cela garantit que l'appel API est effectué à partir du client déclaré et que les données n'ont pas été falsifiées.

Le HMAC doit être construit à partir des données suivantes :

- La clef d'API publique qui vous identifie auprès du serveur api d'Elgg tel que fourni par le plugin APIAdmin
- La clef d'API privée fournie par Elgg (qui accompagne la clef publique)
- L'heure unix actuelle en secondes
- Une valeur créée pour l'occasion pour garantir que deux requêtes faites à la même seconde ont des signatures différentes
- Une représentation de la chaîne encodée pour l'URL de n'importe quels paramètres variables GET, par ex. `method=test.test&foo=bar`

- Si vous envoyez des données post, le hash de ces données

Quelques informations supplémentaires doivent être ajoutées à l'entête HTTP pour que ces données soient traitées correctement :

- **X-Elgg-apikey** - La clef d'API publique
- **X-Elgg-time** - L'heure Unix utilisée dans le calcul HMAC
- **X-Elgg-nonce** - une chaîne aléatoire
- **X-Elgg-hmac** - Le HMAC encodé en base64
- **X-Elgg-hmac-algo** - L'algorithme utilisé dans le calcul HMAC

Si vous envoyez des données POST vous devez également envoyer :

- **X-Elgg-posthash** - Le hash des données POST
- **X-Elgg-posthash-algo** - Algorithme utilisé pour le hachage des données POST
- **Content-type** - Le type de contenu des données que vous envoyez (en cas de doute, utilisez `application/octet-stream`)
- **Content-Length** - La longueur et octets de vos données POST

Elgg fournit un exemple de client d'API qui implémente cette signature HMAC : `send_api_call()`. Il constitue un bon point de départ sur comment l'implémenter.

Algorithmes de hachage pris en charge

- sha256 : recommandé
- sha1 : rapide mais moins sûr
- md5 : faible et sera supprimé à l'avenir

Calcul de hachage des données POST

Le hachage de poste doit être calculé sur toutes les données POST à l'aide de l'un des algorithmes de hachage pris en charge. Le résultat du hachage doit être signalé dans l'en-tête **X-Elgg-posthash** et l'algorithme de hachage utilisé doit être signalé dans l'en-tête **X-Elgg-posthash-algo**.

Calcul du hachage HMAC

Le HMAC global doit être calculé sur les données suivantes (dans l'ordre) en utilisant le secret API comme le secret HMAC et avec l'un des algorithmes de hachage pris en charge :

1. un horodatage UNIX, indiquez cet horodatage dans l'en-tête **X-Elgg-time**
2. une chaîne aléatoire, indiquez cette chaîne dans l'en-tête **X-Elgg-nonce**
3. la clef publique d'API, indiquer cette clef d'API dans l'en-tête **X-Elgg-apikey**
4. la chaîne de requête url (par exemple `method=test.test&foo=bar`)
5. lorsque qu'il s'agit d'une requête POST, ajoutez le posthash tel qu'indiqué dans l'entête **X-Elgg-posthash**

La chaîne résultante doit être codée en base64, puis l'URL codée et indiquée dans l'en-tête **X-Elgg-hmac**. L'algorithme de hachage utilisé doit être signalé dans **X-Elgg-hmac-algo**.

Cache de hachage

Pour des raisons de sécurité, chaque hachage HMAC doit être unique, tous les hachages soumis sont stockés pendant 25 heures pour éviter la réutilisation.

Résultats de l'API

Contenu

- *Structure des résultats de succès*
- *Structure d'un résultat d'erreur*
- *Codes d'état par défaut*

Structure des résultats de succès

Un résultat d'API réussi ressemble à ceci :

```
{
    "status": 0,
    "result": "API result"
}
```

Selon l'appel API `result` peut contenir n'importe quel type de contenu (chaîne, nombre, tableau, objet, etc.).

Exemple de résultat numérique (par exemple un nombre d'utilisateurs) :

```
{
    "status": 0,
    "result": 10
}
```

Exemple de résultat objet (par exemple un utilisateur) :

```
{
    "status": 0,
    "result": {
        "name": "Some user",
        "username": "apiexample",
        "email": "user@example.com"
    }
}
```

Structure d'un résultat d'erreur

Lorsqu'un appel API échoue, le résultat ressemble à ceci :

```
{
  "status": -1,
  "message": "The reason the API call failed"
}
```

Codes d'état par défaut

Le champ `status` contient toujours un nombre représentant le résultat. Toute valeur autre que `0` est considérée comme une erreur.

- `0` : C'est un résultat réussi
- `-1` : Il s'agit d'un résultat d'erreur générique
- `-20` : le jeton d'authentification utilisateur est manquant, est invalide ou a expiré
- `-30` : La clé api a été désactivée
- `-31` : La clé api est inactive
- `-32` : La clef api est invalide

Les développeurs peuvent implémenter leurs propres codes d'état pour représenter différents états d'erreur, de sorte que la demande n'a pas à s'appuyer sur le message d'erreur pour savoir ce qui s'est passé.

Note : `result` et `message` peuvent contenir des messages dans différentes langues. Cela dépend de la langue utilisateur lors de l'utilisation d'appels API authentifiés par l'utilisateur ou de la langue de site pour les autres appels API. Gardez à l'esprit que la langue peut changer, que ce soit par l'utilisateur ou par un administrateur du site pour la langue du site.

3.3.34 Widgets

Les widgets sont des zones de contenu que les utilisateurs peuvent déplacer dans leur page pour personnaliser la mise en page. Ils peuvent typiquement être personnalisés par leur propriétaire pour afficher plus ou moins de contenu et déterminer qui peut voir le widget. Par défaut, Elgg fournit des plugins pour personnaliser la page de profil et le tableau de bord avec des widgets.

Contenu

- *Structure*
- *Enregistrez le widget*
 - *Widgets multiples*
 - *Nom et description magiques du widget*
 - *Comment restreindre les emplacements où les widgets peuvent être utilisés*
 - *Autorisez plusieurs widgets sur la même page*
 - *Enregistrez des widgets dans un hook*
 - *Modifiez les propriétés du widget d'un enregistrement de widget existant*
- *Widgets par défaut*

Structure

Pour créer un widget, créez deux vues :

- widgets/widget/edit
- widgets/widget/content

content.php est responsable pour tout le contenu qui sera affiché dans le widget. Le fichier edit.php contient toutes les fonctions d'édition supplémentaires que vous souhaitez proposer à l'utilisateur. Vous n'avez pas besoin d'ajouter de niveau d'accès car ceci est pris en charge par le framework des widgets.

Note : L'utilisation de cases à cocher HTML pour définir des drapeaux pour le widget est problématique parce que si la case n'est pas cochée, le champ de la case à cocher est omis de l'envoi du formulaire. Le résultat est que vous pouvez seulement définir ces champs, et pas les supprimer. La vue « input/checkboxes » ne va pas fonctionner correctement dans le panneau d'édition d'un widget.

Enregistrez le widget

Une fois que vous avez créé vos pages edit et view, vous devez initialiser le widget du plugin.

La manière la plus simple de faire cela est d'ajouter la section widgets à votre fichier de configuration elgg-plugin.php.

```
return [
    'widgets' => [
        'filerepo' => [
            'context' => ['profile'],
        ],
    ],
];
```

Alternativement, vous pouvez aussi utiliser une fonction pour ajouter un widget. Ceci est effectué au sein de la fonction init() du plugin.

```
// Add generic new file widget
elgg_register_widget_type([
    'id' => 'filerepo',
    'name' => elgg_echo('widgets:filerepo:name'),
    'description' => elgg_echo('widgets:filerepo:description'),
    'context' => ['profile'],
]);
```

Note : Le seul attribut requis est l'id.

Widgets multiples

Il est possible d'ajouter plusieurs widgets pour un plugin. Vous initialisez seulement autant de répertoires de widgets que vous voulez.

```
// Add generic new file widget
elgg_register_widget_type([
    'id' => 'filerepo',
    'name' => elgg_echo('widgets:filerepo:name'),
    'description' => elgg_echo('widgets:filerepo:description'),
    'context' => ['profile'],
]);

// Add a second file widget
elgg_register_widget_type([
    'id' => 'filerepo2',
    'name' => elgg_echo('widgets:filerepo2:name'),
    'description' => elgg_echo('widgets:filerepo2:description'),
    'context' => ['dashboard'],
]);

// Add a third file widget
elgg_register_widget_type([
    'id' => 'filerepo3',
    'name' => elgg_echo('widgets:filerepo3:name'),
    'description' => elgg_echo('widgets:filerepo3:description'),
    'context' => ['profile', 'dashboard'],
]);
```

Assurez-vous que vous avez les répertoires correspondants suivants au sien de la structure de votre plugin :

```
'Plugin'
  /views
    /default
      /widgets
        /filerepo
          /edit.php
          /content.php
        /filerepo2
          /edit.php
          /content.php
        /filerepo3
          /edit.php
          /content.php
```

Nom et description magiques du widget

Quand vous enregistrez un widget vous pouvez omettre de lui donner un nom et une description. Si la traduction est fournie dans le format suivant, elle sera utilisée. Pour le nom : `widgets:<widget_id>:name` et pour la description `widgets:<widget_id>:description`. Si vous vous assurez que ces traductions existent dans le fichier de traduction, vous avez très peu de travail pour enregistrer le widget.

```
elgg_register_widget_type(['id' => 'filerepo']);
```

Comment restreindre les emplacements où les widgets peuvent être utilisés

Le widget peut définir le contexte dans lequel il peut être utilisé (seulement profile, seulement dashboard, etc.).

```
elgg_register_widget_type([
    'id' => 'filerepo',
    'context' => ['profile', 'dashboard', 'other_context'],
]);
```

Autorisez plusieurs widgets sur la même page

Par défaut vous ne pouvez ajouter qu'un seul widget du même type sur une page. Si vous voulez plusieurs widgets du même type sur la page, vous pouvez spécifier ceci quand vous enregistrez le widget :

```
elgg_register_widget_type([
    'id' => 'filerepo',
    'multiple' => true,
]);
```

Enregistrez des widgets dans un hook

Si, par exemple, vous souhaitez enregistrer des widgets de manière conditionnelle, vous pouvez utiliser un hook pour enregistrer des widgets.

```
function my_plugin_init() {
    elgg_register_plugin_hook_handler('handlers', 'widgets', 'my_plugin_conditional_
↳ widgets_hook');
}

function my_plugin_conditional_widgets_hook($hook, $type, $return, $params) {
    if (!elgg_is_active_plugin('file')) {
        return;
    }

    $return[] = \Elgg\WidgetDefinition::factory([
        'id' => 'filerepo',
    ]);

    return $return;
}
```

Modifiez les propriétés du widget d'un enregistrement de widget existant

Si, par exemple, vous souhaitez changer les contextes autorisés pour un widget déjà enregistré, vous pouvez faire cela en ré-enregistrant le widget avec `elgg_register_widget_type`, ce qui va remplacer une définition de widget existante. Si vous voulez encore plus de contrôle vous pouvez également utiliser le hook `handlers, widgets` pour changer la définition du widget.

```
function my_plugin_init() {
    elgg_register_plugin_hook_handler('handlers', 'widgets', 'my_plugin_change_widget_
    ↪definition_hook');
}

function my_plugin_change_widget_definition_hook($hook, $type, $return, $params) {
    foreach ($return as $key => $widget) {
        if ($widget->id === 'filerepo') {
            $return[$key]->multiple = false;
        }
    }

    return $return;
}
```

Widgets par défaut

Si votre plugin utilise le canvas widgets, vous pouvez enregistrer le support d'un widget par défaut avec le noyau d'Elgg, qui va se charger de tout le reste.

Pour déclarer le support d'un widget par défaut dans votre plugin, enregistrez le hook plugin `get_list, default_widgets`:

```
elgg_register_plugin_hook_handler('get_list', 'default_widgets', 'my_plugin_default_
    ↪widgets_hook');
```

Dans le gestionnaire du hook plugin, poussez un tableau dans la valeur de retour qui définit le support de votre widget par défaut et quand créer les widgets par défaut. Les tableaux doivent définir les clefs suivantes :

- `name` - Le nom de la page des widgets. Ceci est affiché sur l'onglet dans l'interface admin.
- `widget_context` - Le contexte depuis lequel la page widgets est appelée. (Si non défini explicitement, ce sera l'id de votre plugin.)
- `widget_columns` - Combien de colonnes la page widgets va utiliser.
- `event` - L'événement Elgg pour lequel créer de nouveaux widgets. C'est habituellement `create`.
- `entity_type` - Le type d'entité pour lequel créer de nouveaux widgets.
- `entity_subtype` - Le sous-type d'entité pour lequel créer de nouveaux widgets. Peut être `ELGG_ENTITIES_ANY_VALUE` pour le créer pour tous les types d'entités.

Quand un objet déclenche un événement qui correspond aux paramètres passés `event, entity_type, et entity_subtype`, le noyau d'Elgg va regarder les widgets par défaut qui correspondent au `widget_context` et va les copier vers l'`owner_guid` et le `container_guid` de l'objet. Tous les paramètres du widget seront également copiés.

```
function my_plugin_default_widgets_hook($hook, $type, $return, $params) {
    $return[] = array(
        'name' => elgg_echo('my_plugin'),
        'widget_context' => 'my_plugin',
        'widget_columns' => 3,
    );
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

        'event' => 'create',
        'entity_type' => 'user',
        'entity_subtype' => ELGG_ENTITIES_ANY_VALUE,
    );

    return $return;
}

```

3.4 Tutoriels

Suivez toutes les étapes requises pour personnaliser Elgg.

Les instructions sont suffisamment détaillées pour que vous n'ayez pas besoin de beaucoup d'expérience de développement avec Elgg.

3.4.1 Hello world

Ce tutoriel vous montre comment créer un nouveau plugin qui consiste en une nouvelle page qui affiche le texte « Hello world ».

Avant toute chose, vous devez *installer Elgg*.

Dans ce tutoriel, nous allons supposer que l'URL de votre site est `https://elgg.example.com`.

Tout d'abord, créez un répertoire qui va contenir les fichiers du plugin. Il devrait être placé dans le répertoire `mod/` situé dans le répertoire d'installation d'Elgg. Dans notre cas, créez `mod/hello/`.

Fichier Manifest

Elgg a besoin que votre plugin dispose d'un fichier manifest qui contient des informations sur le plugin. A cette fin, créez un fichier nommé `manifest.xml` dans le répertoire de votre plugin, et collez ce code dedans :

```

<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>Hello world</name>
  <id>hello</id>
  <author>Your Name Here</author>
  <version>0.1</version>
  <description>Hello world, testing.</description>
  <requires>
    <type>elgg_release</type>
    <version>2.0</version>
  </requires>
</plugin_manifest>

```

Voici le minimum d'informations qui doivent être présentes dans un fichier manifest :

- `<name>` est le nom du plugin tel qu'il sera affiché
- `<id>` doit correspondre au nom du répertoire que vous venez de créer
- `<requires>` doit indiquer la version minimum d'Elgg dont votre plugin a besoin
- `<author>`, `<version>` et `<description>` devraient avoir des valeurs appropriées mais peuvent être renseignés librement

Initialiseur

Puis créez `start.php` dans le répertoire `mod/hello/` et copiez ce code dedans :

```
<?php

elgg_register_event_handler('init', 'system', 'hello_world_init');

function hello_world_init() {

}
```

Le code ci-dessus indique à Elgg qu'il devrait appeler la fonction `hello_world_init()` une fois que le coeur du système Elgg est initialisé.

Enregistrer une route

L'étape suivante est d'enregistrer une route dont l'objectif est de traiter les requêtes que les utilisateurs font vers l'URL `https://elgg.example.com/hello`.

Modifiez `elgg-plugin.php` pour qu'il ressemble à ceci :

```
<?php

return [
    'routes' => [
        'default:hello' => [
            'path' => '/hello',
            'resource' => 'hello',
        ],
    ],
];
```

Cet enregistrement indique à Elgg d'appeler la vue ressource `hello` quand un utilisateur navigue vers `https://elgg.example.com/hello`.

Voir le fichier

Créez `mod/hello/views/default/resources/hello.php` avec ce contenu :

```
<?php

echo elgg_view_page('Hello', [
    'title' => 'Hello world!',
    'content' => 'My first page!',
]);
```

Le code crée un tableau de paramètres à passer à la fonction `elgg_view_layout()`, comprenant :

- Le titre de la page
- Le contenu de la page
- Un filter qui est laissé vide puisqu'il n'y a pour le moment rien à filtrer

Ceci crée l'agencement général (layout) de base pour la page. Cet agencement est ensuite passé à travers `elgg_view_page()` qui assemble et génère la page complète.

Dernière étape

Pour terminer, activez le plugin depuis la page d'administration d'Elgg : `https://elgg.example.com/admin/plugins` (le nouveau plugin apparaît en bas)

Vous pouvez maintenant vous rendre sur l'adresse `https://elgg.example.com/hello/` et vous devriez voir votre nouvelle page !

3.4.2 Personnaliser la page d'accueil

Pour remplacer la page d'accueil, surchargez simplement la vue d'Elgg `resources/index` en créant un fichier à `/views/default/resources/index.php`.

Toute sortie de cette vue deviendra votre nouvelle page d'accueil.

Vous pouvez prendre une approche similaire avec n'importe quelle autre page d'Elgg ou des plugins officiels.

3.4.3 Construire un plugin de Blog

Ce tutorial va vous apprendre à créer un plugin de blog simple. Les fonctionnalités de base du blog seront de créer des articles, les enregistrer et les afficher. Le plugin duplique des fonctionnalités présentes dans le plugin `blog` empaqueté. Vous pouvez désactiver le plugin `blog` empaqueté si vous le souhaitez, mais ce n'est pas nécessaire dans la mesure où les fonctionnalités ne seront pas en conflit les unes avec les autres.

Contenu

- *Créez le répertoire du plugin et le fichier manifest*
- *Créez le formulaire pour la création d'un nouvel article de blog*
- *Créez une page pour écrire les articles de blog*
- *Créez le fichier d'action pour enregistrer l'article de blog*
- *Créer `elgg-plugin.php`*
- *Créez le fichier `start.php`*
- *Créez une page pour afficher l'article de blog*
- *Créez la vue de l'objet*
- *Tester le plugin*
- *Afficher une liste des articles de blog*
- *FIN*

Prérequis :

- *Installation d'Elgg*

Créez le répertoire du plugin et le fichier manifest

Tout d'abord, choisissez un nom simple et descriptif pour votre plugin. Dans ce tutoriel, le nom sera `my_blog`. Puis créez un répertoire pour votre plugin dans le répertoire `/mod/` qui se trouve dans le répertoire d'installation d'Elgg. D'autres plugins sont également situés dans `/mod/`. Dans notre cas, le nom de ce répertoire devrait être `/mod/my_blog/`. Ce répertoire constitue la racine de votre plugin, et tous les fichiers que vous allez créer pour votre nouveau plugin vont l'être quelque part dans ce répertoire.

Ensuite, créez le fichier `manifest.xml` à la racine du plugin :

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>My Blog</name>
  <id>my_blog</id>
  <author>Your Name Here</author>
  <version>0.1</version>
  <description>Adds blogging capabilities.</description>
  <requires>
    <type>elgg_release</type>
    <version>2.0</version>
  </requires>
</plugin_manifest>
```

Voyez la documentation sur les *Plugins* pour plus d'informations sur le fichier manifest.

Créez le formulaire pour la création d'un nouvel article de blog

Créez un fichier `/mod/my_blog/views/default/forms/my_blog/save.php` qui contiendra le corps du formulaire. Le formulaire doit avoir des champs de saisie pour le titre, le corps et les tags de l'article de blog `my_blog`. Il n'est pas nécessaire d'ajouter les balises `<form>`.

```
echo elgg_view_field([
  '#type' => 'text',
  '#label' => elgg_echo('title'),
  'name' => 'title',
  'required' => true,
]);

echo elgg_view_field([
  '#type' => 'longtext',
  '#label' => elgg_echo('body'),
  'name' => 'body',
  'required' => true,
]);

echo elgg_view_field([
  '#type' => 'tags',
  '#label' => elgg_echo('tags'),
  '#help' => elgg_echo('tags:help'),
  'name' => 'tags',
]);

$submit = elgg_view_field(array(
  '#type' => 'submit',
  '#class' => 'elgg-foot',
  'value' => elgg_echo('save'),
));
elgg_set_form_footer($submit);
```

Notez comment le formulaire appelle `elgg_view_field()` pour afficher les champs de saisie. Cette fonction d'aide permet de maintenir la cohérence dans les balises des champs de saisie, et est utilisée comme raccourci pour afficher des éléments des champs, tels que le label, l'aide et le champs de saisie. Voir la *Formulaires + Actions*.

Vous pouvez voir une liste complète des vues d'entrée dans le répertoire `/vendor/elgg/elgg/views/default/input/`.

Il est recommandé que vous rendiez votre plugin traduisible en utilisant `elgg_echo()` à chaque fois qu'une chaîne de texte sera affichée à l'utilisateur. Lisez-en plus sur l'[Internationalisation](#).

Créez une page pour écrire les articles de blog

Créez le fichier `/mod/my_blog/views/default/resources/my_blog/add.php`. Cette page va afficher le formulaire que vous avez créé dans la section précédente.

```
<?php
// make sure only logged in users can see this page
gatekeeper();

// set the title
$title = "Create a new my_blog post";

// add the form to the main column
$content = elgg_view_form("my_blog/save");

// optionally, add the content for the sidebar
$sidebar = "";

// draw the page, including the HTML wrapper and basic page layout
echo elgg_view_page($title, [
    'content' => $content,
    'sidebar' => $sidebar
]);
```

La fonction `elgg_view_form("my_blog/save")` affiche le formulaire que vous avez créé dans la section précédente. Elle ajoute automatiquement au formulaire la balise `<form>` avec les attributs nécessaires ainsi que des jetons anti-csrf.

L'action du formulaire sera `"<?= elgg_get_site_url() ?>action/my_blog/save"`.

Créez le fichier d'action pour enregistrer l'article de blog

Le fichier d'action va enregistrer l'article de blog `my_blog` dans la base de données. Créez le fichier `/mod/my_blog/actions/my_blog/save.php` :

```
<?php
// get the form inputs
$title = get_input('title');
$body = get_input('body');
$tags = string_to_tag_array(get_input('tags'));

// create a new my_blog object and put the content in it
$blog = new ElggObject();
$blog->title = $title;
$blog->description = $body;
$blog->tags = $tags;

// the object can and should have a subtype
```

(suite sur la page suivante)

(suite de la page précédente)

```

$blog->subtype = 'my_blog';

// for now, make all my_blog posts public
$blog->access_id = ACCESS_PUBLIC;

// owner is logged in user
$blog->owner_guid = elgg_get_logged_in_user_guid();

// save to database and get id of the new my_blog
$blog_guid = $blog->save();

// if the my_blog was saved, we want to display the new post
// otherwise, we want to register an error and forward back to the form
if ($blog_guid) {
    system_message("Your blog post was saved.");
    forward($blog->getURL());
} else {
    register_error("The blog post could not be saved.");
    forward(REFERER); // REFERER is a global variable that defines the previous page
}

```

Comme vous pouvez le voir dans le code ci-dessus, les objets Elgg disposent de plusieurs champs intégrés. Le titre de l'article `my_blog` est conservé dans le champs `title` tandis que le contenu est conservé dans le champs `description`. Il y a également un champs pour des tags, qui sont stockés sous la forme de métadonnées.

Les objets dans Elgg sont une sous-classe de quelque chose appelé « entité » (entity). Les utilisateurs, sites et groupes sont également des sous-classes d'entité. Le sous-type (subtype) d'une entité permet un contrôle granulaire pour les listings et l'affichage, c'est pourquoi chaque entité devrait avoir un sous-type. Dans ce tutoriel, le sous-type « `my_blog` » permet d'identifier un article `my_blog` post, mais toute chaîne de caractères alphanumérique peut constituer un sous-type valide. Lorsque vous choisissez des sous-types, veuillez vous assurer d'en choisir un qui soit cohérent avec votre plugin.

La méthode `getURL` charge l'URL du nouvel article. Il est recommandé de surcharger cette méthode. La surcharge sera faite dans le fichier `start.php`.

Créer `elgg-plugin.php`

Le fichier `/mod/my_blog/elgg-plugin.php` est utilisé pour déclarer diverses fonctionnalités du plugin. Il peut, par exemple, être utilisé pour configurer des entités, des actions, des widgets et des routes.

```

<?php

return [
    'entities' => [
        [
            'type' => 'object',
            'subtype' => 'my_blog',
            'searchable' => true,
        ],
    ],
    'actions' => [
        'my_blog/save' => [],
    ],
];

```

(suite sur la page suivante)

(suite de la page précédente)

```

    ],
    'routes' => [
        'view:object:blog' => [
            'path' => '/my_blog/view/{guid}/{title?}',
            'resource' => 'my_blog/view',
        ],
        'add:object:blog' => [
            'path' => '/my_blog/add/{guid?}',
            'resource' => 'my_blog/add',
        ],
        'edit:object:blog' => [
            'path' => '/my_blog/edit/{guid}/{revision?}',
            'resource' => 'my_blog/edit',
            'requirements' => [
                'revision' => '\d+',
            ],
        ],
    ],
],
];

```

Créez le fichier start.php

Le fichier `/mod/my_blog/start.php` doit enregistrer un hook pour surcharger la génération d'URL.

```

<?php

function my_blog_init() {
    // register a hook handler to override urls
    elgg_register_plugin_hook_handler('entity:url', 'object', 'my_blog_set_url');
}

return function() {
    // register an initializer
    elgg_register_event_handler('init', 'system', 'my_blog_init');
}

```

L'enregistrement de l'action de sauvegarde va la rendre disponible via `/action/my_blog/save`. Par défaut, toutes les actions ne sont disponibles qu'aux utilisateurs identifiés. Si vous souhaitez rendre une action disponible aux seuls administrateurs ou l'ouvrir y compris à des visiteurs non identifiés, vous pouvez passer respectivement "admin" ou "public" comme troisième paramètre de `elgg_register_action`.

La fonction de surcharge d'URL va extraire l'ID de l'entité et l'utiliser pour créer une URL simple pour la page qui va afficher cette entité. Dans ce cas l'entité devrait bien sûr être un article `my_blog`. Ajoutez cette fonction à votre fichier `start.php` :

```

function my_blog_set_url($hook, $type, $url, $params) {
    $entity = $params['entity'];
    if ($entity->getSubtype() === 'my_blog') {
        return "my_blog/view/{ $entity->guid }";
    }
}

```

Le gestionnaire de page permet de servir la page qui génère le formulaire et la page qui affiche l'article. La prochaine section va montrer comment créer la page qui affiche l'article. Ajoutez cette fonction à votre fichier *start.php* :

```
function my_blog_page_handler($segments) {
    if ($segments[0] == 'add') {
        echo elgg_view_resource('my_blog/add');
        return true;
    }

    else if ($segments[0] == 'view') {
        $resource_vars['guid'] = elgg_extract(1, $segments);
        echo elgg_view_resource('my_blog/view', $resource_vars);
        return true;
    }

    return false;
}
```

La variable `$segments` contient les différentes parties de l'URL une fois séparées par `/`.

Les fonctions de gestionnaires de pages doivent retourner `true` ou `false`. `true` signifie que la page existe et a bien été traitée par le gestionnaire de pages. `false` signifie que la page n'existe pas et que l'utilisateur va être redirigé vers la page d'erreur 404 du site (la page demandée n'existe pas ou n'a pas été trouvée). Dans cet exemple, l'URL doit contenir `/my_blog/add` ou `/my_blog/view/id` où `id` est un identifiant valide d'une entité avec le sous-type `my_blog`. Vous trouverez plus d'informations sur les gestionnaires de pages sur [Page handler](#).

Créez une page pour afficher l'article de blog

Pour pouvoir afficher un article `my_blog` sur sa propre page, vous devez créer une page d'affichage. Créez le fichier `/mod/my_blog/views/default/resources/my_blog/view.php` :

```
<?php

// get the entity
$guid = elgg_extract('guid', $vars);
$my_blog = get_entity($guid);

// get the content of the post
$content = elgg_view_entity($my_blog, array('full_view' => true));

echo elgg_view_page($my_blog->getDisplayName(), [
    'content' => $content,
]);
```

Cette page a beaucoup de points communs avec la page `add.php`. Les principales différences sont que les informations sont extraites depuis l'entité `my_blog`, et qu'au lieu d'afficher un formulaire, la fonction `elgg_view_entity` est appelée. Cette fonction donne les informations de l'entité à ce qu'on appelle la vue de l'objet.

Créez la vue de l'objet

Quand `elgg_view_entity` est appelé ou quand des articles `my_blogs` sont affichés dans une liste par exemple, la vue de l'objet va générer le contenu approprié. Créez le fichier `/mod/my_blog/views/default/object/my_blog.php` :

```
<?php

echo elgg_view('output/longtext', array('value' => $vars['entity']->description));
echo elgg_view('output/tags', array('tags' => $vars['entity']->tags));
```

Comme vous pouvez le voir dans la section précédente, chaque article `my_blog` est passé à la vue de l'objet sous la forme `$vars['entity']`. (`$vars` est un tableau utilisé dans le système de vues pour passer des variables à une vue.)

La dernière ligne prend les tags de l'article `my_blog` et les affiche automatiquement sous la forme d'une série de liens cliquables. La recherche est gérée automatiquement.

(Si vous vous interrogez sur le « `default` » dans `/views/default/`, vous pouvez créer des vues alternatives. RSS, OpenDD, FOAF, mobile et d'autres sont divers types de vues valides.)

Tester le plugin

Rendez-vous sur la page d'administration d'Elgg, listez les plugins, et activez le plugin `my_blog`.

La page pour créer un nouvel article `my_blog` devrait désormais être accessible via `https://elgg.example.com/my_blog/add`, et après avoir enregistré l'article, vous devriez le voir affiché sur sa propre page.

Afficher une liste des articles de blog

Créons également une page qui liste les entrées `my_blog` qui ont été créées

Créez `/mod/my_blog/views/default/resources/my_blog/all.php` :

```
<?php
$titlebar = "All Site My_Blogs";
$page_title = "List of all my_blogs";

$body = elgg_list_entities(array(
    'type' => 'object',
    'subtype' => 'my_blog',
));

echo elgg_view_page($titlebar, [
    'title' => $page_title,
    'content' => $body,
]);
```

La fonction `elgg_list_entities` prend les derniers articles `my_blog` et les passe à la vue d'affichage de l'objet. Notez que cette fonction ne retourne que les articles que l'utilisateur a le droit de voir, aussi le contrôle d'accès est géré de manière transparente. La fonction (et ses cousines) gère également de manière transparente la pagination, et crée même un flux RSS pour votre `my_blogs` si vous avez défini cette vue.

La fonction de liste peut également restreindre les articles `my_blog` à ceux d'un utilisateur spécifique. Par exemple, la fonction `elgg_get_logged_in_user_guid` récupère l'identifiant unique (GUID) de l'utilisateur connecté, et en le passant à `elgg_list_entities` la liste n'affichera que les articles de l'utilisateur connecté :

```
echo elgg_list_entities(array(
    'type' => 'object',
    'subtype' => 'my_blog',
    'owner_guid' => elgg_get_logged_in_user_guid()
));
```

Puis vous devez modifier votre gestionnaire de page `my_blog` pour récupérer la nouvelle page quand l'URL est `/my_blog/all`. Modifiez la fonction `my_blog_page_handler` dans `start.php` pour qu'elle ressemble à ceci :

```
function my_blog_page_handler($segments) {
    switch ($segments[0]) {
        case 'add':
            echo elgg_view_resource('my_blog/add');
            break;

        case 'view':
            $resource_vars['guid'] = elgg_extract(1, $segments);
            echo elgg_view_resource('my_blog/view', $resource_vars);
            break;

        case 'all':
        default:
            echo elgg_view_resource('my_blog/all');
            break;
    }

    return true;
}
```

Maintenant, si l'URL contient `/my_blog/all`, l'utilisateur verra une page « Tous les My_Blogs du site ». Grâce au cas par défaut, la liste de tous les `my_blogs` sera également affichée si l'URL n'est pas valide, comme par exemple `/my_blog`` ou `/my_blog/xyz`.

Vous pouvez aussi choisir de mettre à jour la vue de l'objet pour gérer différents types d'affichage, faute de quoi la liste de tous les `my_blogs` va également afficher le contenu complet de tous les `my_blogs`. Modifiez `/mod/my_blog/views/default/object/my_blog.php` pour qu'il ressemble à ceci :

```
<?php
$full = elgg_extract('full_view', $vars, FALSE);

// full view
if ($full) {
    echo elgg_view('output/longtext', array('value' => $vars['entity']->description));
    echo elgg_view('output/tags', array('tags' => $vars['entity']->tags));

// list view or short view
} else {
    // make a link out of the post's title
    echo elgg_view_title(
        elgg_view('output/url', array(
            'href' => $vars['entity']->getURL(),
            'text' => $vars['entity']->getDisplayName(),
            'is_trusted' => true,
        )))
    );
```

(suite sur la page suivante)

(suite de la page précédente)

```

echo elgg_view('output/tags', array('tags' => $vars['entity']->tags));
}

```

Désormais, si `full_view` est à `true` (tel qu'il avait été préalablement défini dans [cette section](#)), la vue de l'objet va afficher le contenu et les tags de l'article (le titre est affiché par `view.php`). Sinon la vue de l'objet ne va afficher que le titre et les tags de l'article.

FIN

Il y a tant d'autres choses qui peuvent être faites, mais espérons que ceci vous donne une bonne idée de comment démarrer.

3.4.4 Intégrer un éditeur de texte visuel (Rich Text Editor)

Construisez votre propre plugin wysiwyg.

Elgg est distribué avec un plugin pour [CKEditor](#), et précédemment distribué avec le support de TinyMCE. Cependant, s'il y a un éditeur wysiwyg que vous préférez, vous pourriez utiliser ce tutoriel pour construire le vôtre.

Tous les formulaires dans Elgg devraient essayer d'utiliser les vues de saisie situées dans `views/default/input`. Si ces vues sont utilisées, il est plus aisé pour les auteurs de plugins de remplacer une vue, ici `input/longtext`, par leur vue avec wysiwyg.

Ajoutez la bibliothèque de code WYSIWYG

Maintenant vous devez charger TinyMCE dans un répertoire de votre plugin. Nous recommandons vivement que vous utilisiez `composer` pour gérer les bibliothèques tierces, car il est beaucoup plus facile de gérer la maintenance et les montées de version de cette manière

```
.. code-block:: sh
```

```
composer nécessite bower-asset/tinymce
```

Dites à Elgg quand et comment charger TinyMCE

Maintenant que vous avez :

- créé le fichier `start`
- initialisé le plugin
- chargé le code wysiwyg

Il est temps de dire à Elgg comment appliquer TinyMCE aux champs de saisie de texte.

Nous allons faire ceci en étendant la vue `input/longtext` et en incluant un peu de JavaScript. Créez une vue `tinymce/longtext` et ajoutez le code suivant :

```

<?php

/**
 * Elgg long text input with the tinymce text editor intact
 * Displays a long text input field
 */

```

(suite sur la page suivante)

```

?>
<!-- include tinymce -->
<script language="javascript" type="text/javascript" src="<?php echo $vars['url']; ?>mod/
→tinymce/tinymce/jscripts/tiny_mce/tiny_mce.js"></script>
<!-- initialise tinymce, you can find other configurations here http://wiki.moxiecode.com/
→examples/tinymce/installation_example_01.php -->
<script language="javascript" type="text/javascript">
    tinyMCE.init({
        mode : "textareas",
        theme : "advanced",
        theme_advanced_buttons1 : "bold,italic,underline,separator,striktthrough,justifyleft,
→justifycenter,justifyright, justifyfull,bulldist,numldist,undo,redo,link,unlink,image,
→blockquote,code",
        theme_advanced_buttons2 : "",
        theme_advanced_buttons3 : "",
        theme_advanced_toolbar_location : "top",
        theme_advanced_toolbar_align : "left",
        theme_advanced_statusbar_location : "bottom",
        theme_advanced_resizing : true,
        extended_valid_elements : "a[name|href|target|title|onclick],
→img[class|src|border=0|alt|title|hspace|vspace|width|height|align|onmouseover|onmouseout|name],
→
        hr[class|width|size|noshade],font[face|size|color|style],span[class|align|style]"
    });
</script>

```

Puis, dans la fonction init de votre plugin, étendez la vue input/longtext

```

function tinymce_init() {
    elgg_extend_view('input/longtext', 'tinymce/longtext');
}

```

Et voilà ! Désormais chaque fois que quelqu'un utilise input/longtext, TinyMCE sera chargé et appliqué à cette zone de texte.

3.4.5 Widget basique

Créez un widget qui va afficher "Hello, World !" ainsi que n'importe quel texte souhaité par l'utilisateur.

Dans Elgg, les widgets sont ces composants que vous pouvez déplacer sur votre page de profil ou le tableau de bord d'administration.

Ce tutoriel suppose que vous êtes familier(ère) des concepts de base d'Elgg tels que :

- *Vues*
- *Plugins*

Vous devriez les revoir si cela devient confus en cours de route.

Contenu

- *Ajouter le code de la vue du widget*
- *Enregistrer votre widget*

— Permettre les personnalisations par l'utilisateur

Ajouter le code de la vue du widget

Elgg scanne automatiquement certains répertoires des plugins pour trouver des fichiers spécifiques. Les vues [Vues](#) facilitent l'ajout de votre propre code d'affichage, ou la possibilité de faire d'autres choses telles que surcharger le comportement par défaut d'Elgg. Pour le moment, nous allons simplement ajouter le code d'affichage pour votre widget. Créez un fichier `/views/default/widgets/helloworld/content.php`. "helloworld" sera le nom de votre widget à l'intérieur du plugin hello. Dans ce fichier ajoutez le code :

```
<?php  
  
echo "Hello, world!";
```

Ceci va ajouter ces mots au canevas du widget lorsqu'il sera rendu. Elgg se charge de charger le widget.

Enregistrer votre widget

Elgg a besoin qu'on lui indique explicitement que le plugin contient un widget pour qu'il vérifie le répertoire des vues du widget. Ceci est fait en appelant la fonction `elgg_register_widget_type()`. Modifiez `/start.php` et ajoutez dedans ces lignes :

```
<?php  
  
function hello_init() {  
    elgg_register_widget_type([  
        'id' => 'helloworld',  
        'name' => 'Hello, world!',  
        'description' => 'The "Hello, world!" widget',  
    ]);  
}  
  
return function() {  
    elgg_register_event_handler('init', 'system', 'hello_init');  
}
```

Rendez-vous maintenant sur votre page de profil avec un navigateur web et ajoutez le widget "hello, world". Il devrait afficher "Hello, world!".

Note : Pour de vrais widgets, c'est toujours une bonne idée de respecter [Internationalisation](#).

Permettre les personnalisations par l'utilisateur

Cliquez sur le lien d'ajout dans la barre d'outils du widget que vous avez créé. Vous allez noter que le seul contrôle qu'il vous offre par défaut est sur le niveau d'accès (sur qui peut voir le widget).

Supposez que vous voulez permettre à l'utilisateur de contrôler quel message d'accueil est affiché dans le widget. De la même manière qu'Elgg charge automatiquement `content.php` pour afficher un widget, il charge `edit.php` quand un utilisateur tente de modifier un widget. Ajoutez le code suivant dans `/views/default/widgets/helloworld/edit.php` :

```
<div>
  <label>Message:</label>
  <?php
    //This is an instance of the ElggWidget class that represents our widget.
    $widget = $vars['entity'];

    // Give the user a plain text box to input a message
    echo elgg_view('input/text', array(
      'name' => 'params[message]',
      'value' => $widget->message,
      'class' => 'hello-input-text',
    ));
  ?>
</div>
```

Notez la relation entre les valeurs passées aux champs “name” (nom) et “value” (valeur) d’input/text. Le nom du champ de la boîte de saisie de texte est `params[message]` parce qu’Elgg va automatiquement gérer les variables des widgets placées dans le tableau `params`. Le nom de la variable PHP correspondante sera `message`. Si nous voulions utiliser le champ `greeting` au lieu de `message` nous passerions respectivement les valeurs `params[greeting]` et `$widget->greeting`.

La raison pour laquelle nous définissons l’option “value” du tableau est que ceci indique à la vue d’édition de se souvenir de ce que l’utilisateur a saisi la dernière fois qu’il a modifié la valeur du texte du message.

Maintenant pour afficher le message de l’utilisateur nous devons modifier `content.php` pour qu’il utilise cette variable `message`. Editez `/views/default/widgets/helloworld/content.php` et modifiez-le pour :

```
<?php

$widget = $vars['entity'];

// Always use the corresponding output/* view for security!
echo elgg_view('output/text', array('value' => $widget->message));
```

Vous devriez maintenant pouvoir saisir un message dans la boîte de texte et le voir apparaître dans le widget.

3.5 Docs de conception

Acquérez une compréhension profonde de comment fonctionne Elgg et de pourquoi il est construit de cette manière.

3.5.1 Actions

Les actions sont la première modalité d'interaction des utilisateurs avec un site Elgg.

Aperçu

Dans Elgg, une action est le code qui est exécuté pour faire des modifications dans la base de données quand un utilisateur effectue quelque chose. Par exemple, le fait de se connecter, de publier un commentaire, ou de créer un article de blog sont des actions. Le script d'action traite le processus d'entrée, effectue les modifications appropriées dans la base de données, et fournit un retour à l'utilisateur à propos de l'action.

Gestionnaire d'action

Les actions sont enregistrées durant le processus de démarrage (boot) en appelant `elgg_register_action()`. Toutes les URLs d'action commencent par `action/` et sont servies par le contrôleur front end d'Elgg à travers le service action. Cette approche est différente d'application PHP traditionnelles qui envoient l'information à un fichier spécifique. Le service action réalise des *CSRF vérifications de sécurité*, et appelle le fichier de script d'action enregistré, puis renvoie éventuellement l'utilisateur vers une nouvelle page. En utilisant le service action au lieu d'un seul fichier de script, Elgg fournit automatiquement une sécurité et une extensibilité améliorées.

Voyez *Formulaires + Actions* pour plus de détails sur comment enregistrer et construire une action. Pour regarder les actions du noyau, regardez dans le répertoire `/actions`.

3.5.2 Base de données

Une discussion solide sur le design du modèle de données Elgg et ses motivations.

Contenu

- *Aperçu*
- *Modèle de données (datamodel)*
- *Entités*
 - *Types*
 - *Sous-types (subtypes)*
 - *Trucs à savoir sur les sous-types*
 - *GUIDs*
- *ElggObject*
- *ElggUser*
- *ElggSite*
- *ElggGroup*
 - *Le plugin Groups*
 - *Écrire un plugin conscient des groupes*
- *Propriété*
- *Conteneurs (containers)*
- *Annotations*

- *Ajouter une annotation*
- *Lire les annotations*
- *Fonctions d'aide utiles*
- *Métadonnées*
 - *Le cas simple*
 - *Lire des métadonnées comme des objets*
 - *Erreurs courantes*
- *Relations*
 - *Travailler avec des relations*
- *Contrôle d'accès*
 - *Niveau d'accès dans le modèle de données*
 - *Comment les accès affectent la récupération de données*
 - *Accès en écriture*
- *Schéma*
 - *InnoDB*
 - *Tables principales*
 - *Tables secondaires*

Aperçu

Dans Elgg, tout fonctionne sur un modèle de données unifié construit sur des unités atomiques de données appelées entités.

Il est recommandé que les plugins n'interagissent pas directement avec la base de données, afin de créer un système plus stable et une meilleure expérience utilisateur parce que les contenus créés par différents plugins peut être mélangés ensemble de manières cohérentes. Avec cette approche, les plugins sont plus rapides à développer, et sont en même temps plus puissants.

Chaque entité dans le système hérite de la classe `ElggEntity`. Cette classe contrôle les permissions d'accès, la propriété, le conteneur, et fournit une API cohérente pour accéder aux propriétés des entités et les modifier.

Vous pouvez étendre les entités avec des informations supplémentaires de deux manières :

Metadata : Ces informations décrivent l'entité, elles sont habituellement ajoutées par l'auteur de l'entité quand l'entité est créée ou modifiée. Des exemples de métadonnées incluent les tags, un numéro ISBN ou un ID tierce-partie, une adresse, des coordonnées géographiques, etc. Imaginez les métadonnées comme un simple système de stockage clef-valeur.

Annotations : Ces informations étendent l'entité avec des propriétés, habituellement ajoutées par une tierce-partie. De telles propriétés comprennent les notations, likes et votes.

La principale différence entre métadonnées et annotations :

- les métadonnées n'ont pas de propriétaire, alors que les annotations en ont
- les métadonnées ne disposent pas de contrôle d'accès, alors que les annotations en disposent
- les métadonnées sont préchargées lorsque l'entité est construite, alors que les annotations ne sont chargées que sur demande

Ces différences peuvent avoir des implications sur les performances et votre logique applicative, aussi considérez avec soin la manière dont vous souhaitez attacher des données à vos entités.

Dans certains cas, il peut être préférable d'éviter d'utiliser ces métadonnées et annotations et de créer de nouvelles entités à la place, puis de les attacher via `container_guid` ou une relation.

Modèle de données (datamodel)

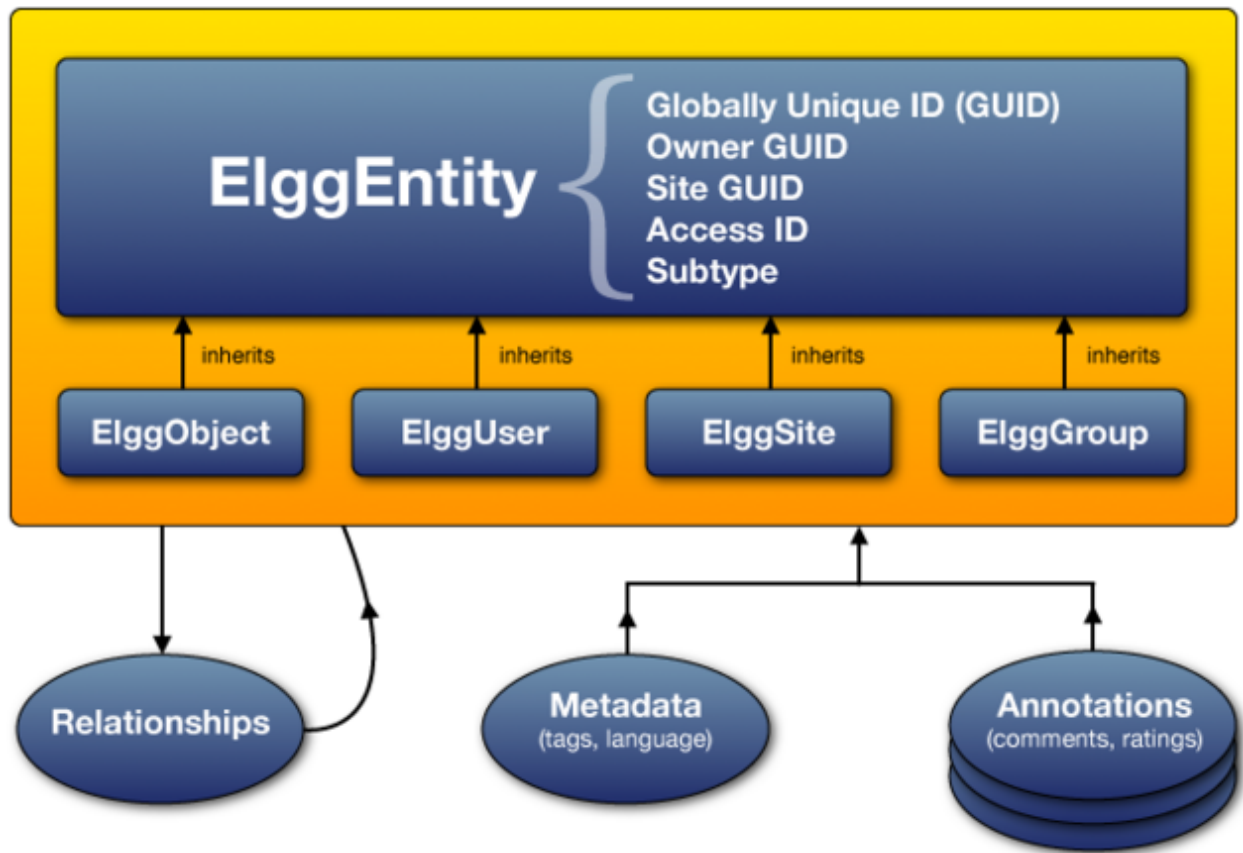


Fig. 9 – Le diagramme du modèle de données d’Elgg

Entités

ElggEntity est la classe de base pour le modèle de données d’Elgg et supporte un jeu de propriétés et méthodes communes.

- Un identifiant global unique (Globally Unique Identifier - Voir *GUIDs*).
- Permissions d’accès . (Quand un plugin demande des données, il n’accède jamais à des données que l’utilisateur actuel n’a pas la permission de voir.)
- Un sous-type arbitraire (plus d’informations ci-dessous).
- Un propriétaire (owner).
- Le site auquel appartient cette entité.
- Un conteneur, utilisé pour associer le contenu avec un groupe ou un utilisateur.

Types

Les *véritables* entités seront des instances de quatre sous-classes différentes, chacune ayant une propriété **type** distincte et leur propres propriétés et méthodes additionnelles.

Type	Classe PHP	Représente
objet	ElggObject	Les plupart des contenus créés par les utilisateurs, tels que des articles de blog, des charge-ments de fichiers, et des signets.
groupe	ElggGroup	Un groupe organisé d'utilisateurs avec sa propre page de profil
utilisa-teur	ElggUser	Un utilisateur du système
site	ElggSite	Le site servi par l'installation Elgg

Chaque type a sa propre API étendue. Par ex. les utilisateurs peuvent être en contact avec d'autres utilisateurs, un groupe peut avoir des membres, tandis que les objets peuvent être « likés » et commentés.

Sous-types (subtypes)

Chaque entité doit définir un **sous-type** (subtype), que les plugins utilisent pour spécialiser encore l'entité. Elgg rend facile le fait de rechercher des entités spécifiques d'un ou de plusieurs sous-types donnés, ainsi que de leur assigner des comportements et vues spéciales.

Les sous-types sont communément donnés aux instances de `ElggEntity` pour définir le type de contenu créé. Par ex. le plugin blog crée des objets avec le sous-type "blog".

Par défaut, les utilisateurs, les groupes et les sites ont respectivement le sous-type `user`, `group` et `site`.

Les plugins peuvent utiliser des classes d'entités personnalisées qui étendent la classe de base du type. Pour cela, ils doivent enregistrer leur classe au démarrage (par ex. dans le gestionnaire `'init', 'system'`), en utilisant `elgg_set_entity_class()`. Par exemple, le plugin blog pourrait utiliser `elgg_set_entity_class('object', 'blog', \ElggBlog::class)`.

Les plugins peuvent utiliser `elgg-plugin.php` pour définir une classe d'entité via le paramètre raccourci `entities`.

Trucs à savoir sur les sous-types

- Avant que la méthode `save()` d'une entité soit appelée, le sous-type doit être défini en écrivant une chaîne de caractère dans la propriété `subtype`.
- *Le sous-type ne peut pas être changé après l'enregistrement.*

GUIDs

Un GUID est un entier qui définit de manière unique chaque entité dans une installation Elgg (un IDentifiant Global Unique - Globally Unique IDentifier). Il est assigné automatiquement la première fois qu'une entité est enregistré, et ne peut jamais être changé.

Certaines fonctions de l'API Elgg fonctionnent avec des GUIDs au lieu d'objets `ElggEntity`.

ElggObject

Le type d'entité `ElggObject` représente un type de contenu arbitraire au sein d'une installation Elgg ; des choses telles que des articles de blog, des fichiers, etc.

Au-delà des propriétés standards de `ElggEntity`, `ElggObjects` supporte également :

- `title` Le titre de l'objet (texte sans HTML)
- `description` Une description de l'objet (HTML)

La plupart des autres données à propos de l'objet sont généralement stockées via des métadonnées.

ElggUser

Le type d'entité `ElggUser` représente les utilisateurs au sein d'une installation Elgg. Ils seront définis comme désactivés jusqu'à ce que leur compte ait été activé (à moins qu'ils n'aient été créés à partir du panneau d'administration).

Au-delà des propriétés standards de `ElggEntity`, `ElggUsers` supporte également :

- `name` Le nom de l'utilisateur en texte brut. Par ex. « Hugh Jackman »
- `username` Leur nom d'utilisateur. Par ex. « hjackman »
- `password` Une version hachée de leur mot de passe
- `email` Leur adresse email
- `language` Leur code de langue par défaut.
- `code` Leur code de session (déplacé vers une table séparée dans la 1.9).
- `last_action` Le timestamp UNIX de la dernière fois qu'ils ont chargé une page
- `prev_last_action` La précédente valeur de `last_action`
- `last_login` Le timestamp UNIX de leur dernière connexion
- `prev_last_login` la précédente valeur de `last_login`

ElggSite

Le type d'entité `ElggSite` représente votre installation Elgg (via l'URL de votre site).

Au-delà des propriétés standards de `ElggEntity`, `ElggSites` supporte également :

- `name` Le nom du site
- `description` Une description du site
- `url` L'adresse du site

ElggGroup

Le type d'entité `ElggGroup` représente une association d'utilisateurs Elgg. Les utilisateurs peuvent rejoindre, quitter les groupes, et y publier du contenu.

Au-delà des propriétés standards de `ElggEntity`, `ElggGroups` supporte également :

- `name` Le nom du groupe (texte sans HTML)
- `description` Une description du groupe (HTML)

`ElggGroup` a des méthodes additionnelles pour gérer le contenu et les adhésions.

Le plugin Groups

A ne pas confondre avec le type d'entité `ElggGroup`, Elgg vient avec un plugin appelé « Groups » qui fournit une UI/UX par défaut pour que les utilisateurs du site interagissent avec les groupes. Chaque groupe dispose d'un forum de discussion et d'une page de profil qui relie les utilisateurs au contenu dans le groupe..

Vous pouvez modifier l'expérience utilisateur via les moyens traditionnels d'extension de plugin, ou remplacer complètement le plugin Groups par votre propre plugin.

Écrire un plugin conscient des groupes

Les développeurs de plugins ne devraient pas trop s'inquiéter d'écrire une fonctionnalité consciente des groupes, mais il y a quelques points clefs :

Ajouter du contenu

En passant le groupe en tant que `container_guid` via un champ de saisie caché, vous pouvez utiliser un seul formulaire et une seule action pour ajouter du contenu à la fois pour un utilisateur ou pour un groupe.

Utilisez `ElggEntity->canWriteToContainer()` pour déterminer si l'utilisateur actuel a ou n'a pas le droit d'ajouter du contenu dans un groupe.

Soyez attentif au fait que vous allez devoir passer le GUID du conteneur ou le nom d'utilisateur à la page responsable de la publication et la valeur associée, de sorte que ceci puisse être ensuite conservé dans votre formulaire sous forme de champ de saisie caché, pour un passage plus aisé vers vos actions. Au sein d'une action « create », vous allez avoir besoin de récupérer ce champ de saisie et de l'enregistrer sous forme de propriété de votre nouvel élément (la valeur par défaut est le conteneur actuel de l'utilisateur) :

```
$user = elgg_get_logged_in_user_entity();
$container_guid = (int)get_input('container_guid');

if ($container_guid) {
    $container = get_entity($container_guid);

    if (!$container->canWriteToContainer($user->guid)) {
        // register error and forward
    }
} else {
    $container_guid = elgg_get_logged_in_user_guid();
}

$object = new ElggObject;
$object->container_guid = $container_guid;

...

$container = get_entity($container_guid);
forward($container->getURL());
```

Propriété

Les entités ont une propriété GUID `owner_guid`, qui définit son propriétaire. Typiquement ceci renvoie au GUID d'un utilisateur, quoique les sites et les utilisateurs eux-même n'ont souvent pas de propriétaire (valeur de 0).

La propriété d'une entité détermine, d'une part, si vous pouvez ou non accéder à, ou modifier cette entité.

Conteneurs (containers)

Afin de rechercher aisément du contenu par groupe ou utilisateur, le contenu est généralement défini comme « contenu » soit par l'utilisateur qui l'a publié, soit par le groupe dans lequel l'utilisateur l'a publié. Ceci signifie que la propriété `container_guid` des nouveaux objets sera définie avec la valeur du `ElggUser` actuel ou du `ElggGroup` cible.

Par ex., trois articles de blog peuvent avoir des propriétaires différentes, mais être tous contenus dans le groupe où ils ont été publiés.

Note : Ceci n'est pas toujours vrai. Les entités Commentaires sont contenues par l'objet commenté, et dans certains plugins tierce-partie la conteneur peut être utilisé pour modéliser des relations parents-enfants entre entités (par ex. un objet « dossier » qui contient un objet fichier).

Annotations

Les annotations sont des éléments de données attachées à une entité qui permet aux utilisateurs de laisser des évaluations, ou d'autres types de réactions pertinentes. Un plugin de sondage pourrait enregistrer des votes sous forme d'annotations.

Les annotations sont stockées sous formes d'instances de la classe `ElggAnnotation`.

Chaque annotation dispose de :

- Un type d'annotation interne (tel que *comment*)
- Une valeur (qui peut être une chaîne de caractères ou un entier)
- Un niveau d'accès distinct de celui de l'entité à laquelle il est attaché
- Un propriétaire

Comme pour les métadonnées, les valeurs sont stockées sous forme de chaînes de caractères à moins que la valeur donnée soit un entier PHP (`is_int($value)` vaut true), ou à moins que `$vartype` soit spécifié manuellement comme `integer`.

Ajouter une annotation

La manière la plus simple d'ajouter une annotation est d'utiliser la méthode `annotate` sur une entité, qui est définie comme :

```
public function annotate(
    $name,           // The name of the annotation type (eg 'comment')
    $value,          // The value of the annotation
    $access_id = 0,  // The access level of the annotation
    $owner_id = 0,   // The annotation owner, defaults to current user
    $vartype = ""    // 'text' or 'integer'
)
```

Par exemple, pour donner une notation à une entité, vous pouvez appeler :

```
$entity->annotate('rating', $rating_value, $entity->access_id);
```

Lire les annotations

Pour récupérer les annotations d'une entité, vous pouvez appeler la méthode suivante :

```
$annotations = $entity->getAnnotations(  
    $name,    // The type of annotation  
    $limit,    // The number to return  
    $offset,  // Any indexing offset  
    $order,    // 'asc' or 'desc' (default 'asc')  
);
```

Si votre type d'annotation utilise largement des valeurs entières, plusieurs fonctions mathématiques utiles sont fournies :

```
$averagevalue = $entity->getAnnotationsAvg($name); // Get the average value  
$total = $entity->getAnnotationsSum($name);        // Get the total value  
$minvalue = $entity->getAnnotationsMin($name);     // Get the minimum value  
$maxvalue = $entity->getAnnotationsMax($name);     // Get the maximum value
```

Fonctions d'aide utiles

Commentaires

Si vous souhaitez fournir une fonctionnalité de commentaire sur les objets de votre plugin, la fonction suivante va fournir le listing complet, le formulaire et les actions :

```
function elgg_view_comments(ElggEntity $entity)
```

Métadonnées

Les métadonnées dans Elgg vous permettent de stocker les données supplémentaires d'une entité au-delà des champs pré-définis que cette entité supporte. Par exemple, `ElggObjects` ne supporte que les champs d'entités basiques plus un titre et une description, mais vous pouvez souhaiter inclure également des tags ou un numéro ISBN. De manière similaire, vous pourriez souhaiter que les utilisateurs puissent enregistrer une date de naissance.

Sous le capot, les métadonnées sont stockées sous forme d'instance de la classe `ElggMetadata`, mais vous n'avez pas à vous soucier de cela en pratique (quoique si vous êtes intéressé, voyez la référence de la classe `ElggMetadata`). Ce que vous avez besoin de savoir :

- Une métadonnée a un propriétaire, qui peut être différent du propriétaire de l'entité à laquelle elle est attachée
- Vous pouvez potentiellement avoir plusieurs éléments de chaque type de métadonnée attachés à la même entité
- Comme pour les annotations, les valeurs sont stockées sous forme de chaîne de caractères à moins que la valeur donnée soit un entier PHP (`is_int($value)` vaut true), ou à moins que le `$value_type` soit défini manuellement comme `integer` (voyez ci-dessous).

Note : A partir de Elgg 3.0, les métadonnées n'ont plus d'`access_id`.

Le cas simple

Ajouter des métadonnées

Pour ajouter une métadonnée à une entité, appelez simplement :

```
$entity->metadata_name = $metadata_value;
```

Par exemple, pour ajouter une date d'anniversaire à un utilisateur :

```
$user->dob = $dob_timestamp;
```

Ou pour ajouter des tags à un objet :

```
$object->tags = array('tag one', 'tag two', 'tag three');
```

Lorsque vous ajoutez une métadonnée de cette manière :

- Le propriétaire est assigné à l'utilisateur actuellement identifié
- Réassigner une métadonnée va remplacer l'ancienne valeur

Ceci convient pour la plupart des objectifs. Faites attention à bien noter quels attributs sont des métadonnées et lesquels sont natifs au type d'entité avec lequel vous travaillez. Vous n'avez pas besoin d'enregistrer une entité après avoir ajouté ou modifié des métadonnées. Vous devez enregistrer une entité si vous avez changé l'un des ses attributs natifs. A titre d'exemple, si vous avez changé l'id d'accès d'un ElggObject, vous devez l'enregistrer, faute de quoi la modification ne sera pas conservée dans la base de données.

Note : A partir de Elgg 3.0, la propriété `access_id` des métadonnées est ignorée.

Lire les métadonnées

Pour récupérer une métadonnée, traitez-la comme une propriété d'une entité :

```
$tags_value = $object->tags;
```

Notez que ceci va retourner la valeur absolue de la métadonnée. Pour récupérer des métadonnées sous forme d'objet ElggMetadata, vous aurez besoin d'utiliser les méthodes décrites dans la section *contrôle plus fin* ci-dessous.

Si vous stockez de multiples valeurs dans cette métadonnée (comme dans l'exemple « tags » ci-dessus), vous obtiendrez un tableau de toutes ces valeurs. Si vous stockez seulement une valeur, vous récupérerez une chaîne de caractères ou un entier. Stocker un tableau avec seulement une valeur vous retournera une chaîne de caractères. Par ex.

```
$object->tags = array('tag');
$tags = $object->tags;
// $tags will be the string "tag", NOT array('tag')
```

Pour récupérer toujours un tableau, castez simplement vers un tableau :

```
$tags = (array)$object->tags;
```

Lire des métadonnées comme des objets

`elgg_get_metadata` est la meilleure fonction pour récupérer les métadonnées sous forme d'objets `ElggMetadata` :

Par ex., pour récupérer la date de naissance d'un utilisateur

```
elgg_get_metadata(array(
    'metadata_name' => 'dob',
    'metadata_owner_guid' => $user_guid,
));
```

Ou pour récupérer toutes les objets des métadonnées :

```
elgg_get_metadata(array(
    'metadata_owner_guid' => $user_guid,
    'limit' => 0,
));
```

Erreurs courantes

« Ajouter » des métadonnées

Notez que vous ne pouvez pas « ajouter » des valeurs aux tableaux de métadonnées comme si c'étaient des tableaux php normaux. Par exemple, le code suivant ne fera pas ce qu'il semblerait qu'il devrait faire.

```
$object->tags[] = "tag four";
```

Essayer de stocker des tableaux indexés

Elgg ne supporte pas le stockage de tableaux indexés (paires clef/valeur) dans les métadonnées. Par exemple, le code suivant ne fonctionne pas comme vous pourriez le penser de prime abord :

```
// Won't work!! Only the array values are stored
$object->tags = array('one' => 'a', 'two' => 'b', 'three' => 'c');
```

Au lieu de cela, vous pouvez conserver cette information de cette manière :

```
$object->one = 'a';
$object->two = 'b';
$object->three = 'c';
```

Conserver des GUIDs dans des métadonnées

Quoiqu'il existe certains cas pour stocker les GUIDs d'entités dans des métadonnées, les relations *Relationships* sont une construction bien plus adaptée pour relier les entités les unes aux autres.

Relations

Les relations vous permettent d'associer des entités ensemble. Exemples : un artiste a des fans, un utilisateur est membre d'une organisation, etc.

La classe `ElggRelationship` modélise une relation directe entre deux entités, en faisant la déclaration :

« **{subject}** est un **{noun}** de **{target}**. »

Nom de l'API	Modèles	Représente
<code>guid_one</code>	Le sujet	Quelle entité est reliée
<code>relationship</code>	Le nom	Le type de relation
<code>guid_two</code>	La cible	L'entité à laquelle le sujet est relié

Le type de relation peut également être un verbe, faisant la déclaration :

« **{sujet}** **{verbe}** **{cible}**. »

Par ex. Utilisateur A « aime » l'article de blog B

Chaque relation a une direction. Imaginez un archer qui tire une flèche vers une cible ; la flèche se déplace dans une direction, en reliant le sujet (l'archer) à la cible.

Une relation n'implique pas de réciprocité. A suit B n'implique pas que B suive A.

Les **Relations** **n'ont pas de contrôle d'accès**. Elles ne sont jamais cachées d'une vue et peuvent être modifiées par le code à n'importe quel niveau de privilège, avec l'avertissement que *les entités* d'une relation peuvent être invisible en raison du contrôle d'accès !

Travailler avec des relations

Créer une relation

Par exemple pour établir que « **\$utilisateur** est un **fan** de **\$artiste** » (utilisateur est le sujet, artiste est la cible) :

```
// option 1
$success = add_entity_relationship($user->guid, 'fan', $artist->guid);

// option 2
$success = $user->addRelationship($artist->guid, 'fan');
```

Ceci déclenche l'événement [create, relationship], en lui passant l'objet `ElggRelationship` créé. Si un gestionnaire retourne `false`, la relation ne sera pas créée et `$success` vaudra `false`.

Vérifier une relation

Par exemple pour vérifier que « **\$utilisateur** est un **fan** de **\$artiste** » :

```
if (check_entity_relationship($user->guid, 'fan', $artist->guid)) {
    // relationship exists
}
```

Notez que si la relation existe, `check_entity_relationship()` retourne un objet `ElggRelationship` :

```
$relationship = check_entity_relationship($user->guid, 'fan', $artist->guid);
if ($relationship) {
    // use $relationship->id or $relationship->time_created
}
```

Supprimer une relation

Par exemple pour pouvoir affirmer que « **\$utilisateur** n'est désormais plus un **fan** de **\$artiste** » :

```
$was_removed = remove_entity_relationship($user->guid, 'fan', $artist->guid);
```

Ceci déclenche l'événement [delete, relationship], en lui passant l'objet ElggRelationship associé. Si un gestionnaire retourne false, la relation sera conservée, \$was_removed vaudra false.

Autres fonctions utiles :

- delete_relationship() : supprimer par ID
- remove_entity_relationships() : supprime ceux qui sont liés à une entité

Trouver des relations et des entités associées

Voici ci-dessous quelques fonctions pour récupérer des relations entre objets et/ou des entités reliées :

- get_entity_relationships() : récupère les relations par entité sujet ou entité cible
- get_relationship() : récupère un objet relation par ID
- elgg_get_entities() : récupère les entités dans les relations de différentes manières

Par ex. retrouver les utilisateurs qui ont rejoint votre groupe en janvier 2014.

```
$entities = elgg_get_entities(array(
    'relationship' => 'member',
    'relationship_guid' => $group->guid,
    'inverse_relationship' => true,

    'relationship_created_time_lower' => 1388534400, // January 1st 2014
    'relationship_created_time_upper' => 1391212800, // February 1st 2014
));
```

Contrôle d'accès

Les contrôle d'accès granulaires sont l'un des principes de design fondamentaux dans Elgg, et une fonctionnalité qui a été au cœur du système tout au long de son développement. L'idée est simple : un utilisateur devrait avoir un contrôle total sur qui peut voir un élément de donnée qu'il ou elle a créé.

Niveau d'accès dans le modèle de données

Pour parvenir à cela, chaque entité et annotation contient une propriété `access_id`, qui correspond à l'un des niveau d'accès prédéfinis ou à une entrée dans la table `access_collections` de la base de données.

Niveaux d'accès pré-définis

- `ACCESS_PRIVATE` (valeur : 0) Privé.
- `ACCESS_LOGGED_IN` (value : 1) Logged in users.
- `ACCESS_PUBLIC` (value : 2) Public data.

Niveaux d'accès définis par l'utilisateur

Vous pouvez définir des groupes d'accès additionnels et les assigner à une entité, ou une annotation. Un certain nombre de fonctions ont été définies pour vous y aider; voyez *Listes de Contrôle d'Accès* pour plus d'informations.

Comment les accès affectent la récupération de données

Toutes les récupérations de données fonctionnent au-dessus de la couche de la base de données - par exemple `elgg_get_entities` ne retournera que les éléments que l'utilisateur actuel a l'autorisation de voir. Il n'est pas possible de récupérer des éléments auxquels l'utilisateur actuel n'a pas accès. Ceci rend très difficile de créer un trou de sécurité pour récupérer des données.

Accès en écriture

Les règles suivantes gouvernent les accès en écriture :

- Le propriétaire d'une entité peut toujours la modifier
- Le propriétaire d'un conteneur peut modifier tout ce qui est dedans (notez que cela ne signifie pas que le propriétaire d'un groupe peut modifier toutes les publications dans ce groupe)
- Les administrateurs peuvent tout éditer

Vous pouvez surcharger ce comportement en utilisant un *hook plugin* appelé `permissions_check`, qui passe l'entité en question à toute fonction qui a annoncé qu'elles veut être référencée. Retourner `true` va autoriser l'accès en écriture; retourner `false` va l'interdire. Voyez *les références des plugin hook pour les permissions_check* pour plus de détails.

Schéma

La base de données contient un certain nombre de tables primaires et secondaires. Vous pouvez suivre les évolutions du schéma dans `engine/schema/migrations/`

L'ensemble de caractères de la base de données doit être `utf8mb4`, ce qui fournira une prise en charge complète des caractères unicode lors du stockage des données.

InnoDB

À partir d'Elgg 3.0, la base de données utilise le moteur InnoDB. Pour une installation ou une migration correcte, certains paramètres peuvent devoir être ajustés dans les paramètres de la base de données.

- `innodb_large_prefix` devrait être on
- `innodb_file_format` devrait être Barracuda
- `innodb_file_per_table` devrait être 1

Tables principales

Voici la description des tables principales. Gardez à l'esprit que pour une installation d'Elgg données les tables vont avoir un préfixe (typiquement « `elgg_` »).

Table : `entities` (entités)

Ceci est la table principale *Entities* qui contient les utilisateurs, sites, objets et groupes Elgg. Quand vous installez Elgg la première fois elle est automatiquement peuplée avec votre premier site.

Elles contient les champs suivants :

- **guid** Un compteur auto-incrémenté qui produit un GUID qui identifie de manière unique cette entité dans le système
- **type** Le type d'entity - object, user, group ou site
- **subtype** Le sous-type d'entité
- **owner_guid** Le GUID de l'entité du propriétaire
- **container_guid** Le GUID dans laquelle cette entité est contenue - soit un utilisateur (user) soit un groupe (group)
- **access_id** Contrôles d'accès sur cette entité
- **time_created** Le timestamp Unix de création de l'entité
- **time_updated** Le timestamp Unix de la dernière modification de l'entité
- **enabled** Si oui ("yes") l'entité est accessible, si non ("no") l'entité a été désactivée (Elgg traite cela comme si elle avait été supprimée sans toutefois la supprimer réellement de la base de données)

Table : `metadata`

Cette table contient des métadonnées *Metadata*, des informations supplémentaires attachées à une entité.

- **id** Un IDentifiant unique
- **entity_guid** L'entité à laquelle cette métadonnée est attachée
- **name** Le nom de la chaîne
- **value** La valeur de la chaîne
- **value_type** La classe de la valeur, soit texte (text) ou un entier (integer)
- **time_created** Le timestamp Unix de la date de création de la métadonnée
- **enabled** Si oui ("yes") l'élément est accessible, si non ("no") l'élément a été désactivé

Table : annotations

Cette table contient les *Annotations*, ce qui est distinct des *Metadata*.

- **id** Un Identifiant unique
- **entity_guid** L'entité à laquelle cette métadonnée est attachée
- **name** Le nom de la chaîne
- **value** La valeur de la chaîne
- **value_type** La classe de la valeur, soit texte (text) ou un entier (integer)
- **owner_guid** Le GUID du propriétaire qui a défini cette annotation
- **access_id** Un Contrôle d'Accès sur cette annotation
- **time_created** Le timestamp Unix de la date de création de l'annotation.
- **enabled** Si oui ("yes") l'élément est accessible, si non ("no") l'élément a été désactivé

Table : relationships (relations)

Cette table définit les *Relationships*, qui relient une entité avec une autre.

- **guid_one** Le GUID de l'entité sujet.
- **relationship** Le type de relation.
- **guid_two** Le GUID de l'entité cible.

Tables secondaires**Table : access_collections**

Cette table définit les Collections d'Accès, qui donnent accès aux utilisateurs aux *Entités* ou aux *Annotations*.

- **id** Un Identifiant unique
- ***name** Le nom de la collection d'accès
- **owner_guid** Le GUID de l'entité propriétaire (par ex. un utilisateur ou un groupe)
- **subtype** le sous-type de la collection d'accès (par ex. *friends* ou *group_acl*)

3.5.3 Événements et Hooks des plugins**Contenu**

- *Aperçu*
 - *Événements Elgg (Events) vs. Hooks des Plugins*
- *Événements Elgg (Events)*
 - *Événements Avant (Before) et Après (After)*
 - *Gestionnaires d'événement Elgg (Event Handlers)*
 - *Enregistrez un gestionnaire d'événement (Elgg Event)*
 - *Déclencher un événement (Elgg Event)*
 - *Déclenche une séquence Elgg Event*
- *Hooks des plugins*
 - *Gestionnaires des Hooks plugin*
 - *Enregistrez un gestionnaire de Hook plugin*
 - *Déclencher un Hook plugin*
 - *Dé-enregistrer des gestionnaires d'événement ou de hook*
 - *Ordre d'appel des gestionnaires*

Aperçu

Elgg a un événement système qui peut être utilisé pour remplacer ou étendre les fonctionnalités du noyau.

Les plugins influencent le système en créant des gestionnaires ou handlers ([appelables](#) tels que des fonctions et des méthodes) et en les enregistrant pour gérer deux types d'événements : *Événements Elgg (Events)* et *Hooks des plugins*.

Quand un événement est déclenché, un jeu des gestionnaires (handlers) est exécuté par ordre de priorité. Les arguments sont passés à chaque gestionnaire qui peut influencer le processus. Après exécution, la fonction déclencheuse (« trigger ») retourne une valeur qui dépend du comportement des gestionnaires.

Voir aussi :

- *Liste des événements dans le cœur*
- *Liste des hooks de plugin du noyau*

Événements Elgg (Events) vs. Hooks des Plugins

Les principales différences entre *Événements Elgg (Events)* et *Hooks des plugins* sont :

1. La majorité des événements Elgg peuvent être annulé ; à moins que l'événement soit un événement « after », un gestionnaire qui retourne *false* peut annuler l'événement, et plus aucun autre gestionnaire n'est appelé.
2. Les hooks plugin ne peuvent pas être annulé ; tous les gestionnaires sont toujours appelés.
3. Les hooks lugin passent une valeur arbitraire à travers les gestionnaires, leur donnant à chacun une chance de la modifier en cours de route.

Événements Elgg (Events)

Les événements Elgg Events sont déclenchés quand un objet Elgg est créé, modifié ou supprimé ; et à différentes étapes importantes du chargement du framework Elgg. Exemples : lors de la création d'un article de blog ou de l'identification d'un utilisateur.

Contrairement aux *Hooks des plugins*, *la plupart des événements Elgg peuvent être annulé*, ce qui interrompt l'exécution des gestionnaires, et potentiellement annulent certaines actions dans le noyau d'Elgg.

Chaque événement Elgg a un nom et un type d'objet (system, user, object, nom de relation, annotation, group) qui décrit le type d'objet passé aux gestionnaires.

Événements Avant (Before) et Après (After)

Certains événements sont séparés en « before » (avant) et « after » (après). Ceci évite la confusion autour de l'état fluctuant du système. Par ex. Est-ce que l'utilisateur est identifié au cours de l'événement [login, user] ?

Les événements Avant ont des noms qui se terminent par « :before » et sont exécutés avant qu'il ne se passe quelque chose. Comme pour les événements traditionnels, les gestionnaires peuvent annuler l'événement en retournant *false*.

Les événements Après, avec des noms qui se terminent par « :after », sont exécutés après qu'il se soit passé quelque chose. Au contraire des événements traditionnels, ces gestionnaires *ne peuvent pas* annuler ces événements ; tous les gestionnaires seront toujours appelés.

Là où des événements :before et :after sont disponibles, les développeurs sont encouragés à faire la transition vers eux, même si d'anciens événements resteront supportés pour des raisons de compatibilité descendante.

Gestionnaires d'événement Elgg (Event Handlers)

Les gestionnaires d'événement Elgg sont appelables avec l'un des prototypes suivants :

```
<?php

/**
 * @param \Elgg\Event $event The event object
 *
 * @return bool if false, the handler is requesting to cancel the event
 */
function event_handler1(\Elgg\Event $event) {
    ...
}

/**
 * @param string $event      The name of the event
 * @param string $object_type The type of $object (e.g. "user", "group")
 * @param mixed  $object     The object of the event
 *
 * @return bool if false, the handler is requesting to cancel the event
 */
function event_handler2($event, $object_type, $object) {
    ...
}
```

Dans `event_handler1`, l'objet `Event` a diverses méthodes pour récupérer le nom, le type d'objet, et l'objet de l'événement. Voyez l'interface `Elgg\Event` pour plus de détails.

Dans tous les cas, si un gestionnaire retourne `false`, l'événement est annulé, ce qui empêche l'exécution des autres gestionnaires. Toutes les autres valeurs de retour sont ignorées.

Note : Si le type d'événement est « object » ou « user », utilisez le type de données `\Elgg\ObjectEvent` ou `\Elgg\UserEvent` à la place, ce qui clarifie le type de retour de la méthode `getObject()`.

Enregistrez un gestionnaire d'événement (Elgg Event)

Enregistrez votre gestionnaire pour un événement en utilisant `elgg_register_event_handler` :

```
<?php

elgg_register_event_handler($event, $object_type, $handler, $priority);
```

Paramètres :

- **\$event** Le nom de l'événement.
- **\$object_type** Le type d'objet (par ex. « user » ou « object ») ou « all » pour tous les types pour lesquels l'événement est déclenché.
- **\$handler** Le callback ou la fonction gestionnaire.
- **\$priority** La priorité - 0 en premier et la valeur par défaut est 500.

Objet ne fait pas ici référence à un `ElggObject` mais plutôt à une chaîne de caractères qui décrit n'importe quel objet dans le framework : le système, un utilisateur, un objet, une relation, une annotation, un groupe.

Exemple :

```
<?php

// Register the function myPlugin_handle_create_object() to handle the
// create object event with priority 400.
elgg_register_event_handler('create', 'object', 'myPlugin_handle_create_object', 400);
```

Avertissement : Si vous gérez l'événement « update » d'un objet, évitez d'appeler `save()` dans votre gestionnaire d'événement. Tout d'abord ce n'est probablement pas nécessaire car l'objet est enregistré après que l'événement soit terminé, mais aussi parce que `save()` appelle un autre événement « update » et ne rend plus disponible `$object->getOriginalAttributes()`.

Classes invocables comme gestionnaires

Vous pouvez utiliser une classe avec une méthode `__invoke()` comme gestionnaire. Enregistrez simplement le nom de la classe et elle sera instanciée (sans argument) pour toute la durée de vie de l'événement (ou du hook).

```
<?php

namespace MyPlugin;

class UpdateObjectHandler {
    public function __invoke(\Elgg\ObjectEvent $event) {

    }
}

// in init, system
elgg_register_event_handler('update', 'object', MyPlugin\UpdateObjectHandler::class);
```

Déclencher un événement (Elgg Event)

Vous pouvez déclencher un événement personnalisé en utilisant `elgg_trigger_event` :

```
<?php

if (elgg_trigger_event($event, $object_type, $object)) {
    // Proceed with doing something.
} else {
    // Event was cancelled. Roll back any progress made before the event.
}
```

Pour les événements avec des états ambigus, tels que l'identification d'un utilisateur, vous devriez utiliser *Événements Avant (Before) et Après (After)* en appelant `elgg_trigger_before_event` ou `elgg_trigger_after_event`. Ceci clarifie pour le gestionnaire d'événement l'état auquel attendre et quels événements peuvent être annulés.

```
<?php

// handlers for the user, login:before event know the user isn't logged in yet.
if (!elgg_trigger_before_event('login', 'user', $user)) {
```

(suite sur la page suivante)

(suite de la page précédente)

```

    return false;
}

// handlers for the user, login:after event know the user is logged in.
elgg_trigger_after_event('login', 'user', $user);

```

Paramètres :

- **\$event** Le nom de l'événement.
- **\$object_type** Le type d'objet (par ex. « user » ou « object »).
- **\$object** L'objet (par ex. une instance de `ElggUser` ou `ElggGroup`)

La fonction va retourner `false` si n'importe lequel des gestionnaires sélectionnés retourne `false` et si l'événement est interruptible, sinon elle retournera `true`.

Déclenche une séquence Elgg Event

Au lieu de déclencher manuellement l'événement `:before` et `:after`, il est possible de déclencher une séquence d'événements. Cela déclenchera l'événement `:before`, puis l'événement réel et enfin l'événement `:after`.

```
elgg()->events->triggerSequence($event, $type, $object, $callable);
```

When called with for example `'cache:clear', 'system'` the following three events are triggered

- `'cache:clear:before', 'system'`
- `'cache:clear', 'system'`
- `'cache:clear:after', 'system'`

Paramètres :

- **\$event** Le nom de l'événement.
- **\$object_type** Le type d'objet (par ex. « user » ou « object »).
- **\$object** L'objet (par ex. une instance de `ElggUser` ou `ElggGroup`)
- **\$callable** Appelable pour exécuter sur l'événement réussi, avant event `:after`

Hooks des plugins

Les Hooks Plugin fournissent un moyen pour les plugins de déterminer ou de modifier collaborativement une valeur. Par exemple, pour décider si un utilisateur a la permission de modifier une entité ou d'ajouter des options de configurations supplémentaires à un plugin.

Un hook plugin a une valeur passée à la fonction déclencheuse (« trigger »), et chaque gestionnaire a la possibilité de modifier cette valeur avant qu'elle soit passée au prochain gestionnaire. Après que le dernier gestionnaire a été exécuté, la valeur finale est retournée par le trigger.

Gestionnaires des Hooks plugin

Les gestionnaires de hooks sont appelables avec l'un des prototypes suivants :

```

<?php

/**
 * @param \Elgg\Hook $hook The hook object
 *
 * @return mixed if not null, this will be the new value of the plugin hook

```

(suite sur la page suivante)

```

*/
function plugin_hook_handler1(\Elgg\Hook $hook) {
    ...
}

/**
 * @param string $hook    The name of the plugin hook
 * @param string $type     The type of the plugin hook
 * @param mixed $value    The current value of the plugin hook
 * @param mixed $params   Data passed from the trigger
 *
 * @return mixed if not null, this will be the new value of the plugin hook
 */
function plugin_hook_handler2($hook, $type, $value, $params) {
    ...
}

```

Dans `plugin_hook_handler1`, l'objet `Hook` a diverses méthodes pour récupérer le nom, le type, la valeur et les paramètres du hook. Voyez l'interface `Elgg\Hook` pour plus de détails.

Dans les deux cas, si le gestionnaire ne retourne aucune valeur (ou explicitement la valeur `null`), la valeur du hook plugin n'est pas modifiée. Dans le cas contraire, la valeur retournée devient la nouvelle valeur du hook plugin, et il deviendra alors disponible via `$hook->getValue()` (ou `$value`) dans le prochain gestionnaire.

Enregistrez un gestionnaire de Hook plugin

Enregistrez votre gestionnaire pour un hook plugin en utilisant `elgg_register_plugin_hook_handler` :

```

<?php

elgg_register_plugin_hook_handler($hook, $type, $handler, $priority);

```

Paramètres :

- **\$hook** Le nom du hook plugin.
- **\$type** Le type de hook, ou "all" pour tous les types.
- **\$handler** Le callback ou la fonction gestionnaire.
- **\$priority** La priorité - 0 en premier et la valeur par défaut est 500.

Type peut avoir des sens différents. Cela peut signifier un type d'entité Elgg ou quelque chose de spécifique au nom du hook plugin.

Exemple :

```

<?php

// Register the function myPlugin_hourly_job() to be called with priority 400.
elgg_register_plugin_hook_handler('cron', 'hourly', 'myPlugin_hourly_job', 400);

```

Déclencher un Hook plugin

Vous pouvez déclencher un plugin hook personnalisé en utilisant `elgg_trigger_plugin_hook` :

```
<?php

// filter $value through the handlers
$value = elgg_trigger_plugin_hook($hook, $type, $params, $value);
```

Paramètres :

- **\$hook** Le nom du hook plugin.
- **\$type** Le type de hook, ou “all” pour tous les types.
- **\$params** Des données arbitraires passées par le déclencheur aux gestionnaires.
- **\$value** La valeur initiale du hook plugin.

Avertissement : Les arguments *\$params* et *\$value* sont inversés entre les gestionnaires de hook plugin et les fonctions de déclenchement !

Dé-enregistrer des gestionnaires d'événement ou de hook

Les fonctions `elgg_unregister_event_handler` et `elgg_unregister_plugin_hook_handler` peuvent être utilisées pour retirer des gestionnaires déjà enregistrés par un autre plugin ou le noyau d'Elgg. Les paramètres sont dans le même ordre que pour les fonctions d'enregistrement, excepté qu'il n'y a pas de paramètre de priorité.

```
<?php

elgg_unregister_event_handler('login', 'user', 'myPlugin_handle_login');
```

Les fonctions anonymes ou les objets invocables ne peuvent pas être enregistrés, mais des fonctions de rappel (callback) de méthode dynamique peuvent être dé-enregistrés en donnant la version statique de la fonction de rappel :

```
<?php

$obj = new MyPlugin\Handlers();
elgg_register_plugin_hook_handler('foo', 'bar', [$obj, 'handleFoo']);

// ... elsewhere

elgg_unregister_plugin_hook_handler('foo', 'bar', 'MyPlugin\Handlers::handleFoo');
```

Même si le gestionnaire d'événement référence un appel à une méthode dynamique, le code ci-dessus va bien supprimer le gestionnaire.

Ordre d'appel des gestionnaires

Les gestionnaires sont d'abord appelés par ordre de priorité, puis par ordre d'enregistrement.

Note : Avant Elgg 2.0, l'enregistrement avec le mot-clef `all` provoquait un appel tardif des gestionnaires, même s'ils avaient été enregistrés avec des priorités plus faibles.

3.5.4 Internationalisation

Elgg 1.0 et supérieur diffèrent des versions précédentes dans le sens où ces versions utilisent un tableau de valeur texte personnalisé plutôt que `gettext`. Ceci améliore la performance du système et la fiabilité du système de traduction.

TODO : plus svp

3.5.5 AMD

Aperçu

Il existe deux systèmes JavaScript dans Elgg : le système 1.8 devenu obsolète, et le nouveau système compatible [AMD \(Asynchronous Module Definition\)](#) introduit dans la 1.9.

Discussions au sujet des bénéfices de l'utilisation d'AMD dans Elgg.

Pourquoi AMD ?

Nous avons beaucoup travaillé pour rendre le JavaScript d'Elgg plus maintenable et plus utile. Nous avons fait quelques grands pas dans la 1.8 en introduisant l'objet et la bibliothèque JavaScript « `elgg` », mais avons rapidement réalisé que l'approche que nous prenions n'était pas susceptible de passer à l'échelle.

La taille du [JS sur le web](#) est [rapidement croissante](#), et le JS dans Elgg s'accroît également. Nous voulons qu'Elgg soit capable d'offrir une solution qui rende le développement JS aussi productif et maintenable que possible pour aller plus loin.

Les [raisons de choisir AMD](#) sont nombreuses et bien documentées. Mettons en avant seulement les raisons les plus pertinentes dans la mesure où elles sont liées à Elgg spécifiquement.

1. Gestion des dépendances simplifiée

Les modules AMD se chargent de manière asynchrone et s'exécutent dès que leurs dépendances sont disponibles, ce qui élimine le besoin de préciser des « priorités » et des emplacements « location » lorsqu'on enregistre une bibliothèque JS dans Elgg. De plus, vous n'avez pas besoin de vous soucier de charger explicitement les dépendances d'un module dans PHP. Le chargeur AMD (RequireJS en l'occurrence) prend soin de tous ces tracas pour vous. Il est également possible d'avoir des [dépendances texte](#) avec le plugin `text` de RequireJS, de sorte que le templating côté client devrait être un plaisir.

2. AMD fonctionne dans tous les navigateurs. Dès maintenant.

Les développeurs Elgg ont déjà écrit beaucoup de JavaScript. Nous savons que vous voulez en écrire plus. Nous ne pouvons pas accepter d'attendre 5-10 ans pour qu'une solution native de modules JS soit disponible dans tous les navigateurs avant que nous puissions organiser notre JavaScript d'une manière maintenable.

3. Vous n'avez pas besoin d'une étape de build pour développer avec AMD.

Nous apprécions le cycle édition-mise à jour (« edit-refresh ») du web. Nous voulions faire en sorte que toute personne qui développe avec Elgg puisse continuer à expérimenter ce plaisir. Des formats de modules synchrones comme Closure ou CommonJS n'étaient tout simplement pas une option pour nous. Mais même si AMD n'a pas besoin d'une étape de build, *il reste cependant très adapté pour le build*. A cause du wrapper `define()`, il est possible de concaténer de multiples modules dans un seul fichier et de livrer l'ensemble en une seule fois dans un environnement de production.¹

AMD est un système de chargement de modules largement éprouvé et bien pensé pour le web d'aujourd'hui. Nous avons beaucoup de gratitude pour le travail qui a été mis dedans, et sommes excités de l'offrir comme la solution standard pour le développement JavaScript dans Elgg à partir de Elgg 1.9.

3.5.6 Sécurité

L'approche d'Elgg pour les diverses questions de sécurité est commune à toutes les applications web.

Astuce : Pour signaler une vulnérabilité potentielle dans Elgg, envoyez un email à security@elgg.org.

Contenu

- *Mots de passe*
 - *Validation du mot de passe*
 - *Hachage du mot de passe*
 - *Mitigation du mot de passe*
 - *Réinitialisation du mot de passe*
- *Sessions*
 - *Fixation de session*
 - *Cookie « Se souvenir de moi »*
- *Authentification alternative*
- *HTTPS*
- *XSS*
- *CSRF / XSRF*
- *URLs signées*
- *Injection SQL*
- *Vie privée*
- *Sécurisation*

1. Ceci n'est pas encore supporté par le noyau Elgg, mais nous sommes en train de nous pencher dessus dans la mesure où la réduction des allers-retours est critique pour une bonne première expérience, particulièrement sur les appareils mobiles.

Mots de passe

Validation du mot de passe

La seule restriction qu'Elgg impose sur le mot de passe est qu'il doit comporter au moins 6 caractères par défaut, quoique ceci puisse être changé dans `/elgg-config/settings.php`. Des critères additionnels peuvent être ajoutés en enregistrant un plugin hook pour `registeruser:validate:password`.

Hachage du mot de passe

Les mots de passe ne sont jamais stockés en clair, seulement sous forme de valeur de hachage avec sel produit par `bcrypt`. Ceci est effectué via la fonction standard `password_hash()`. Sur des systèmes plus anciens, le `polyfill password-compat` est utilisé, mais l'algorithme est identique.

Les installations Elgg créées avant la version peuvent avoir des chaînes de hachage de mot de passe « historiques » résiduelles créées en utilisant MD5 avec sel. Elles ont été migrées vers `bcrypt` lorsque les utilisateurs se connectent, et seront totalement retirées lorsque le système est mis à niveau vers Elgg 3.0. Dans le même temps, nous sommes heureux d'assister les propriétaires de sites à retirer manuellement ces chaînes de hachage historique, même si cela force ces utilisateurs à réinitialiser leur mot de passe.

Mitigation du mot de passe

Elgg a un mécanisme de mitigation des mots de passe qui rend les attaques par dictionnaire depuis l'extérieur très difficiles. Un utilisateur n'est autorisé qu'à 5 tentatives de connexion par période de 5 minutes.

Réinitialisation du mot de passe

Si un utilisateur oublie son mot de passe, la génération d'un nouveau mot de passe aléatoire peut être demandée. Après la demande, un email est envoyé avec une URL unique. Quand l'utilisateur visite cette URL, un nouveau mot de passe est envoyé par email à l'utilisateur.

Sessions

Elgg utilise la gestion des session PHP avec des gestionnaires personnalisés. Les données de sessions sont stockées dans la base de données. Le cookie de session contient l'id de session qui lie l'utilisateur au navigateur. Les métadonnées de l'utilisateur sont conservées dans la session, notamment le GUID, le nom d'utilisateur et l'adresse email.

La durée de vie de la session est contrôlée par la configuration PHP du serveur et additionnellement à travers les options dans le fichier `/elgg-config/settings.php`.

Fixation de session

Elgg protège contre la fixation de session en régénérant l'id de session lorsqu'un utilisateur se déconnecte.

Cookie « Se souvenir de moi »

Afin de permettre aux utilisateurs de rester identifiés pour une période plus longue que le navigateur ait été fermé ou pas, Elgg utilise un cookie (nommé par défaut `elggperm`) qui contient ce qui peut être considéré comme un super identifiant de session. Cet identifiant est conservé dans la table des cookies. Quand une session est initiée, Elgg vérifie la présence du cookie `elggperm`. S'il existe et que le code de session dans le cookie correspond au code dans la table des cookies, l'utilisateur correspondant est automatiquement connecté.

Lorsqu'un utilisateur change son mot de passe tous les cookies permanents sont supprimés de la base de données.

La durée de vie des cookies persistants peut être contrôlée dans le fichier `/elgg-config/settings.php`. La durée de vie par défaut est de 30 jours. Les entrées de la base de données pour les cookies persistants seront supprimées après l'expiration de la durée de vie.

Authentification alternative

Note : Cette section est très lacunaire

Pour remplacer le système d'authentification par défaut des utilisateurs d'Elgg, un plugin pourrait remplacer l'action par défaut `login` par sa propre action. Une meilleure alternative est d'enregistrer un gestionnaire d'authentification (PAM) en utilisant `register_pam_handler()` qui gère l'authentification des utilisateurs sur la base de nouvelles exigences.

HTTPS

Note : Vous devez activer le support de SSL sur votre serveur pour que chacune de ces techniques fonctionne.

Vous pouvez servir l'ensemble de votre site par SSL en changeant l'URL du site pour inclure « `https` » au lieu de seulement « `http` ».

XSS

Le filtrage est utilisé dans Elgg pour rendre les attaques XSS plus difficiles. L'objectif du filtrage est de supprimer les JavaScript et autres saisies dangereuses des utilisateurs.

Le filtrage est assuré à travers la fonction `filter_tags()`. Cette fonction prend une chaîne de caractères et retourne une chaîne filtrée. Elle déclenche le hook plugin `validate, input`.

Par défaut Elgg est fourni avec le code de filtrage `htmlawed`. Les développeurs peuvent ajouter n'importe quel code additionnel ou de remplacement sous forme de plugin.

La fonction `filter_tags()` est appelée pour chaque saisie utilisateur dès lors que la saisie est obtenue à travers un appel à `get_input()`. Si pour quelque raison un développeur souhaite ne pas appliquer le filtrage par défaut sur certaines saisies utilisateur, la fonction `get_input()` a un paramètre pour désactiver le filtrage.

CSRF / XSRF

Elgg génère des jetons de sécurité pour empêcher la contrefaçon de requêtes inter-sites. Ceux-ci sont intégrés dans tous les formulaires et les requêtes AJAX modificatrices d'état dès lors que la bonne API est utilisée. Lisez-en plus dans le guide de développement *Formulaires + Actions*.

URLs signées

Il est possible de protéger les URLs avec une signature unique. Lisez-en plus dans le the *Formulaires + Actions* Guide de développement.

Injection SQL

L'API d'Elgg's assainit toutes les entrées avant d'effectuer des requêtes en base de données. Lisez-en plus dans la documentation de design *Base de données*.

Vie privée

Elgg utilise un système d'ACL pour contrôler quels utilisateurs ont accès à divers éléments de contenu. Lisez-en plus dans la *Base de données* documentation de design.

Sécurisation

Les administrateurs du site peuvent configurer les paramètres qui vont aider à sécuriser le site. Lisez-en plus dans le Guide d'administration *Sécurité*.

3.5.7 Loggable

Loggable iest une interface héritée par toute classe qui veut que les événements liés à ses objets membres soient enregistrés dans le journal système. ElggEntity et ElggExtender héritent tous deux de Loggable.

Loggable définit plusieurs méthodes de classe qui sont utilisées pour l'enregistrement du journal système par défaut, et peuvent être utilisées pour définir vos propres journaux (ainsi que pour d'autres objectifs) :

- `getSystemLogID()` Retourne un identifiant unique pour l'objet à des fins de conservation dans le journal système. C'est généralement le GUID de l'objet
- `getClassName()` Retourne le nom de la classe de l'objet
- `getType()` Retourner le type d'objet
- `getSubtype()` Récupère le sous-type de l'objet
- `getObjectFromID($id)` Pour un ID donné, retourne l'objet qui lui est associé

Détails de la base de données

Le journal système par défaut est enregistré dans la table `system_log` de la *base de données*. Il contient les champs suivants :

- **id** - Un identifiant numérique unique de la ligne
- **object_id** - Le GUID de l'entité sur laquelle l'action est effectuée
- **object_class** - La classe de l'entité sur laquelle l'action est effectuée (par ex. ElggObject)
- **object_type** - Le type d'entité sur laquelle l'action est effectuée (par ex. object)
- **object_subtype** - Le sous-type d'entité sur laquelle l'action est effectuée (par ex. blog)
- **event** - L'événement enregistré (par ex. create ou update)

- **performed_by_guid** - Le GUID de l'entité agissante (l'utilisateur qui réalise l'action)
- **owner_guid** - Le GUID de l'utilisateur à qui appartient l'entité sur laquelle l'action est effectuée
- **access_id** - Le niveau d'accès associé avec cette entrée de journal
- **time_created** - Le timestamp UNIX de la date de l'événement

3.6 Guides du contributeur

Participez à rendre Elgg encore meilleur.

Elgg est un projet communautaire. Il dépend du soutien de volontaires pour réussir. Voici plusieurs manières d'aider :

3.6.1 Écrire du code

Comprenez les standards et process d'Elgg pour que vos propositions de modification soient acceptées aussi vite que possible.

Contenu

- *Agrément de licence*
- *Demandes de fusion (Pull Requests)*
- *Standards de développements*
- *Tester*
- *Meilleures pratiques de développement*
- *APIs obsolètes*

Agrément de licence

En proposant un patch vous accordez de publier le code sous une [licence GPLv2](#) et une [licence MIT](#).

Demandes de fusion (Pull Requests)

Les Pull requests (PRs) sont le meilleur moyen de contribuer au noyau d'Elgg. L'équipe de développement les utilise y compris pour les modifications les plus mineures.

Pour de nouvelles fonctionnalités, [soumettez une demande de fonctionnalité](#) ou [parlez-en avec nous](#) en premier lieu et assurez-vous que l'équipe du noyau approuve votre proposition avant de passer beaucoup de temps sur du code.

Listes de vérification (checklists)

Utilisez ces listes de vérification réduites pour les nouveaux PRs sur github afin de garantir des contributions de haute qualité et d'aider tout le monde à comprendre le statut des PRs ouverts.

Demandes de correction de bug (bugfix PR) :

- [] Commit messages are in the standard format
- [] Includes regression test
- [] Includes documentation update (if applicable)
- [] Is submitted against the correct branch
- [] Has LGTM from at least one core developer

Demandes de fonctionnalité (feature PR) :

- [] Commit messages are in the standard format
- [] Includes tests
- [] Includes documentation
- [] Is submitted against the correct branch
- [] Has LGTM from at least two core developers

Choisir une branche vers laquelle publier

Le tableau suivant suppose que la dernière version stable est la 2.1.

Type de changement	Branche vers laquelle publier
Correctif de sécurité	Ne le faites pas ! Envoyez un email à security@elgg.org pour des conseils.
Correctif de bug	1.12 (ou 2.1 si le correctif pour 1.12 est trop complexe)
Performance	2.x
Dépréciation	2.x
Fonctionnalité mineure	2.x
Fonctionnalité majeure	master (maître)
A n'importe quel changement non rétro-compatible	master (maître)

Si vous ne savez pas quelle branche choisir, demandez !

La différence entre des changements mineurs et majeurs est subjective et soumise à l'appréciation de l'équipe du noyau.

Format du message de commit

Nous exigeons un format particulier afin de permettre de publier plus souvent, avec des journaux des modifications et un historique des modifications améliorés. Suivez simplement les étapes suivantes :

- Commencez par le **type** en sélectionnant la *dernière catégorie qui s'applique* dans cette liste :
 - **docs** - *uniquement* la documentation a été mise à jour
 - **chore** - (corvée) ceci comprend les remaniements (refactoring), les changements de style de code, l'ajout de tests manquants, ce qui concerne l'intégration continue, etc.
 - **perf** - l'objectif principal est d'améliorer la performance
 - **fix** - ceci corrige un bug
 - **deprecate** - la modification rend obsolète toute partie de l'API
 - **feature** - ceci ajoute une nouvelle fonctionnalité pour les utilisateurs ou les développeurs
 - **security** - la modification affecte une question de sécurité d'une manière ou d'une autre. *Merci de ne pas pousser ce commit vers un repository public.* Contactez plutôt security@elgg.org.
Par ex., si votre commit fait des remaniements pour régler (fix) un bug, cela reste un « fix ». Cependant, si ce bug est lié à la sécurité, le type doit être « security » et vous devriez envoyer un email à security@elgg.org avant de procéder. En cas de doute, faites au mieux, et un relecteur (reviewer) vous fournira des conseils.
- Entre parenthèses, ajoutez le **component**, une courte chaîne qui décrit les sous-syst-me en train d'être modifié. Quelques exemples : `views`, `i18n`, `seo`, `all`, `cache`, `db`, `session`, `router`, `<plugin_name>`.
- Ajoutez une virgule, un espace, et un court résumé **summary** des modifications, qui va apparaître dans le journal des modifications (changelog).
Aucune ligne ne devrait dépasser 100 caractères en longueur, aussi gardez un résumé concis.

Bon résumé	Mauvais résumé (problème)
les propriétaires de pages (pages owners) voient leur propre bloc propriétaire sur les pages	correction de bug (vague)
la vue du graphique en barres n'échoue plus si "foo" n'est pas défini	met à jour views/default/bar.php de sorte que la vue bar ne soit plus... (info redondante)
réduit la mise en page de la rivière pour rentrer sur un iPhone	modifie la mise en page de la rivière (vague)
elgg_foo() gère les tableaux pour \$bar	dans elgg_foo() vous pouvez maintenant passer un tableau pour \$bar et la fonction sera... (déplacez les précisions vers la description)
supprime la couleur du lien de l'entête des commentaires dans la rivière	corrige la bdd pour que... (info redondante)
requiert un titre non-vide pour enregistrer les pages	peut enregistrer des pages sans titre (résumé de manière confuse l'ancien comportement)

4. (recommandé) Sauter une ligne et ajoutez une **description** des modification. Incluez leur raison d'être, toute information sur la compatibilité ascendante ou descendante, et toute raison pour laquelle cette modification devait être faite d'une certaine manière. Exemple :

Nous accélérons la migration des tables en utilisant une seule requête INSERT INTO ... SELECT au lieu de ligne par ligne. Cette migration se produit durant la mise à niveau vers Elgg 1.9.

A moins que votre modification soit triviale/évidente, une description est requise.

5. Si le commit correspond à une demande (« issue ») Github, sautez une ligne et ajoutez **Fixes #** suivi par le numéro de demande. Par ex. **Fixes #1234**. Vous pouvez inclure de multiples demande en les séparant par des virgules.

GitHub va fermer automatiquement la demande quand le commit est fusionné. Si vous souhaitez simplement faire référence à une demande, utilisez **Refs #**.

Quand c'est terminé, votre message de commit aura le format :

```
type(component): summary

Optional body
Details about the solution.
Opportunity to call out as breaking change.

Closes/Fixes/Refs #123, #456, #789
```

Voici un exemple d'un bon message de commit :

```
perf(upgrade): speeds up migrating remember me codes

We speed up the Remember Me table migration by using a single INSERT INTO ... SELECT
↳ query instead of row-by-row.
This migration takes place during the upgrade to 1.9.

Fixes #6204
```

Pour valider les messages de commit localement, vérifiez que `.scripts/validate_commit_msg.php` est exécutable, et faites une copie ou un lien symbolique dans le dossier `.git/hooks/commit-msg`.

```
chmod u+x .scripts/validate_commit_msg.php
ln -s .scripts/validate_commit_msg.php .git/hooks/commit-msg/validate_commit_msg.php
```

Réécrire des messages de commits

Si votre PR ne se conforme pas au format standard de message de commit, nous vous demanderons de le réécrire.

Pour modifier seulement le dernier commit :

1. Amender le commit : `git commit --amend` (git ouvre le message dans un éditeur de texte).
2. Changer le message et enregistrer/quitter l'éditeur.
3. Forcez l'envoi de votre branche : `git push -f your_remote your_branch` (votre PR sera mis à jour).
4. Renommez le titre du PR pour correspondre

Sinon vous pourrez avoir besoin d'effectuer un « rebase » interactif :

1. Faites un rebase des derniers N commits : `git rebase -i HEAD~N` où N est le nombre. (Git va ouvrir le fichier `git-rebase-todo` pour modification)
2. Pour les commits qui ont besoin d'être modifiés, modifiez pick en r (pour « reword », reformulation) et enregistrez/quitez l'éditeur.
3. Modifiez le(s) message(s) de commit, enregistrez/quitter l'éditeur (git va présenter un fichier pour chaque commit qui a besoin d'être reformulé).
4. `git push -f your_remote your_branch` pour forcer un push de la branche (qui met à jour votre PR).
5. Renommez le titre du PR pour correspondre

Standards de développements

Elgg utilise des jeux de standards qui sont basés en partie sur les standards PEAR et PSR2. Vous pouvez voir le jeu de règles dans `vendor/elgg/sniffs/elgg.xml`.

Pour vérifier les violations des standards de votre code (à condition que vous ayez installé Elgg avec les dépendances dev), exécutez :

```
phpcs --standard=vendor/elgg/sniffs/elgg.xml -s path/to/dir/to/check
```

Pour corriger automatiquement des violations réparables, exécutez :

```
phpcbf --standard=vendor/elgg/sniffs/elgg.xml path/to/dir/to/fix
```

Pour vérifier les répertoires du noyau, vous pouvez utiliser le raccourci `composer lint` et `composer lint-fix`.

Tester

Elgg dispose de tests automatisés à la fois pour PHP et JavaScript. Toutes les nouvelles contributions doivent intégrer les tests appropriés.

Voir aussi :

Écrire des tests

Recommandations générales

Découpez les tests par comportements que vous souhaitez tester et utilisez des noms qui décrivent le comportement. Par ex. :

- Pas si bon : Une seule grosse méthode `testAdd()`.
- Mieux : Méthodes `testAddingZeroChangesNothing` et `testAddingNegativeNumberSubtracts`

Efforcez-vous d'utiliser des *designs à base de composants* qui permettent de tester en isolement, sans de grands graphes de dépendances ou d'accès à la base de données. L'injection de dépendances est ici critique.

Tests PHP

PHPUnit

Située dans `engine/tests/phpunit`, c'est notre suite de tests préférée. Elle n'utilise pas d'accès à la base de données, et n'a qu'un accès superficiel à l'API des entités.

- Si possible, nous vous encourageons à créer des composants qui sont testables dans cette suite.
- Envisagez de séparer le stockage de votre composant de sorte que la logique métier puisse être testée ici.
- Dépendez des classes `Elgg\FileSystem*` plutôt que d'utiliser les fonctions du système de fichiers PHP.

Tester les interactions entre services

Dans l'idéal, vos tests devraient construire vos propres graphes d'objet isolés pour des manipulations directes, mais ce n'est pas toujours possible.

Si votre test dépend du Fournisseur de Service Elgg (`_elgg_services()` retourne un `Elgg\Di\ServiceProvider`), réalisez que cela maintient une instance singleton pour la plupart des services qu'il fournit, et que beaucoup de services conservent également leurs propres références locales pour ces services.

Du fait de ces références locales, replacer les services sur le SP (SP = « Service Provider », Fournisseur de Service) au sein d'un test n'aura souvent pas l'effet désiré. Au lieu de cela, vous pourriez avoir besoin d'utiliser la fonctionnalité disponible dans les services eux-mêmes :

- Les services `events` et `hooks` ont des méthodes `backup()` et `restore()`.
- Le service `logger` a des méthodes `disable()` et `enable()`.

Tests Jasmine

Les fichiers de test doivent être nommés `*Test.js` et devraient être placés soit dans `js/tests/` ou à côté de leurs fichiers sources dans `views/default/**.js`. Karma va automatiquement sélectionner les nouveaux fichiers `*Test.js` et exécuter ces tests.

Test générique

```
define(function(require) {
    var elgg = require('elgg');

    describe("This new test", function() {
        it("fails automatically", function() {
            expect(true).toBe(false);
        });
    });
});
```

(suite sur la page suivante)

(suite de la page précédente)

```
});  
});
```

Effectuer les tests

Elgg utilise [Karma](#) avec [Jasmine](#) pour effectuer des tests unitaires JS.

Vous devez avoir nodejs et yarn installés.

Tout d'abord installez toutes les dépendances de développement :

```
yarn
```

Exécutez les tests une seule fois puis quittez :

```
yarn test
```

Vous pouvez également exécuter les tests en continu pendant le développement de sorte qu'ils s'exécutent à chaque enregistrement :

```
karma start js/tests/karma.conf.js
```

Déboguer les tests JS

Vous pouvez exécuter la suite de test au sein des outils de développement de Chrome :

```
yarn run chrome
```

Ceci va renvoyer une URL telle que `http://localhost:9876/`.

1. Ouvrez l'URL dans Chrome, et cliquez sur « Debug ».
2. Ouvrez les outils de développement de Chrome et l'onglet Console.
3. Rechargez la page.

Si vous modifiez un test, vous devrez quitter Karma avec `Ctrl-c` et le redémarrer.

Meilleures pratiques de développement

Rendez votre code plus facile à lire, plus facile à maintenir, et plus facile à déboguer. Un usage consistant de ces recommandations signifie moins de travail de devinettes pour les développeurs, ce qui signifie des développeurs plus heureux, et plus productifs.

Développement en général

Ne Vous Répétez Pas

Si vous copiez-collez des morceaux significatifs de code, considérez qu'il y a une opportunité de réduire la duplication en introduisant une fonction, un argument supplémentaire, une vue, ou une nouvelle classe de composant.

Par ex. Si vous trouvez des vues qui sont identiques à l'exception d'une seule valeur, remaniez-les en une seule vue qui accepte une option.

Note : Dans la publication d'une correction de bugs, *un peu de duplication est préférable à de la refactorisation*. Corrigez les bugs de la manière la plus simple possible, et refactorisez pour réduire la duplication dans la branche de la prochaine version mineure.

Adoptez SOLID et GRASP

Utilisez ces principes pour le design OO pour résoudre des problèmes en utilisant des composants couplés librement, et essayez de rendre tous les composants et le code d'intégration testables.

Les espacements sont gratuits

N'ayez pas peur d'utiliser des blocs de code séparés. Utilisez un espace unique pour séparer les paramètres des fonctions et les concaténations de chaînes.

Nom des variables

Utilisez des noms de variables auto-documentés. `$group_guids` est mieux que `$array`.

Évitez les doubles négations. Préférez `$enable = true` à `$disable = false`.

Nom des interfaces

Utilisez le motif `Elgg\{Namespace}\{Name}`.

N'ajoutez pas de préfixe `I` ou de suffixe `Interface`.

Nous n'utilisons aucun préfixe ou suffixe de sorte que nous sommes encouragés à :

- nommer les classes d'implémentation de manière plus descriptive (le nom « par défaut » est déjà pris).
- faites du typage sur les interfaces, parce que c'est la manière la plus courte et la plus simple de faire.

Nommez les implémentations comme `Elgg\{Namespace}\{Interface}\{Implementation}`.

Fonctions

Autant que possible, ayez des fonctions/méthodes qui renvoient un type unique. Utilisez des valeurs vides telle que `array()`, `""`, ou `0` pour indiquer l'absence de résultat.

Faites attention aux cas où des valeurs de retour valides (telles que `""`) pourraient être interprétées comme vides.

Les fonctions qui ne déclenchent pas une exception lors d'une erreur devraient retourner `false` lorsqu'elles échouent.

Note : Les fonctions/méthodes particulièrement bas-niveau et sans API (par ex. `entity_row_to_elggstar`), qui ne devraient pas échouer dans des conditions normales, devraient renvoyer une exception au lieu de retourner `false`.

Les fonctions qui ne renvoient qu'un booléen devraient être préfixées par `is_` ou `has_` (par ex., `elgg_is_logged_in()`, `elgg_has_access_to_entity()`).

Syntaxe ternaire

Acceptable seulement pour des déclarations sur une seule ligne, non imbriquées.

Minimisez la complexité

Minimisez les blocs imbriqués et les chemins d'exécution distinct au sein du code. Faites un [Return rapide](#) pour réduire les niveaux imbriqués et la charge cognitive lors de la lecture du code.

Utilisez les commentaires à bon escient

Les bons commentaires décrivent le « pourquoi. » Le bon code décrit le « comment. » Par ex. :

Mauvais :

```
// increment $i only when the entity is marked as active.
foreach ($entities as $entity) {
    if ($entity->active) {
        $i++;
    }
}
```

Bon :

```
// find the next index for inserting a new active entity.
foreach ($entities as $entity) {
    if ($entity->active) {
        $i++;
    }
}
```

Ajoutez toujours un commentaire s'il n'est pas évident que quelque chose doit être fait d'une certaine manière. D'autres développeurs qui regardent le code devraient être découragés de réorganiser le code d'une manière qui le casserait.

```
// Can't use empty()/boolean: "0" is a valid value
if ($str === '') {
    register_error(elgg_echo('foo:string_cannot_be_empty'));
    forward(REFERER);
}
```


Commitez de manière efficace

- Péchez par excès de **commits atomiques** qui sont précisément ciblés sur la modification d'un seul aspect du système.
- Eviter de mélanger des modifications qui ne sont pas liées ou des modifications importants des espaces. Les commits avec de nombreux changements sont effrayants et rendent les demandes de fusion (Pull Requests) difficiles à évaluer.
- Utilisez des outils git visuels pour façonner **des diffs extrêmement précis et lisibles**.

Incluez des tests

A chaque fois que c'est possible, *incluez des tests unitaires* pour le code que vous ajoutez ou modifiez.

Gardez les corrections de bugs simples

Évitez la tentation de refactoriser le code pour la publication d'une correction de bug. Faire cela a tendance à introduire des régressions, à casser la fonctionnalité dans ce qui devrait être une version stable.

Recommandations PHP

Voici les standards de développement requis pour le noyau d'Elgg et tous les plugins associés. Les développeurs de plugins sont fortement encouragés à adopter ces standards.

Les développeurs devraient lire en premier le [Guide des Standards de Développement PSR-2](#).

Les standards d'Elgg étendent PSR-2, mais diffèrent des manières suivantes :

- Indentez en utilisant le caractère tabulation, pas des espaces.
- Les parenthèses ouvrantes pour les classes, méthodes, et fonctions doivent être sur la même ligne.
- Si une ligne dépasse 100 caractères, envisagez de la remanier (par ex. en introduisant des variables).
- La compatibilité avec [PSR-1](#) est encouragée, mais pas strictement requise.

Documentation

- Incluez des commentaires PHPDoc sur les fonctions et les classes (toutes les méthodes ; les propriétés déclarées quand c'est approprié), y compris les type et description de tous les paramètres.
- Dans les listes de déclarations `@param`, le début des noms de variables et des descriptions doivent être alignées.
- Annoter les classes, les méthodes, les propriétés et les fonctions avec `@internal` à moins qu'elles ne soient destinées au public, qu'elles ne soient déjà d'une visibilité limitée ou qu'elles ne soient dans une classe déjà marquée comme `@internal`.
- Utilisez `//` ou `/* */` pour les commentaires.
- Utilisez seulement les commentaires `//` à l'intérieur du corps des fonctions et méthodes.

Nommage

- Utilisez des traits de soulignements (« underscores ») pour séparer les mots dans les noms de fonctions, de variables, et les propriétés. Les noms de méthodes utilisent la syntaxe camelCase.
- Les noms de fonctions pour un usage public doivent commencer par `elgg_`.
- Tous les autres noms de fonctions doivent commencer par `_elgg_`.
- Nommez les globales et les constantes en MAJUSCULE (`ACCESS_PUBLIC`, `$CONFIG`).

Divers

Pour les pré-requis PHP, voyez `composer.json`.

N'utilisez pas les balises PHP raccourcies `<?` ou `<%`. Il est possible d'utiliser `<?=` dans la mesure où cela est toujours activé à partir de PHP 5.4.

Quand vous créez des chaînes de caractères avec des variables :

- utilisez des chaînes de caractères avec des guillemets doubles
- encadrez les variables avec des accolades seulement quand c'est nécessaire.

Mauvais (difficile à lire, mauvais usage des guillemets et des `{}`) :

```
echo 'Hello, '.$name.'"! How is your {$time_of_day}?"';
```

Bon :

```
echo "Hello, $name! How is your $time_of_day?";
```

Retirez l'espace blanc à la fin des lignes.

Validation de la valeur

Quand vous travaillez avec des saisies utilisateur, préparez les saisies en dehors de la méthode de validation.

Mauvais :

```
function validate_email($email) {
    $email = trim($email);

    // validate
}

$email = get_input($email);

if (validate_email($email)) {
    // the validated email value is now out of sync with an actual input
}
```

Bon :

```
function validate_email($email) {
    // validate
}

$email = get_input($email);
```

(suite sur la page suivante)

(suite de la page précédente)

```
$email = trim($email);

if (validate_email($email)) {
    // green light
}
```

Utilisez les exceptions

N'ayez pas peur d'utiliser les exceptions. Elles sont plus faciles à gérer que des sorties de fonction mixtes :

Mauvais :

```
/**
 * @return string|bool
 */
function validate_email($email) {
    if (empty($email)) {
        return 'Email is empty';
    }

    // validate

    return true;
}
```

Bon :

```
/**
 * @return void
 * @throw InvalidArgumentException
 */
function validate_email($email) {
    if (empty($email)) {
        throw new InvalidArgumentException('Email is empty');
    }

    // validate and throw if invalid
}
```

Documenter les valeurs de retour

N'utilisez pas `@return void` sur des méthodes qui renvoient une valeur ou null.

Mauvais :

```
/**
 * @return bool|void
 */
function validate_email($email) {
    if (empty($email)) {
        return;
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }

    // validate

    return true;
}
```

Bon :

```
/**
 * @return bool|null
 */
function validate_email($email) {
    if (empty($email)) {
        return null;
    }

    // validate

    return true;
}
```

Recommandations CSS

Enregistrez le css dans les fichiers avec une extension `.css`.

Utilisez des abréviations quand c'est possible

Mauvais :

```
background-color: #333333;
background-image: url(...);
background-repeat: repeat-x;
background-position: left 10px;
padding: 2px 9px 2px 9px;
```

Bon :

```
background: #333 url(...) repeat-x left 10px;
padding: 2px 9px;
```

Utilisez les traits d'union « - », pas les underscores « _ »

Mauvais :

```
.example_class {}
```

Bon :

```
.example-class {}
```

Note : Vous devriez préfixer vos identifiants et noms de classe avec un texte qui identifie votre plugin.

Une propriété par ligne

Mauvais :

```
color: white;font-size: smaller;
```

Bon :

```
color: white;
font-size: smaller;
```

Déclarations des propriétés

Celles-ci devraient être espacées comme ceci : `propriété: valeur;`

Mauvais :

```
color:value;
color :value;
color : value;
```

Bon :

```
color: value;
```

Préfixes fournisseurs (vendor)

- Regroupez les préfixes fournisseurs pour la même propriété
- La version la plus longue des préfixes fournisseurs en premier
- Incluez toujours la version sans préfixe fournisseur
- Ajoutez une ligne supplémentaire entre les groupes avec préfixes fournisseurs et les autres propriétés

Mauvais :

```
-moz-border-radius: 5px;
border: 1px solid #999999;
-webkit-border-radius: 5px;
width: auto;
```

Bon :

```
border: 1px solid #999999;

-webkit-border-radius: 5px;
-moz-border-radius: 5px;
border-radius: 5px;

width: auto;
```

Regroupez les sous-propriétés

Mauvais :

```
background-color: white;
color: #0054A7;
background-position: 2px -257px;
```

Bon :

```
background-color: white;
background-position: 2px -257px;
color: #0054A7;
```

Recommandations JavaScript

Les mêmes standards de formatage que PHP s'appliquent.

Toutes les fonctions devraient être dans l'espace de nom `elgg`.

Les expressions d'une fonction devraient se terminer par un point-virgule.

```
elgg.ui.toggles = function(event) {
    event.preventDefault();
    $(target).slideToggle('medium');
};
```

APIs obsolètes

De temps en temps, des fonctions et des classes doivent être dépréciées au profit de nouveaux remplacements. Dans la mesure où les auteurs de plugins tierce-partie s'appuient sur une API cohérente, la compatibilité ascendante doit être maintenue, mais ne sera pas maintenue indéfiniment dans la mesure où les auteurs de plugins sont supposés mettre à jour leurs plugins. Afin de maintenir la compatibilité ascendante, les API obsolètes vont suivre les recommandations suivantes :

- Les versions mineures (1.x) qui déprécient une API doivent inclure une fonction/classe d'emballage (« wrapper ») -ou tout autre moyen approprié- qui maintient la compatibilité ascendante, y compris tout bug de la fonction/classe originelle. Cette couche de compatibilité utilise `elgg_deprecated_notice(..., '1.11')` pour journaliser le fait que cette fonction est obsolète.
- La révision majeure suivante (2.0) supprime la couche de compatibilité. Toute utilisation de l'API obsolète devrait être corrigé auparavant.

3.6.2 Base de données

Contribuer aux modifications du schéma de la base de données

Contenu

— *Migrations de la base de données*

Migrations de la base de données

Elgg utilise [Phinx](#) pour gérer les migrations de la base de données.

Créer une migration

Pour créer une nouvelle migration, exécutez ce qui suit dans votre console :

```
vendor/bin/phinx create -c engine/conf/migrations.php MigrationClassName
```

Ceci va générer un squelette de migration horodaté dans `engine/schema/migrations/`. Suivez la documentation de Phinx pour appeler les méthodes nécessaires pour modifier les tables de la base de données.

Exécuter une migration

Les migrations sont exécutées à chaque fois que vous exécutez `upgrade.php`. Si vous préférez exécuter les migrations manuellement, vous pouvez le faire via la ligne de commande :

```
// When Elgg is the root project
vendor/bin/phinx migrate -c engine/conf/migrations.php

// When Elgg is installed as a Composer dependency
vendor/bin/phinx migrate -c vendor/elgg/elgg/engine/conf/migrations.php
```

Vérifiez la documentation Phinx pour des drapeaux additionnels qui vous autorisent à effectuer une seule migration ou un jeu de migrations dans un intervalle de temps.

3.6.3 Contribuer à la Documentation

La nouvelle documentation devrait s'intégrer correctement avec le reste de la documentation d'Elgg.

Contenu

- *Tester les documentations localement*
- *Suivez l'organisation du document existant*
- *Utilisez « Elgg » d'une manière grammaticalement correcte*
- *Évitez les pronoms à la première personne*
- *Supprimer le délayage*
- *Préférez les dates absolues aux dates relatives*
- *Ne rappelez pas au lecteur de contribuer*

Tester les documentations localement

Elgg dispose d'un script `grunt` qui construit automatiquement les docs, les ouvre dans une fenêtre de navigateur, et les recharge automatiquement pendant que vous faites des modifications (le rechargement ne prend que quelques secondes). Vous aurez besoin d'avoir `yarn` et `sphinx` installés pour pouvoir utiliser ces scripts.

```
cd path/to/elgg/  
yarn  
grunt
```

C'est aussi simple que cela ! Grunt va continuer à s'exécuter, à vérifier les docs pour des modifications, et à reconstruire automatiquement.

Note : Vous devrez peut-être installer "sphinxcontrib-phpdomain". Vous pouvez le faire avec la commande suivante :
``pip install -U sphinxcontrib-phpdomain``

Suivez l'organisation du document existant

L'organisation actuelle n'est pas nécessairement La Bonne Manière d'organiser les documentations, mais la cohérence est mieux que l'aléatoire.

intro/*

C'est tout ce que les tout nouveaux utilisateurs ont besoin de savoir (installation, fonctionnalités, licence, etc.)

admin/*

Guides pour les administrateurs. Orienté tâches.

guides/*

Guides de l'API pour les développeurs de plugins. Style recette de cuisine. Exemple solide. Élément de code solide. Cassé par les services (actions, i18n, routage, bd, etc.). Ceci devrait discuter seulement de l'API publique et de son comportement, pas des détails d'implémentation ou du raisonnement.

design/*

Documentation de conception (design docs) pour les personnes qui veulent avoir une meilleure compréhension de comment/pourquoi le noyau est construit de cette manière. Ceci devrait discuter des détails d'implémentaiton internes des divers services, de quels compromis ont été faits, et du raisonnement derrière la décision finale. Devrait être utile aux personnes qui veulent contribuer ou pour la communication entre les développeurs du noyau.

contribute/*

Guides de contributeurs pour les diverses manières dont des personnes peuvent participer au sein du projet.

appendix/*

Des informations plus détaillées/méta/de contexte à propos du projet (historique, feuille de route, etc.)

Utilisez « Elgg » d'une manière grammaticalement correcte

Elgg n'est pas un acronyme, l'écrire en majuscules (ELGG ou E-LGG) est incorrect. Veuillez ne pas le faire.

En Français, Elgg ne prend pas d'article quand il est utilisé comme un nom. Voici quelques exemples à imiter :

- “J'utilise Elgg pour mon site web”
- “Installez Elgg pour mettre votre communauté en ligne”

Quand il est utilisé comme adjectif, l'article s'applique au nom principal, aussi vous devriez en utiliser un. Par exemple :

- « Allez sur le site de la communauté Elgg pour trouver de l'aide. »
- « J'ai construis un réseau avec Elgg hier »

Ce conseil peut ne pas être valable pour d'autres langues que l'anglais.

Évitez les pronoms à la première personne

Faites référence au lecteur comme « vous ». Ne vous incluez pas dans la narration habituelle.

Avant :

Quand nous aurons terminé l'installation d'Elgg, nous irons rechercher quelques plugins !

Après :

Quand vous aurez terminé l'installation d'Elgg, recherchez quelques plugins !

Pour faire référence à vous-même (à éviter si possible), utilisez votre nom et écrivez à la troisième personne. Ceci permet aux futurs lecteurs/éditeurs de savoir de qui les opinions sont exprimées.

Avant :

Je pense que le meilleur moyen pour faire X est d'utiliser Y.

Après :

Evan pense que le meilleur moyen pour faire X est d'utiliser Y.

Supprimer le délayage

Avant :

Si vous souhaitez utiliser une bibliothèque javascript tierce-partie au sein du framework Elgg, vous devriez prendre soin d'appeler la fonction `elgg_register_js` pour l'enregistrer.

Après :

Pour utiliser une bibliothèque javascript tierce-partie, appelez `elgg_register_js` pour l'enregistrer.

Préférez les dates absolues aux dates relatives

Il est difficile de dire quand une phrase ou un paragraphe particuliers ont été écrits, aussi les dates relatives deviennent vite incompréhensibles. Les dates absolues donnent également au lecteur une bonne indication de si un projet a été abandonné, ou si un conseil est toujours d'actualité.

Avant :

Récemment le truc a été machin. Bientôt, la chose sera machin aussi.

Après :

Récemment (à partir de septembre 2013), le truc a été machin. Il est prévu que la chose soit également machin d'ici octobre 2013.

Ne rappelez pas au lecteur de contribuer

Concentrez-vous pour ne traiter que du sujet en question. Des sollicitations constantes pour du travail gratuit sont agaçantes et donnent l'impression que le projet est dans le besoin. Si des personnes souhaitent contribuer au projet, elles peuvent visiter le guide du contributeur.

3.6.4 Internationaliser la documentation

Quand vous modifiez la documentation, pensez à mettre à jour les modèles de traduction de la documentation avant de faire un commit :

```
cd docs/  
make gettext
```

Pour plus d'information, voyez <http://www.sphinx-doc.org/en/stable/intl.html#translating-with-sphinx-intl>

Points d'attention spécifiques

Quand vous traduisez la documentation soyez conscients de la syntaxe spécifique dans les fichiers de documentation.

Liens de traduction

- Traduisez le texte dans des liens anonymes (par ex., ``prononciation`__`), mais maintenez l'ordre de tous les liens anonymes dans un unique bloc. S'il y a deux liens anonymes à traduire dans un seul bloc de traduction, il ne doivent pas être inversés l'un par rapport à l'autre.
- Traduisez le texte des liens nommés (par ex., ``site de démo`_`) mais seulement si vous maintenez le noms en utilisant la bonne syntaxe rST. Dans ce cas ce devrait être ``traduction de "site de démo" <demo site_>`_`.

NE traduisez PAS

- Tout ce qui se situe entre des caractères trait vertical ne devrait pas être traduit (par ex., master).
- Le code, à moins que ce ne soit un commentaire dans le code.

3.6.5 Traductions

Les traductions démultiplient l'impact que Elgg peut avoir en le rendant accessible à un plus grand pourcentage du monde.

La communauté sera toujours reconnaissante à ceux qui travaillent dur pour fournir des traductions de haute qualité pour l'UI et la documentation de Elgg.

Transifex

Toutes les traductions pour le projet Elgg sont organisées à travers Transifex.

<https://www.transifex.com/organization/elgg>

Les auteurs de plugins sont encouragés à coordonner les traductions via Transifex également, de sorte que la communauté entière puisse être unie, et que cela facilite la possibilité pour les traducteurs de contribuer à la traduction de n'importe quel plugin dans l'écosystème Elgg.

Récupérer les traductions

Les traductions faites dans Transifex ont besoin d'être récupérées périodiquement dans le dépôt de code d'Elgg. Ceci peut être effectué avec le script `.scripts/languages.php` livré avec le code source de Elgg.

Les pré-requis pour exécuter le script sont :

- Accès à la ligne de commande
- [Git](#)
- [Transifex CLI tool](#)

Le script va effectuer les étapes suivantes :

1. Créer une nouvelle branche git nommée `{branch}_i18n_{timestamp}`
2. Récupérer les traductions pour toutes les langues qui ont 95% des chaînes traduites
3. Retirer les éventuels codes de langue invalides
4. Commiter les modifications vers la branche

Après cela, faites un push de la branche vers Github et faites une nouvelle Pull Request.

Par exemple, si vous voulez récupérer les traductions pour la branche `3.x`, exécutez les commandes suivantes :

```
php .scripts/languages.php 3.x
git push -u your_fork 3.x_i18n_1515151617
```

Exécutez la commande sans paramètre pour avoir des informations plus détaillées sur son utilisation.

Configuration Transifex

La configuration pour Transifex se trouve dans le code source d'Elgg dans le fichier `.tx/config`.

Ce fichier définit :

- Le projet Transifex associé avec la version majeure d'Elgg
- L'emplacement des tous les fichiers qui ont du contenu traduisible

Lisez la [documentation Transifex](#) pour plus d'information.

Nouvelle version majeure d'Elgg

Chaque version majeure d'Elgg doit disposer de son propre projet dans Transifex. De cette manière nous pouvons nous assurer que les chaînes ajoutées et retirées entre les versions ne créent pas de conflit l'une avec l'autre. Par exemple une clef de traduction retirée dans Elgg 3 ne devrait pas être retirée des traductions faites pour Elgg 2. De même, une nouvelle chaîne ajoutée seulement pour Elgg 3 ne devrait pas être incluses dans les traductions prévues pour Elgg 2.

Le processus pour mettre en place une nouvelle version majeure est le suivant :

1. Récupérez les dernières traductions depuis Transifex vers la précédente version majeure
2. Fusionnez la branche git de la version précédente vers la nouvelle afin de s'assurer que toutes les dernières clefs de traduction sont présentes
3. Créez un nouveau projet Transifex sur <https://www.transifex.com/elgg/>
4. Mettez à jour le fichier `.tx/config` dans la branche de développement de la nouvelle version majeure
 - Mettez à jour la configuration pour pointer vers le nouveau projet Transifex
 - Retirez la configuration des plugins retirés
 - Ajoutez la configuration pour les nouveaux plugins
5. Poussez les sources de traduction vers le nouveau projet Transifex avec la commande :

```
tx push -s
```

6. Copiez le nouveau fichier de configuration temporairement (pas de commit) vers la version majeure précédente, et poussez les traductions existantes depuis là vers le nouveau projet :

```
tx push -t -f --no-interactive
```

Plus tard, une fois que la branche dédiée (par ex. `3.x`) a été créée pour la version majeure, configurez Transifex pour récupérer automatiquement les nouvelles clefs à partir d'elle dans <https://www.transifex.com/elgg/elgg-core-3/content/>. De cette manière vous n'avez pas à répéter manuellement l'étape 5 à chaque fois que de nouvelles clefs de traduction sont ajoutées.

Il est important de toujours avoir une branche `n.x` en plus des branches créées pour des versions mineures spécifiques (`n.1`, `n.2`, etc.). De cette manière les URLs des sources automatiquement mises à jour n'ont pas besoin d'être mises à jour à chaque fois qu'une nouvelle branche mineure est créée.

3.6.6 Signaler des problèmes

Signalez bugs et demandes de fonctionnalités sur <https://github.com/Elgg/Elgg/issues>. Voyez ci-dessous pour des recommandations.

LIMITES DE RESPONSABILITÉ

Attention : Les problèmes de sécurité devraient être signalés à [security @ elgg . org](mailto:security@elgg.org) ! Merci de ne publier aucun faille de sécurité sur github !!

Note : Les demandes d'aide relèvent du [site de la communauté](#). Les tickets avec des demandes d'aide seront fermés.

Important : Nous ne pouvons donner aucune garantie sur quand votre ticket sera résolu.

Rapports de bugs

Avant de soumettre un rapport de bug :

- Recherchez un ticket existant à propos du problème que vous rencontrez. Ajouter toute information supplémentaire à cet endroit.
- Vérifiez que le problème est reproductible
 - Sur la dernière version d'Elgg
 - Avec tous les plugins tierce-partie désactivés

Liste de contrôle pour un bon signalement de bug :

- Comportement attendu et comportement constaté
- Des étapes claires pour reproduire le problème
- La version d'Elgg que vous utilisez
- Les navigateurs affectés par ce problème

Demandes de fonctionnalité

Avant de soumettre une demande de fonctionnalité :

- Vérifiez sur le [site de la communauté](#) si un plugin existe avec les fonctionnalités dont vous avez besoin.
- Envisagez si vous le pouvez de *développer un plugin* qui fasse ce dont vous avez besoin.
- Recherchez parmi les tickets fermés pour voir si quelqu'un d'autre a suggéré la même fonctionnalité, mais que cela a été refusé. Vous devrez pouvoir expliquer pourquoi votre suggestion devrait être examinée cette fois-ci.

Liste de contrôle pour une bonne demande de fonctionnalité :

- Explication détaillée de la fonctionnalité
- Cas d'usages réels
- API proposée

3.6.7 Devenir un soutien financier

Tous les fonds recueillis par l'intermédiaire du réseau de supporters Elgg vont directement pour :

- Développement du noyau Elgg
- La fourniture d'infrastructures (elgg.org, [github](https://github.com), etc)

Il s'agit d'un excellent moyen d'aider au développement d'Elgg !

Avantages

Pour seulement \$50 par an pour les individus ou \$150 par an pour les organisations, vous pouvez être listé comme soutien sur [notre page des soutiens](#). Les soutiens d'Elgg apparaissent dans cette liste sauf s'ils demandent à ne pas l'être.

Les soutiens ont l'autorisation d'ajouter ce logo officiel sur leur site s'ils le souhaitent :



Limite de responsabilité

Nous avons une politique de non-remboursement sur les souscriptions des soutiens. Si vous souhaitez arrêter votre soutien, allez sur Paypal et annulez votre souscription. Vous ne serez pas débité l'année suivante.

Etre un soutien d'Elgg ne donne en aucun cas le droit à un individu ou à une organisation le droit de parler au nom du projet Elgg, de commercialiser en son nom, ou de laisser entendre qu'ils sont liés au projet Elgg. Ils peuvent, toutefois, mentionner le fait qu'ils sont soutien du projet Elgg.

Si vous avez des questions à propos de cet avertissement, écrivez à info@elgg.org.

Nous nous réservons le droit de retirer ou refuser une souscription sans avertissement préalable à notre entière discrétion. Il n'y a pas de politique de remboursement.

S'il n'y a pas d'utilisation évidente de Elgg, votre site sera relié avec l'attribut « nofollow ».

S'inscrire

Si vous souhaitez devenir un supporter d'Elgg :

- lisez l'[avertissement](#) ci-dessus
- sur la page des soutiens, [souscrivez via Paypal](#)
- envoyez un email à info@elgg.org avec :
 - la date de votre souscription
 - votre nom (et le nom de l'organisation, s'il y a lieu)
 - votre site web
 - votre profil sur la communauté Elgg

Une fois que tous les détails ont été reçus, nous vous ajouterons à la liste appropriée. Merci pour votre soutien !

3.6.8 Ajouter un Service à Elgg

Le *guide sur les services* présente des informations générales sur l'utilisation des services Elgg.

Pour ajouter un nouvel objet service à Elgg :

1. Annotez votre classe comme `@internal`.
2. Ouvrez la classe `Elgg\Di\ServiceProvider`.
3. Ajoutez une annotation `@property-read` pour votre service au tout début. Ceci permet aux EDIs et aux analyseurs de code statique de comprendre le type de la propriété.
4. Pour le constructeur, ajoutez le code pour indiquer au fournisseur de service quoi retourner. Voyez la classe `Elgg\Di\DiContainer` pour plus d'information sur comment le conteneur DI (« Dependency Injection ») d'Elgg fonctionne.

A ce stade votre service sera disponible depuis l'objet fournisseur de service, mais ne sera pas encore accessible aux plugins.

Injectez vos dépendances

Concevez votre constructeur de classe de sorte qu'il *demande* les dépendances nécessaires plutôt que de les créer ou d'utiliser `_elgg_services()`. La méthode `setFactory()` du fournisseur de service fournit l'accès à l'instance du fournisseur de service dans la méthode de votre fabrique.

Voici un exemple de fabrique de service `foo`, qui injecte les services `config` et `db` dans le constructeur :

```
// in Elgg\Di\ServiceProvider::__construct()

$this->setFactory('foo', function (ServiceProvider $c) {
    return new Elgg\FooService($c->config, $c->db);
});
```

La liste complète des services internes peut être vue dans les déclarations `@property-read` au début de `Elgg\Di\ServiceProvider`.

Avertissement : Evitez de faire du travail dans le constructeur de votre service, en particulier si cela requiert des requêtes sur la base de données. Actuellement les tests PHPUnit tests ne peuvent pas les effectuer.

Faire qu'un service fasse partie de l'API publique

Si votre service est conçu pour être utilisé par des développeurs de plugins :

1. Faites une interface `Elgg\Services\<Name>` qui ne contient que ces méthodes nécessaires dans l'API publique.
2. Faites que la classe de votre service implémente cette interface.
3. Pour les méthodes qui sont dans l'interface, déplacez la documentation dans l'interface. Vous pouvez simplement utiliser `{@inheritdoc}` dans les PHPDocs des méthodes de votre classe effective.
4. Documentez votre service dans `docs/guides/services.rst` (ce fichier)
5. Ouvrez le test PHPUnit `Elgg\ApplicationTest` et ajoutez la clef de votre service au tableau `$names` dans `testServices()`.
6. Ouvrez la classe `Elgg\Application`.
7. Ajoutez une déclaration `@property-read` pour documenter votre service, mais utilisez votre **interface** pour le type, et *pas* le nom de la classe de votre service.

8. Ajoutez la clef de votre service au tableau dans la propriété `$public_services`, par ex. `'foo' => true`, Désormais votre service sera disponible via l'accès à la propriété sur l'instance `Elgg\Application` :

```
// using the public foo service

```

Note : Pour des exemples, voyez le service `config`, qui comprend l'interface `Elgg\Services\Config` et la classe d'implémentation effective `Elgg\Config`.

Cycle de vie d'un Service et Fabriques (« factories »)

Par défaut, les services enregistrés sur le fournisseur de service sont « partagés », ce qui signifie que le fournisseur de service va conserver l'instance créée pour le reste de la requête, et servir la même instance à tout ce qui demande la propriété.

Si vous avez besoin que les développeurs puissent construire des objets qui soient pré-connectés aux services Elgg, vous pouvez avoir besoin d'ajouter une méthode de fabrique publique à `Elgg\Application`. Voici un exemple qui retourne une nouvelle instance en utilisant les services d'Elgg :

```
public function createFoo($bar) {
    $logger = $this->services->logger;
    $db = $this->services->db;
    return new Elgg\Foo($bar, $logger, $db);
}
```

3.6.9 Écrire des tests

Contenu

- *Vision*
- *Effectuer les tests*
 - *Suite de tests du noyau Elgg*
 - *Tests des plugins*
 - *Tests de bout en bout*
- *Motivation*
- *Stratégie*
 - *Intégration continue*
 - *Injection des dépendances*
 - *Développement orienté comportement*

Vision

Nous voulons *rendre les tests manuels inutiles* pour les développeurs du noyau, les auteurs de plugins, et les administrateurs de sites en rendant possible et en faisant la promotion de tests rapides et automatisés à tous les niveaux de Elgg.

Nous avons hâte d'un monde dans lequel les développeurs du noyau n'ont pas besoin de faire de tests manuels pour vérifier la correction du code contribué à Elgg. De manière similaire, nous envisageons un monde dans lequel les administrateurs de site peuvent mettre à niveau et installer de nouveaux plugins avec la certitude que tout fonctionne bien ensemble.

Effectuer les tests

Suite de tests du noyau Elgg

Actuellement nos tests se partagent en deux parties :

- Les tests PHPUnit sont situés dans `/tests/phpunit` – ceux-ci sont répartis entre les tests unitaires et les tests d'intégration.

Dans la mesure où il existe un fichier de configuration `phpunit.xml` à la racine d'Elgg, les tests devraient être aussi simples que :

```
git clone http://github.com/Elgg/Elgg
cd Elgg
phpunit
```

Tests des plugins

Idéalement, les plugins sont configurés d'une telle manière qu'ils peuvent faire l'objet de tests unitaires de la même manière que le noyau d'Elgg. Les développeurs de plugins sont libres d'implémenter leurs propres méthodes pour les tests unitaires, mais nous encourageons tout le monde à les rendre aussi simple que pour le noyau de Elgg :

```
git clone http://github.com/developer/elgg-plugin plugin
cd plugin
phpunit
```

Tests de bout en bout

Dans la mesure où les plugins de Elgg ont une telle capacité à remplacer, filtrer et modifier Elgg et le comportement d'autres plugins, il est important de pouvoir exécuter des tests de bout en bout sur un serveur de pré-production avec votre configuration finale avant de déployer vers la production.

Note : `ToDo` : Faciliter l'exécution de tous les tests d'intégration et d'acceptabilité depuis le panneau admin en utilisant la configuration courante du plugin. (sans s'inquiéter de corruption de la base de données !)

Motivation

Brièvement, les gains escomptés des tests sont :

- Une confiance accrue dans le système.
- Une plus grande liberté pour remanier le code.
- Une documentation intégrée et à jour.

Nous adorons les contributions communautaires, mais afin de maintenir la stabilité » nous ne pouvons pas accepter de contributions extérieures sans vérifier au préalable leur correction. En promouvant les tests automatisés, les développeurs du noyau peuvent éviter les ennuis d'une vérification manuelle avant d'accepter un patch. Cela signifie également que les développeurs externes n'ont pas à perdre de temps à gagner la confiance de l'équipe du noyau. Si un patch arrive et qu'il dispose de tests pour le vérifier, nous pouvons être confiants dans le fait qu'il fonctionne sans avoir besoin de se soucier de la réputation de l'auteur.

Notez que ces bénéfices peuvent également s'étendre au dépôt des plugins. Les propriétaires de sites sont encouragés à « tester les plugins minutieusement » avant de les déployer sur un site de production. A la date de mars 2013, ceci signifie tester manuellement toutes les fonctionnalités que le plugin promet d'offrir. Mais Elgg fournit un grand nombre de fonctionnalités, et il n'est pas raisonnable de tester *chacune* d'entre elles sur *chaque navigateur* que vous voulez supporter et sur *chaque appareil* que vous voulez supporter ? Mais que se passerait-il si les développeurs de plugins pouvaient écrire des tests pour leurs plugins, et si les propriétaires de sites pouvaient simplement exécuter ces tests pour tous les plugins installés pour vérifier si la fonctionnalité est bien conservée ? Alors ils ne seraient pas limités à seulement prendre des plugins de développeurs « de confiance » ou des versions « stables ». Ils pourraient voir que, en réalité, rien n'a été cassé quand ils ont mis à niveau ce plugin critique depuis la version 1.3 vers la 2.5, et passer la mise à niveau en production en toute confiance.

La raison pour laquelle ce n'est pas le cas actuellement est parce qu'Elgg lui-même n'est pas encore si facile à tester à ce niveau. Nous voulons changer cela.

Stratégie

Nous avons quelques principes pour nous guider, dont nous pensons qu'ils seront utiles pour faire de cette vision une réalité.

En bref, nous prôtons :

- Intégration continue – si les vérifications de Github ne sont pas satisfaisantes, nous ne sommes pas satisfaits
- L'injection des dépendances – Pour créer un code hautement testable et modulaire
- BDD (« Behaviour Driven Development ») – Les tests devraient vérifier des fonctionnalités et fournir de la documentation, pas reprendre l'API de la Classe

Intégration continue

Nous exécutons tous nos tests avec les Actions GitHub afin de pouvoir obtenir des commentaires en temps réel sur l'exactitude des demandes de fusion (pull requests) et du développement au fur et à mesure de sa progression. ** Si les vérifications de GitHub ne réussissent pas, nous ne faisons pas de commit dans le dépôt. ** Cela nous permet de fusionner les demandes de fusion à un rythme rapide, tant qu'elles réussissent les tests. Cela nous permet également de rejeter les demandes de fusion sans examen détaillé si elles ne réussissent pas les tests. Nous pouvons dépasser la question « ça marche ou pas » et parler des choses dont les humains ont besoin de parler : conception d'API, utilité pour le projet, s'il appartient au noyau ou à un plugin, etc. Nous voulons qu'autant de fonctionnalités que possible fournies par le noyau de Elgg (Elgg core) puissent être vérifiées automatiquement par des tests exécutés sur GitHub Actions.

Injection des dépendances

Afin de maximiser la testabilité, **toutes les dépendances doivent être explicites**. Les fonctions globales, les Singletons, et les localiseurs de service sont mortels pour la testabilité parce qu'il est impossible de dire quelles dépendances se cachent sous le capot, et qu'il est encore plus difficile de scénariser (« mock out ») ces dépendances. La capacité de simulation est critique parce que vous voulez que vos tests unitaires ne testent qu'une seule classe à la fois. Les échecs d'un test dans un TestCase ne devraient pas résulter d'une rupture dans une dépendance; les échecs des tests ne devraient indiquer de rupture que dans la classe en train d'être testée. Ceci rend tout plus facile à déboguer. A la date de mars 2013, la majorité d'Elgg assume et utilise toujours un état global, et ceci a rendu Elgg et les plugins Elgg historiquement très difficiles à tester. Fort heureusement nous avons pris une autre direction depuis, et une grande partie du travail dans Elgg 1.9 a consisté à refactorer des composants du noyau pour qu'ils soient plus injectables sous forme de dépendances. Nous récoltons déjà les bénéfices de cet effort.

Développement orienté comportement

Pour nous ceci signifie que **nous nommons les tests d'après les fonctionnalités plutôt que les méthodes**. Quand vous testez des fonctionnalités, vous êtes encouragés à écrire moins de tests, plus petits, et plus logiques. Quand un test échoue, cela permet de savoir exactement quelle fonctionnalité est compromise. De plus, en nommant vos tests d'après les fonctionnalités, la liste des tests fournit une documentation des fonctionnalités que le système supporte. La documentation est typiquement quelque chose de problématique à conserver à jour, mais quand la documentation et la vérification sont une seule et même chose, il devient très facile de maintenir la documentation à jour.

Considérez ces deux méthodes de test :

- `testRegister()`
- `testCanRegisterFilesAsActionHandlers()`

En regardant seulement les noms, `testRegister` vous indique que la classe testée a probablement une méthode nommée « register ». Si ce test est passé, il vérifie probablement qu'elle se comporte correctement, mais ne vous dit pas ce qui caractérise un comportement correct, ou ce que l'auteur original du test essayait de vérifier. Si cette méthode a plusieurs usages corrects que vous devez tester, cette convention de nommage succincte vous encourage également à écrire un très long test qui teste toutes les conditions et fonctionnalités de ladite méthode. L'échec du test pourrait être causée par la compromission de n'importe laquelle d'entre elles, et cela prendra plus de temps pour savoir où se situe le véritable problème.

De l'autre côté, `testCanRegisterFilesAsActionHandlers` vous indique qu'il y a ces choses appelées « actions » qui ont besoin d'être « gérées » (« handled »), et que ces fichiers peuvent être enregistrés comme des gestionnaires valides pour les actions. Ceci expose mieux aux nouveaux venus la terminologie du projet et communique clairement l'intention du test pour ceux qui sont déjà familiers avec la terminologie.

Pour un bon exemple de ce que nous recherchons, regardez `/tests/phpunit/Elgg/ViewServiceTest.php`

3.6.10 Tâches du noyau

Certaines tâches autour d'Elgg sont réservées à l'équipe des développeurs du noyau dans la mesure où elles demandent certaines permissions spéciales. Les guides ci-dessous décrivent le processus pour ces actions.

Déplacer un plugin vers son propre dépôt

Contenu

- *Étapes d'extraction d'un plugin*
 - *Déplacez le code vers son propre dépôt*
 - *Dépendances*
 - *Commitez le code*
 - *Packagist*
 - *Tagguer une release*
 - *Traductions*
- *Nettoyage du noyau Elgg*
 - *Supprimez le plugin*
 - *Traductions*
 - *Plugins joints*
 - *Composer*
 - *Documentation*

Étapes d'extraction d'un plugin

Déplacez le code vers son propre dépôt

Suivez le guide Github pour transformer un sous-dossier en un nouveau dépôt ([Splitting a subfolder out into a new repository](#)). Ceci va faire en sorte que l'historique des commits soit préservé.

Dépendances

Si le plugin a des dépendances sur n'importe quelles bibliothèques externes, assurez-vous que ces dépendances soient gérées. Par exemple, si une bibliothèque PHP est requise qui est livrée avec le noyau d'Elgg, assurez-vous de l'ajouter dans le `composer.json` de ce plugin dans la mesure où vous ne pouvez pas vous appuyer sur le fait que le noyau d'Elgg conserve cette bibliothèque.

Commitez le code

Au cours du guide GitHub un nouveau dépôt est créé pour le plugin que vous essayez de déplacer.

Dans la mesure où une tentative a déjà été faite pour extraire certains des plugins vers leur propre dépôt il se peut que le dépôt existe déjà.

Si le dépôt n'existait pas pour le plugin, assurez-vous que vous le créez dans [Elgg organisation](#).

Si le dépôt existe déjà, le meilleur moyen de mettre à jour le code serait en faisant une Pull Request. Ceci risque cependant probablement d'échouer à cause d'une différence dans la manière dont le dépôt a été initialement créé (comme discuté dans [Problème avec GitHub](#)).

Les dépôts initiaux ont été créés avec

```
git subtree split
```

et le guide appelle

```
git filter-branch --prune-empty --subdirectory-filter
```

Ceci va laisser une différence dans les commits que Github est incapable de résoudre. Dans ce cas vous devrez forcer un push des modifications vers le dépôt du plugin Elgg existant.

Avvertissement : Dans la mesure où cela va réécrire tout l'historique du dépôt du plugin, assurez-vous que vous savez que c'est bien ce que vous souhaitez faire.

Packagist

Assurez-vous que le `composer.json` du plugin contient bien toutes les informations pertinentes. Voici un exemple :

```
{
    "name": "elgg/<name of the repository>",
    "description": "<a description of the plugin>",
    "type": "elgg-plugin",
    "keywords": ["elgg", "plugin"],
    "license": "GPL-2.0-only",
    "support": {
        "source": "https://github.com/elgg/<name of the repository>",
        "issues": "https://github.com/elgg/<name of the repository>/issues"
    },
    "require": {
        "composer/installers": ">=1.0.8"
    },
    "conflict": {
        "elgg/elgg": "< <minimal Elgg required version>"
    }
}
```

La règle `conflict` est ici pour aider à éviter l'installation de ce plugin dans une version d'Elgg non supportée.

Ajoutez le dépôt à [Packagist](#), pour les dépôts existants ceci a déjà été fait. Assurez-vous que [Packagist](#) est mis à jour correctement avec tous les commits.

Tagguer une release

Afin que Composer soit capable de mettre en cache le plugin pour une installation plus rapide, une release doit être créée sur le dépôt. A priori, la première version qui doit être tagguée est la même version que celle mentionnée dans le `manifest.xml`. Après cela le développement peut commencer, suivi par le schéma de version [Semver](#).

Traductions

Si les traductions pour le plugin doivent être gérées par [Transifex](#), ajoutez le plugin à [Transifex](#).

Nettoyage du noyau Elgg

Maintenant que le plugin a été déplacé dans son propre dépôt, il est temps de faire une Pull Request sur le noyau d'Elgg pour supprimer le code originel.

Supprimez le plugin

- Supprimez le plugin du dossier `mod`
- Recherchez le nom du plugin dans le noyau pour trouver toute référence qui aurait également besoin d'être retirée (par ex. anciennes docs, tests spéciaux, etc.)

Traductions

Dans la mesure où le plugin ne fait plus partie du noyau Elgg, assurez-vous que la configuration de [Transifex](#) ne contient plus le plugin.

Plugins joints

Si le plugin est toujours livrés conjointement avec la publication d'une nouvelle version de Elgg, assurez-vous d'ajouter le plugin dans `composer.json`.

Composer

Vérifiez les dépendances `composer` du noyau pour identifier si des exigences qui étaient spécifiques pour le plugin retiré peuvent également être retirées des dépendances du noyau.

Documentation

Ajoutez une mention dans la documentation *Notes de mise à niveau* qui indique que le plugin a été retiré du noyau de Elgg.

Processus de publication d'une release

Publier une nouvelle version d'Elgg.

Voici le processus suivi par l'équipe du noyau pour publier une nouvelle release d'Elgg. Nous publions cette information dans un esprit d'ouverture, et pour faciliter l'intégration de nouveaux membres de l'équipe.

Contenu

- *Pré-requis*
- *Fusionnez les commits à partir des branches les plus basses*
 - *Pour chaque branche*
- *Préparation pour la première nouvelle version stable mineure/majeure*
 - *Préparation pour une nouvelle version majeure*
- *Préparer la sortie*
 - *Faites un PR avec des mises à jour de traduction*
 - *Faire le PR de la version*
- *Tagguez la release*
 - *Actions supplémentaires pour la première nouvelle mineure / majeure*
 - *Action supplémentaire pour la première nouvelle majeure*
- *Mettez à jour le site web*
 - *Construisez le package zip*
 - *Mettez à jour la page de téléchargement de elgg.org*
 - *Mettre à jour elgg.org*
- *Faites l'annonce*

Pré-requis

- Accès SSH à elgg.org
- Accès aux commits sur <http://github.com/Elgg/Elgg>
- Accès admin à <https://elgg.org/>
- Accès au compte Twitter [Twitter account](#)
- Node.js et Yarn installés
- Sphinx installé (`easy_install sphinx && easy_install sphinx-intl`)
- Client Transifex installé (`easy_install transifex-client`)
- Compte Transifex avec accès au projet Elgg

Fusionnez les commits à partir des branches les plus basses

Déterminez la branche LTS. Nous devons fusionner tous les nouveaux commits jusqu'ici en passant par les autres branches.

Pour chaque branche

Vérifiez la branche, assurez-vous qu'elle est à jour, et créez une nouvelle branche de travail avec la fusion. Par ex. ici nous fusionnons les commits de la 1.12 dans la 2.0 :

```
git checkout 2.0
git pull
git checkout -b merge112
git merge 1.12
```

Note : Si elle est déjà à jour (aucun commit à fusionner), nous pouvons nous arrêter ici pour cette branche.

S'il y a des conflits, résolvez-les, `git add .`, et `git merge`.

Faites un PR pour la branche et attendez le résultat des tests automatiques et l'approbation par un ou plusieurs autre(s) dev(s).

```
git push -u my_fork merge112
```

Une fois la fusion effectuée, nous répéterions le processus pour fusionner les commits de la 2.0 dans la 2.1.

Préparation pour la première nouvelle version stable mineure/majeure

- Mettez à jour *Politique de support* pour inclure la date de la nouvelle version mineure/majeure et complétez les vides de la version précédente.
- Mettez à jour les badges du fichier README.md pour indiquer les nouveaux numéros de version corrects.

Préparation pour une nouvelle version majeure

- Modifiez la configuration Transifex pour pousser les traductions vers un autre projet

Préparer la sortie

Faites un PR avec des mises à jour de traduction

Installez les pré-requis :

```
easy_install transifex-client
```

Note : Sur Windows, vous devrez exécuter ces commandes dans une console avec des privilèges admin

Exécutez le script `languages.php`. Par exemple, pour extraire les traductions :


```
php .scripts/languages.php 3.x
```

Faites une demande de fusion (PR) avec les nouvelles traductions et faites-la fusionner avant l'étape suivante.

Ensuite, accédez manuellement à la page `/admin/site_settings` et vérifiez qu'elle se charge. Si ce n'est pas le cas, un fichier de langue de Transifex peut avoir une erreur de syntaxe PHP. Corrigez l'erreur et modifiez votre validation avec le nouveau fichier :

```
# only necessary if you fixed a language file
git add .
git commit --amend
```

Faire le PR de la version

Mettez à jour votre clone git local.

Fusionnez les derniers commits vers le haut en commençant par la branche supportée la plus basse.

Visitez <https://github.com/Elgg/Elgg/compare/<new>...<old>> et soumettez le PR si quoi que ce soit a besoin d'être fusionné plus haut.

Installez les pré-requis :

```
yarn install elgg-conventional-changelog
easy_install sphinx
easy_install sphinx-intl
```

Note : Sur Windows, vous devrez exécuter ces commandes dans une console avec des privilèges admin

Exécutez le script `release.php`. Par exemple, pour publier 1.12.5 :

```
git checkout 1.12
php .scripts/release.php 1.12.5
```

Ceci crée une branche `release-1.12.5` dans votre repo local.

Ensuite, soumettez une demande de fusion (PR) via GitHub pour les tests automatisés et l'approbation par un autre développeur :

```
git push your-remote-fork release-1.12.5
```

Tagguez la release

Une fois approuvée et fusionnée, tagguez les release :

```
git checkout release-${version}
git tag -a ${version} -m'Elgg ${version}'
git push --tags origin release-${version}
```

Ou créez une release sur Github

- Allez sur les releases
- Cliquez sur "Create a new release"

- Saisissez la version
- Sélectionnez la bonne branche (par ex. 1.12 pour une release 1.12.x, 2.3 pour une release 2.3.x, etc.)
- Définissez le titre de la release tel que “Elgg {version}”
- Collez la partie de CHANGELOG.md relative à cette release dans la description

Un peu d’administration pour finir

- Marquez les jalons de release Github comme terminés
- Déplacez les tickets non résolus des jalons publiés vers des jalons ultérieurs

Actions supplémentaires pour la première nouvelle mineure / majeure

- Créez une nouvelle branche sur GitHub (par exemple 3.3)
- Définissez la nouvelle branche comme branche par défaut (facultatif, mais suggéré pour les versions stables)
- Configurer Lire la documentation pour construire la nouvelle branche (et non la nouvelle balise)
- Vérifiez le projet de démarrage Elgg pour les besoins potentiels / modifications de configuration dans le `composer.json`
- Ajoutez la nouvelle version mineure/majeure au référentiel Elgg/community_plugins afin que les développeurs puissent télécharger des plugins pour la nouvelle version

Action supplémentaire pour la première nouvelle majeure

- Sur GitHub ajouter une règle de protection de branche (par exemple 4.*)
- Configurer Scrutinizer pour suivre les nouvelles branches principales (par exemple 4.*)

Mettez à jour le site web

- ssh vers `elgg.org`
- Clonez <https://github.com/Elgg/elgg-scripts>

Construisez le package zip

Utilisez `elgg-scripts/build/elgg-starter-project.sh` pour générer le fichier `.zip`. Exécutez sans argument pour voir son utilisation.

```
# login as user deploy
sudo -su deploy

# regular release
./elgg-starter-project.sh master 3.0.0 /var/www/www.elgg.org/download/

# MIT release
./elgg-starter-project.sh master 3.0.0-mit /var/www/www.elgg.org/download/
```

Note : Pour les releases Elgg 2.x utilisez la branche 2.x du starter-project (par ex. `./elgg-starter-project.sh 2.x 2.0.4 /var/www/www.elgg.org/download/`)

- Vérifiez que `vendor/elgg/elgg/composer.json` dans le fichier zip a bien la version attendue.
- Si ce n’est pas le cas, assurez-vous que GitHub a bien le tag de release, et que le projet de démarrage a un élément compatible `elgg/elgg` dans la liste « requires » de `composer`.

Mettez à jour la page de téléchargement de elgg.org

- Clonez <https://github.com/Elgg/community>
- Ajoutez la nouvelle version dans `classes/Elgg/Releases.php`
- Committez et poussez les modifications (push)
- Mettez à jour le plugin sur www.elgg.org

```
composer update elgg/community
```

Mettre à jour elgg.org

- Cloner <https://github.com/Elgg/www.elgg.org>
- Modifiez la version d'Elgg requise dans `composer.json`
- Mettre à jours vendors

```
composer update
```

- Committez et poussez les modifications (push)
- Faites un pull vers le site en activité

```
cd /var/www/www.elgg.org && sudo su deploy && git pull
```

- Mettez à jour les dépendances

```
composer install --no-dev --prefer-dist --optimize-autoloader
```

- **Allez sur la panneau admin de la communauté**
 - Vider le cache APC
 - Exécuter la mise à niveau

Faites l'annonce

Ce devrait être la toute dernière chose à faire.

1. Ouvrez <https://github.com/Elgg/Elgg/blob/<tag>/CHANGELOG.md> et copiez le contenu pour cette version
2. Connectez-vous sur <https://elgg.org/blog> et rédigez un nouvel article de blog avec un sommaire
3. Copiez le contenu dans le CHANGELOG, supprimez le formatage, et supprimez manuellement les balises SVG
4. Ajoutez les tags `release` et `elgg2.x` où `x` est le nom de la branche en train d'être publiée
5. Tweetez depuis le [compte Twitter account](#) d'Elgg

3.7 Annexes

Informations diverses sur le projet.

3.7.1 Notes de mise à niveau

Si vous mettez à niveau vos plugins et votre site Web vers une nouvelle version majeure de Elgg, les modifications notables suivantes s'appliquent.

Voir les guides d'administration pour savoir *comment mettre à niveau un site en production*.

De 3.2 à 3.3

Contenus

- *Version de PHP*
- *Utilisation plus simple de la disposition "default"*
- *Noms de disposition dépréciés*
- *Modifications du manifeste du plugin*
- *APIs dépréciées*
- *Valeurs de configuration obsolètes*
- *Commandes CLI obsolètes*
- *Hooks obsolètes*

Version de PHP

PHP 7.1 a atteint sa *fin de vie* en décembre 2019. Pour assurer la sécurité des sites Elgg, nous demandons maintenant au moins PHP 7.2 pour les nouvelles installations.

Si vous faites une mise à niveau depuis une installation de Elgg précédente, assurez-vous que vous avez la bonne version PHP installée.

Afin de pouvoir tester Elgg sur PHP 7.4, nous avons dû mettre à jour la suite de test PHPUnit à la version 8.5. Cela peut nécessiter quelques réécritures de vos propres tests d'unité/intégration.

Utilisation plus simple de la disposition "default"

Actuellement, le modèle le plus courant est de faire ce qui suit :

```
$title = 'All blogs';

$content = elgg_list_entities([
    'type' => 'object',
    'subtype' => 'blog',
]);

$layout = elgg_view_layout('default', [
    'title' => $title,
    'content' => $content,
```

(suite sur la page suivante)

(suite de la page précédente)

```
]);

echo elgg_view_page($title, $layout);
```

Nous avons simplifié ce genre de modèle. Vous pouvez maintenant passer un tableau d’options de disposition au deuxième paramètre de *elgg_view_page*. La fonction *elgg_view_page* utilisera ce tableau pour l’envelopper dans la disposition “default”. Elle ajoute également le titre de la page à la mise en page.

La nouvelle façon de le faire est :

```
echo elgg_view_page('All blogs', [
    'content' => elgg_list_entities([
        'type' => 'object',
        'subtype' => 'blog',
    ]),
]);
```

Noms de disposition dépréciés

Pour faciliter la transition de Elgg 2.x à 3.x, nous avons gardé certains anciens noms de mise en page (“one_sidebar”, “one_column”, “two_sidebar” et “content”) intacts lorsqu’ils sont utilisés dans la fonction *elgg_view_layout*. À partir d’Elgg 3.3, ces noms de disposition sont obsolètes et vous devez mettre à jour votre code pour utiliser la nouvelle disposition “default”.

Lorsque vous modifiez l’utilisation du nom de disposition “content”, vous devez tenir compte du fait que le “content” génère un menu de filtre. Ce menu est légèrement différent du menu “filter” habituel généré dans la disposition “default”. Il utilise d’autres noms de hook. Si vous avez déjà désactivé le filtre dans votre disposition, vous pouvez modifier la disposition sans aucun problème.

Modifications du manifeste du plugin

Nous travaillons à la suppression du fichier de manifeste de plugin. Certaines caractéristiques du manifeste seront remplacées et d’autres seront abandonnées. Pour faciliter un peu cette transition, nous avons déjà déprécié les fonctions suivantes de l’api *ElggPluginManifest* :

- *getCopyright()*
- *getDonationsPageURL()*
- *getSuggests()* utilisez les suggestions dans composer

APIs dépréciées

- *elgg_disable_query_cache()*
- *elgg_enable_query_cache()*
- *elgg_format_attributes()* utiliser *elgg_format_element()*
- *elgg_flush_caches()* utiliser *elgg_clear_caches()*
- *elgg_get_menu_item()*
- *elgg_get_ordered_event_handlers()* utilisez *elgg()->events->getOrderedHandlers()*
- *elgg_get_ordered_hook_handlers()* utilisez *elgg()->hooks->getOrderedHandlers()*
- *elgg_invalidate_simplecache()* utilisez *elgg_clear_caches()*
- *elgg_is_menu_item_registered()*
- *elgg_view_entity_annotations()*

- `execute_delayed_write_query()` utilisez `elgg()->db->registerDelayedQuery()`
- `execute_delayed_read_query()` utilisez `elgg()->db->registerDelayedQuery()`
- `run_sql_script()`
- `elgg_unset_all_plugin_settings()` utilisez `\ElggPlugin::unsetAllSettings()`
- `elgg_get_file_simple_type()` utilisez `elgg()->mimetype->getSimpleType()`
- `ElggFile::detectMimeType()` utilisez `elgg()->mimetype->getMimeType()`
- `generate_action_token()` utilisez `elgg()->csrf->generateActionToken()`
- `elgg_split()`

Valeurs de configuration obsolètes

- `simplecache_lastupdate` utiliser `lastcache`

Commandes CLI obsolètes

- `elgg-cli flush` utiliser `elgg-cli cache:clear`

Hooks obsolètes

- `entity:annotate, <entity_type>`

De 3.1 à 3.2

Contenus

- *Accès en écriture de l'utilisateur*
- *État activé des éléments de la rivière*

Accès en écriture de l'utilisateur

Pour résoudre un problème où des collections d'accès appartenant à l'utilisateur comme *Friends* ou *Friend Collections* s'affichent toujours dans la liste déroulante d'accès lors de la création de contenu, même si les plugins connexes sont désactivés, nous avons dû modifier une logique interne. Si vous souhaitez avoir un sous-type de collection d'accès disponible dans l'accès à l'écriture, vous devez maintenant enregistrer le sous-type avec un hook plugin. Voir [la référence des hooks plugin pour "access:collections:write:subtypes"](#) pour plus d'informations.

État activé des éléments de la rivière

L'état *enabled* (activé) des éléments de la rivière a été déprécié. Vous ne devez plus utiliser cette propriété lorsque vous travaillez avec des éléments de la rivière.

De 3.0 à 3.1

Contenus

- *Version de PHP*
- *Captures d'écran du plugin*
- *Chargement de fichiers externes*
- *Définition du propriétaire de la page*
- *Simpletests*
- *Fonctions de rappel de hook et d'événement*
- *Routes obsolètes*
- *Bibliothèques CSS obsolètes*
- *Bibliothèques JS obsolètes*
- *APIs dépréciées*
- *Actions obsolètes*

Version de PHP

PHP 7.0 a atteint sa fin de vie en janvier 2019. Pour assurer la sécurité des sites Elgg, nous demandons maintenant au moins PHP 7.1 pour les nouvelles installations.

Si vous faites une mise à niveau depuis une installation de Elgg précédente, assurez-vous que vous avez la bonne version PHP installée.

Captures d'écran du plugin

Les captures d'écran ajoutées aux plugins ne sont plus prises en charge et ne seront plus affichées dans les détails du plugin.

Chargement de fichiers externes

L'utilisation de `elgg_register_js`, de `elgg_unregister_js` et de `elgg_load_js` est déconseillée. Assurez-vous que votre javascript est un module AMD et utilisez `elgg_require_js` pour l'inclure.

L'utilisation de `elgg_register_css`, de `elgg_unregister_css` et de `elgg_load_css` est déconseillée. Vous pouvez enregistrer et inclure les css avec la nouvelle fonction `elgg_require_css`.

Définition du propriétaire de la page

La définition du propriétaire de la page via un paramètre de la fonction `elgg_get_page_owner_guid` est obsolète. Utilisez `elgg_set_page_owner_guid`.

Simpletests

Les tests simples du noyau ont été supprimés du système. Ils sont tous remplacés par des tests unitaires PHP ou des tests d'intégration. La commande cli simpletest est devenue obsolète.

Fonctions de rappel de hook et d'événement

Les arguments des hooks et des fonctions de rappel d'origine sont déconseillés. Vous devez passer au nouveau style d'arguments dès que possible.

```
// old style hook callback
function hook_callback($name, $type, $return_value, $params) {

}

// new style hook callback
function hook_callback(\Elgg\Hook $hook) {
    // now you can use a few new functions
    $params = $hook->getParams();
    $return = $hook->getValue();

    $specific_param = $hook->getParam('specific_param', 'default')
    $entity = $hook->getEntityParam();
    $user = $hook->getUserParam();
}

// old style event callback
function event_callback($name, $type, $object) {

}

// new style event callback
function event_callback(\Elgg\Event $event) {
    // now you can use a few new functions
    $object = $event->getObject();
}
```


Routes obsolètes

- `previous:object:thewire` Cette route n'était pas utilisée. Elle a maintenant été marquée comme obsolète.

Bibliothèques CSS obsolètes

- `jquery.imgareaselect` N'utilisez pas ce fichier css externe.
- `jquery.treeview` N'utilisez pas ce fichier css externe.

Bibliothèques JS obsolètes

- `elgg.avatar_cropper` Évitez de dépendre de cette bibliothèque javascript externe.
- `jquery.imgareaselect` Évitez de dépendre de cette bibliothèque javascript externe.
- `jquery.treeview` Évitez de dépendre de cette bibliothèque javascript externe.

APIs dépréciées

- `access_show_hidden_entities()` Utilisez `elgg_call()` avec le drapeau `ELGG_SHOW_DISABLED_ENTITIES`.
- `autoregister_views()`
- `count_messages()` Utilisez `elgg()->system_messages->count()`.
- `disable_user_entities()`
- `elgg_enable_entity()` Utilisez `ElggEntity::enable()`.
- `elgg_get_file_list()` Utilisez un itérateur de répertoire PHP.
- `elgg_instanceof()` Utilisez l'opérateur de type de PHP `instanceof`.
- `elgg_is_admin_user()` Utilisez `ElggUser::isAdmin()`.
- `elgg_set_ignore_access()` Utilisez `elgg_call()` avec le drapeau `ELGG_IGNORE_ACCESS`.
- `elgg_sort_3d_array_by_value()`
- `get_access_list()` Utilisez `get_access_array()`.
- `get_language()` Utilisez `get_current_language()`.
- `get_number_users()` Utilisez `elgg_count_entities()`.
- `pages_register_navigation_tree()`
- `ini_get_bool()`
- `is_not_null()`
- `update_access_collection()`

Actions obsolètes

- `admin/delete_admin_notice` Remplacé par une action générique `entity/delete`.
- `avatar/crop` Géré dans l'action `avatar/upload`.
- `avatar/remove` Géré dans l'action `avatar/upload`.
- `blog/delete` Remplacé par l'action générique `entity/delete`.
- `messages/delete` Remplacé par l'action générique `entity/delete`.
- `site_notifications/delete` Remplacé par l'action générique `entity/delete`.

De 2.x à 3.0

Contenus

- *PHP 7.0 est désormais requis*
- *\$CONFIG est supprimé !*
- *Vues supprimées*
- *Fonctions/méthodes supprimées*
- *APIs dépréciées*
- *Variables globales supprimées*
- *Classes/interfaces supprimées*
- *Modifications de schéma*
- *Changements dans les fonctions de récupération `elgg_get_entities`, `elgg_get_metadata` et `elgg_get_annotations`*
- *Propriétés de l'entité booléenne*
- *Modifications des métadonnées*
- *Permissions et droits d'accès*
- *Modifications multisites*
- *Modifications des sous-tables des entités*
- *Collection d'accès des Contacts et des Groupes*
- *Les sous-types n'ont plus d'ID*
- *Chargement de classe personnalisée*
- *Conteneur d'injection de dépendance (DI = Dependency Injection)*
- *Modifications de la recherche*
- *Modifications liées aux formulaires et aux champs*
- *Modifications du menu des entités et de la rivière*
- *Bibliothèques supprimées*
- *Gestionnaires de pages supprimés*
- *Actions supprimées*
- *Modifications de l'héritage*
- *APIs JavaScript supprimées*
- *Hooks/événements retirés*
- *Formulaires/actions supprimés*
- *APIs qui n'acceptent plus qu'un tableau `$options`*
- *Fonctions plugin qui nécessitent maintenant un `$plugin_id` explicite*
- *Constructeurs de classes qui n'acceptent désormais qu'un objet `stdClass` ou `null`*
- *Modifications d'API diverses*
- *Le comportement d'extension des vues a été modifié*
- *L'ordre d'appel des hooks JavaScript pourrait changer*
- *Modifications liées à la disposition widget*
- *Routage*
- *Étiquetage*
- *Filtrage des valeurs de la requête*
- *Réponses des actions*
- *HtmLawed n'est plus un plugin*
- *Nouvelle approche des mises en page (layouts)*
- *Plugin Likes*
- *Plugin Notifications*
- *Plugin Pages*
- *Plugin Profile*
- *Plugin Data Views (vues de données)*
- *Plugin Twitter API*

- *Plugin des URLs d'origine*
- *Plugin de validation de l'utilisateur par email*
- *Livraison d'e-mail*
- *Changements de thème et de style*
- *Commentaires*
- *Vues de listing d'objets*
- *Modifications des menus*
- *Icônes des entités*
- *Glyphes d'icône*
- *Autocomplétion / saisie automatique (sélecteurs d'utilisateurs et de contacts)*
- *Collections de Contacts*
- *Mise en page des éléments .elgg-body*
- *Supprimer des éléments de la rivière*
- *Les réponses aux discussions ont été converties en commentaires*
- *Nettoyage des traductions*
- *Journal système*
- *Journalisation des erreurs*
- *Le plugin d'actifs de composer n'est plus requis*
- *Journaux du Cron*
- *Clefs de traduction supprimées / modifiées*
- *Les nouvelles fonctionnalités du schéma MySQL ne sont pas appliquées*
- *Changements divers*
- *Plugin Twitter API*
- *Tests Unitaires et d'Intégration*

PHP 7.0 est désormais requis

5.6 atteint sa fin de vie. PHP 7.0 est maintenant nécessaire pour installer et exécuter Elgg.

\$CONFIG est supprimé !

Pas exactement, cependant vous **devez** auditer cet usage et *devriez* le remplacer par `elgg_get_config()` et `elgg_set_config()`, comme recommandé depuis Elgg 1.9.

Le \$CONFIG global est maintenant un proxy pour le conteneur de configuration d'Elgg, et les modifications **échouent** si vous essayez de modifier directement les propriétés du tableau. Par exemple, `$CONFIG->cool_fruit [] = ' Pear' ;`. Le bon côté est que ces échecs émettront des NOTICES.

Vues supprimées

- `forms/account/settings` : l'extension de `usersettings` peut désormais étendre l'affichage `forms/usersettings/save`
- `forms/admin/site/advanced/system`
- `resources/file/download`
- `output/checkboxes` : utilisez `output/tags` si vous voulez le même comportement
- `input/write_access` : `mod/pages` utilise désormais le hook de plugin `access :collections :write`.
- `invitefriends/form`
- `page/layouts/content` : utilisez `page/layouts/default`
- `page/layouts/one_column` : utilisez `page/layouts/default`
- `page/layouts/one_sidebar` : utilisez `page/layouts/default`

- `page/layouts/two_sidebar` : utilisez `page/layouts/default`
- `page/layouts/walled_garden` : utilisez `page/layouts/default`
- `page/layouts/walled_garden/cancel_button`
- `page/layouts/two_column_left_sidebar`
- `page/layouts/widgets/add_panel`
- `page/elements/topbar_wrapper` : mettez à jour votre utilisation de `page/elements/topbar` pour inclure une vérification si l'utilisateur est connecté
- `pages/icon`
- `groups/group_sort_menu` : utilisez le hook plugin `register`, `filter:menu:groups/all`
- `groups/my_status`
- `groups/profile/stats`
- `subscriptions/form/additions` : étendez plutôt `notifications/parameters/other`
- `likes/count` : des modifications peuvent désormais être apportées à l'élément de menu `likes_count`
- `likes/css` : les likes utilisent désormais `elgg/likes.css`
- `resources/members/index`
- `messageboard/css`
- `notifications/subscriptions/personal`
- `notifications/subscriptions/collections`
- `notifications/subscriptions/form`
- `notifications/subscriptions/jsfuncs`
- `notifications/subscriptions/forminternals`
- `notifications/css`
- `pages/input/parent`
- `river/item` : utilisez `elgg_view_river_item()` pour rendre les articles de la rivière
- `river/user/default/profileupdate`
- `admin.js`
- `aalborg_theme/homepage.png`
- `aalborg_theme/css`
- `resources/avatar/view` : Utilisez l'API des icônes des entités
- `ajax_loader.gif`
- `button_background.gif`
- `button_graduation.png`
- `elgg_toolbar_logo.gif`
- `header_shadow.png`
- `powered_by_elgg_badge_drk_bckgnd.gif`
- `powered_by_elgg_badge_light_bckgnd.gif`
- `sidebar_background.gif`
- `spacer.gif`
- `toptoolbar_background.gif`
- `two_sidebar_background.gif`
- `ajax_loader_bw.gif` : utilisez `graphics/ajax_loader_bw.gif`
- `elgg_logo.png` : utilisez `graphics/elgg_logo.png`
- `favicon-128.png` : utilisez `graphics/favicon-128.png`
- `favicon-16.png` : utilisez `graphics/favicon-16.png`
- `favicon-32.png` : utilisez `graphics/favicon-32.png`
- `favicon-64.png` : utilisez `graphics/favicon-64.png`
- `favicon.ico` : utilisez `graphics/favicon.ico`
- `favicon.svg` : utilisez `graphics/favicon.svg`
- `friendpicker.png` : utilisez `graphics/friendpicker.png`
- `walled_garden.jpg` : utilisez `graphics/walled_garden.jpg`
- `core/friends/collection`
- `core/friends/collections`
- `core/friends/collectiontabs`
- `core/friends/tablelist`

- `core/friends/talbelistcountupdate`
- `lightbox/elgg-colorbox-theme/colorbox-images/*``
- `navigation/menu/page` : utilisez désormais `navigation/menu/default` et un hook de préparation
- `navigation/menu/site` : utilisez désormais la vue par défaut
- `page/elements/by_line` : Use `object/elements/imprint`
- `forms/admin/site/advanced/security` : les informations secrètes du site ont été déplacées vers `forms/admin/security/settings`
- `river/object/file/create` : regardez `:doc:/guides/river`
- `river/object/page/create` : regardez *Rivière (Flux d'activité)*
- `river/object/page_top/create` : regardez *Rivière (Flux d'activité)*
- `river/relationship/member` : regardez *Rivière (Flux d'activité)*
- `object/page_top` : utilisez `object/page`
- `ajax/discussion/reply/edit` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion/replies` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `object/discussion_reply` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `resources/discussion/reply/edit` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `resources/elements/discussion_replies` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `river/elements/discussion_replies` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `river/object/discussion/create`
- `river/object/discussion_reply/create` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `search/object/discussion_reply/entity` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `rss/discussion/replies` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `search/header`
- `search/layout` dans les types de vues `default` et `rss`
- `search/no_results`
- `search/object/comment/entity`
- `search/css` : Déplacé vers `search/search.css`
- `search/startblurb`
- `bookmarks/bookmarklet.gif`
- `blog_get_page_content_list`
- `blog_get_page_content_archive`
- `blog_get_page_content_edit`
- `forms/invitefriends/invite` : utilisez `forms/friends/invite`
- `resources/invitefriends/invite` : utilisez `resources/friends/invite`
- `resources/reportedcontent/add`
- `resources/reportedcontent/add_form`
- `resources/site_notifications/view` : Utilisez `resources/site_notifications/owner`
- `resources/site_notifications/everyone` : Utilisez `resources/site_notifications/all`

Fonctions/méthodes supprimées

Toutes les fonctions dans `engine/lib/deprecated-1.9.php` ont été supprimées. Voir <https://github.com/Elgg/Elgg/blob/2.0/engine/lib/deprecated-1.9.php> pour ces fonctions. Chaque déclaration `@deprecated` comprend des instructions sur ce qu'il faut utiliser à la place. Toutes les fonctions de `engine/lib/deprecated-1.10.php` ont été supprimées. Voir <https://github.com/Elgg/Elgg/blob/2.0/engine/lib/deprecated-1.10.php> pour ces fonctions. Chaque déclaration `@deprecated` comprend des instructions sur ce qu'il faut utiliser à la place.

- `elgg_register_library` : exigez (require) vos bibliothèques afin qu'ils soient disponibles globalement pour d'autres plugins
- `elgg_load_library`
- `activity_profile_menu`
- `can_write_to_container` : Utilisez `ElggEntity->canWriteToContainer()`
- `create_metadata_from_array`
- `metadata_array_to_values`
- `datalist_get`
- `datalist_set`
- `detect_extender_valuetype`
- `developers_setup_menu`
- `elgg_disable_metadata`
- `elgg_enable_metadata`
- `elgg_get_class_loader`
- `elgg_get_metastring_id`
- `elgg_get_metastring_map`
- `elgg_register_class`
- `elgg_register_classes`
- `elgg_register_viewtype`
- `elgg_is_registered_viewtype`
- `file_delete` : Utilisez `ElggFile->deleteIcon()`
- `file_get_type_cloud`
- `file_type_cloud_get_url`
- `get_default_filestore`
- `get_site_entity_as_row`
- `get_group_entity_as_row`
- `get_missing_language_keys`
- `get_object_entity_as_row`
- `get_user_entity_as_row`
- `update_river_access_by_object`
- `garbagecollector_orphaned_metastrings`
- `groups_access_collection_override`
- `groups_get_group_tool_options` : Utilisez `elgg()->group_tools->all()`
- `groups_join_group` : Utilisez `ElggGroup::join`
- `groups_prepare_profile_buttons` : Utilisez le hook `register`, menu:title
- `groups_register_profile_buttons` : Utilisez le hook `register`, menu:title
- `groups_setup_sidebar_menus`
- `groups_set_icon_url`
- `groups_setup_sidebar_menus`
- `messages_notification_msg`
- `set_default_filestore`
- `generate_user_password` : Utilisez `ElggUser::setPassword`
- `row_to_elggrelationship`
- `run_function_once` : Utilisez l'interface `Elgg\Upgrade\Batch`
- `system_messages`
- `notifications_plugin_pagesetup`
- `elgg_format_url` : Utilisez `elgg_format_element()` ou la vue « `output/text` » pour échapper le HTML.

- `get_site_by_url`
- `elgg_override_permissions` : N'est plus utilisé comme gestionnaire pour les hooks `permissions_check` et `container_permissions_check`
- `elgg_check_access_overrides`
- `AttributeLoader` est devenu obsolète et a été supprimé
- `Application::loadSettings`
- `ElggEntity::addToSite`
- `ElggEntity::disableMetadata`
- `ElggEntity::enableMetadata`
- `ElggEntity::getSites`
- `ElggEntity::removeFromSite`
- `ElggEntity::isFullyLoaded`
- `ElggEntity::clearAllFiles`
- `ElggPlugin::getFriendlyName` : Utilisez `ElggPlugin::getDisplayname()`
- `ElggPlugin::setID`
- `ElggPlugin::unsetAllUsersSettings`
- `ElggFile::setFilestore` : les objets `ElggFile` ne peuvent plus utiliser de répertoire de fichiers personnalisé.
- `ElggFile::size` : Utilisez `getSize`
- `ElggDiskFilestore::makeFileMatrix` : Utilisez `Elgg\EntityDirLocator`
- `ElggData::get` : Peut habituellement être remplacé par la lecture de propriété
- `ElggData::getClassName` : Utilisez `get_class()`
- `ElggData::set` : Peut habituellement être remplacé par l'écriture de propriété
- `ElggEntity::setURL` : Voyez `getURL` pour plus de détails sur le hook plugin
- `ElggMenuBuilder::compareByWeight` : Utilisez `compareByPriority`
- `ElggMenuItem::getWeight` : Utilisez `getPriority`
- `ElggMenuItem::getContent` : Utilisez `elgg_view_menu_item()`
- `ElggMenuItem::setWeight` : Utilisez `setPriority`
- `ElggRiverItem::getPostedTime` : Utilisez `getTimePosted`
- `ElggSession` a supprimé toutes les méthodes dépréciées
- `ElggSite::addEntity`
- `ElggSite::addObject`
- `ElggSite::addUser`
- `ElggSite::getEntities` : Utilisez `elgg_get_entities()`
- `ElggSite::getExportableValues` : Utilisez `toObject`
- `ElggSite::getMembers` : Utilisez `elgg_get_entities()`
- `ElggSite::getObjects` : Utilisez `elgg_get_entities()`
- `ElggSite::listMembers` : Utilisez `elgg_list_entities()`
- `ElggSite::removeEntity`
- `ElggSite::removeObject`
- `ElggSite::removeUser`
- `ElggSite::isPublicPage` : Logique déplacée vers le routeur et qui ne devrait pas être consultée directement
- `ElggSite::checkWalledGarden` : Logique déplacée vers le routeur et qui ne devrait pas être consultée directement
- `ElggUser::countObjects` : Utilisez `elgg_get_entities()`
- `Logger::getClassName` : Utilisez `get_class()`
- `Elgg\Application\Database::getTablePrefix` : Lit la propriété `prefix`
- `elgg_view_access_collections()`
- `ElggSession::get_ignore_access` : Utilisez `getIgnoreAccess`
- `ElggSession::set_ignore_access` : Utilisez `setIgnoreAccess`
- `profile_pagesetup`
- `pages_can_delete_page` : Utilisez `$entity->canDelete()`
- `pages_search_pages`
- `pages_is_page` : utilisez `$entity instanceof ElggPage`
- `discussion_comment_override` : Voyez *Les réponses aux discussions ont été converties en commentaires*

- `discussion_can_edit_reply` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_reply_menu_setup` : Voir *Les réponses aux discussions ont été converties en commentaires*
- `discussion_reply_container_logic_override` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_reply_container_permissions_override` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_update_reply_access_ids` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_search_discussion` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_add_to_river_menu` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_prepare_reply_notification` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_redirect_to_reply` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `discussion_ecml_views_hook` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `search_get_where_sql`
- `search_get_ft_min_max`
- `search_get_order_by_sql`
- `search Consolidate_substrings`
- `search_remove_ignored_words`
- `search_get_highlighted_relevant_substrings`
- `search_highlight_words`
- `search_get_search_view`
- `search_custom_types_tags_hook`
- `search_tags_hook`
- `search_users_hook`
- `search_groups_hook`
- `search_objects_hook`
- `members_list_popular`
- `members_list_newest`
- `members_list_online`
- `members_list_alpha`
- `members_nav_popular`
- `members_nav_newest`
- `members_nav_online`
- `members_nav_alpha`
- `uservalidationbyemail_generate_code`

Toutes les fonctions autour de la table des sous-types d'entités :

- `add_subtype` : Utilisez `elgg_set_entity_class` lors de l'exécution
- `update_subtype` : Utilisez `elgg_set_entity_class` lors de l'exécution
- `remove_subtype`
- `get_subtype_id`
- `get_subtype_from_id`
- `get_subtype_class` : Utilisez `elgg_get_entity_class`
- `get_subtype_class_from_id`

Tous les caches ont été consolidés en une seule couche d'API. Les fonctions et méthodes suivantes ont été supprimées :

- `is_memcache_available`
- `_elgg_get_memcache`
- `_elgg_invalidate_memcache_for_entity`
- `ElggMemcache`
- `ElggFileCache`
- `ElggStaticVariableCache`
- `ElggSharedMemoryCache`

- L'interface Elgg\Cache\Pool et toutes les classes d'extension

À la suite des modifications apportées au journal du système :

- system_log_default_logger : déplacé dans le plugin system_log
- system_log_listener : déplacé dans le plugin system_log
- system_log : déplacé dans le plugin system_log
- get_system_log : renommé system_log_get_log et déplacé dans le plugin system_log
- get_log_entry : renommé system_log_get_log_entry et déplacé dans le plugin system_log
- get_object_from_log_entry : renommé system_log_get_object_from_log_entry et déplacé dans le plugin system_log
- archive_log : renommé system_log_archive_log et déplacé dans le plugin system_log
- logbrowser_user_hover_menu : renommé system_log_user_hover_menu et déplacé dans le plugin system_log
- logrotate_archive_cron : renommé system_log_archive_cron et déplacé dans le plugin system_log
- logrotate_delete_cron : renommé system_log_delete_cron et déplacé dans le plugin system_log
- logrotate_get_seconds_in_period : renommé system_log_get_seconds_in_period et déplacé dans le plugin system_log
- log_browser_delete_log : renommé system_log_browser_delete_log et déplacé dans le plugin system_log

APIs dépréciées

- ban_user : Utilisez ElggUser->ban()
- create_metadata : Utilisez la définition avec ElggEntity ou ElggEntity->setMetadata()
- update_metadata : Utilisez ElggMetadata->save()
- get_metadata_url
- create_annotation : Utilisez ElggEntity->annotate()
- update_metadata : Utilisez ElggAnnotation->save()
- elgg_get_user_validation_status : Utilisez ElggUser->isValidated()
- make_user_admin : Utilisez ElggUser->makeAdmin()
- remove_user_admin : Utilisez ElggUser->removeAdmin()
- unban_user : Utilisez ElggUser->unban()
- elgg_get_entities_from_attributes : Utilisez elgg_get_entities()
- elgg_get_entities_from_metadata : Utilisez elgg_get_entities()
- elgg_get_entities_from_relationship : Utilisez elgg_get_entities()
- elgg_get_entities_from_private_settings : Utilisez elgg_get_entities()
- elgg_get_entities_from_access_id : Utilisez elgg_get_entities()
- elgg_list_entities_from_metadata : Utilisez elgg_list_entities()
- elgg_list_entities_from_relationship : Utilisez elgg_list_entities()
- elgg_list_entities_from_private_settings : Utilisez elgg_list_entities()
- elgg_list_entities_from_access_id : Utilisez elgg_list_entities()
- elgg_list_registered_entities : Utilisez elgg_list_entities()
- elgg_batch_delete_callback
- \Elgg\Project\Paths::sanitize : Utilisez \Elgg\Project\Paths::sanitize()
- elgg_group_gatekeeper : Utilisez elgg_entity_gatekeeper()
- get_entity_dates : Utilisez elgg_get_entity_dates()
- messages_set_url : Utilisez ElggEntity::getURL()

Variables globales supprimées

- `$CURRENT_SYSTEM_VIEWTYPE`
- `$DEFAULT_FILE_STORE`
- `$ENTITY_CACHE`
- `$SESSION` : Utilisez l'API fournie par `elgg_get_session()`
- `$CONFIG->site_id` : Utilisez 1
- `$CONFIG->search_info`
- `$CONFIG->input` : Utilisez `set_input` et `get_input`

Classes/interfaces supprimées

- `FilePluginFile` : remplacez par `ElggFile` (ou chargez avec `get_entity()`)
- `Elgg_Notifications_Notification`
- `Elgg\Database\EntityTable\UserFetchResultException.php`
- `Elgg\Database\MetastringsTable`
- `Elgg\Database\SubtypeTable`
- `Exportable` et ses méthodes `export` et `getExportableValues` : Utilisez `toObject`
- `ExportException`
- `Importable` et sa méthode `import`.
- `ImportException`
- `ODD` et toutes les classes commençant par `ODD*`.
- `XmlElement`
- `Elgg_Notifications_Event` : Utilisez `\Elgg\Notifications\Event`
- `Elgg\Mail\Address` : utilisez `Elgg>Email\Address`
- `ElggDiscussionReply` : utilisez `ElggComment` voyez *Les réponses aux discussions ont été converties en commentaires*

Modifications de schéma

Le moteur de stockage des tables de la base de données a été changé de MyISAM à InnoDB. Vous devrez peut-être optimiser les paramètres de votre base de données pour cette modification. La table `datalists` a été supprimée. Tous les paramètres de la table `datalists` ont été fusionnés dans la table `config`.

Les chaînes de métadonnées (`metastrings`) de la base de données ont été dénormalisées à des fins de performance. Nous avons supprimé la table `metastrings` et mis toutes les valeurs de chaîne directement dans les tables de métadonnées et d'annotation. Vous devez mettre à jour vos requêtes personnalisées pour refléter ces modifications. Les alias de table `msv` et `msn` ne sont plus disponibles. Il est préférable de ne pas s'appuyer sur les alias de table utilisés dans les requêtes du noyau. Si vous avez besoin d'utiliser des clauses personnalisées, vous devriez faire vos propres jointures.

Dans la table `users_entity`, les colonnes `password` et `hash` ont été supprimées.

La table `geocode_cache` a été supprimée car elle n'était plus utilisée.

La colonne `subtype` dans la table `entities` ne contient plus d'ID de sous-type, mais une chaîne avec le sous-type. La table `entity_subtypes` a été supprimée.

les colonnes `type`, `subtype` et `access_id` de la table `river` ont été abandonnées. Pour les requêtes sans `elgg_get_river()` joignez la table `entities` sur `object_guid` pour vérifier le type et le sous-type de l'entité. La colonne d'accès n'est pas utilisée depuis un certain temps : les requêtes sont conçues pour garantir l'accès aux trois entités (subject, object et target).

Changements dans les fonctions de récupération `elgg_get_entities`, `elgg_get_metadata` et `elgg_get_annotations`

`elgg_get_entities` accepte désormais toutes les options qui étaient auparavant réparties entre `elgg_get_entities_from_metadata`, `elgg_get_entities_from_annotations`, `elgg_get_entities_from_relationship`, `elgg_get_entities_from_private_settings` et `elgg_get_entities_from_access_id`. Ces dernières ont été dépréciées.

La transmission d'instructions MySQL brutes aux options est déconseillée. Il est conseillé d'utiliser les Closures qui reçoivent une instance de `\Elgg\Database\QueryBuilder` et de préparer l'instruction à l'aide de la couche d'abstraction de base de données. D'une part, cela permettra de s'assurer que toutes les instructions sont correctement aseptisées à l'aide du pilote de base de données, d'autre part, il nous permettra de passer à la construction de requête orientée objet testable.

Les instructions `wheres` ne doivent pas utiliser de chaînes SQL brutes, au lieu de passer une instance de `\Elgg\Database\Clauses\WhereClause` ou une Closure qui renvoie une instance de `\Doctrine\DBAL\Query\Expression\CompositeExpression` :

```
elgg_get_entities([
    'wheres' => [
        function(\Elgg\Database\QueryBuilder $qb, $alias) {
            $joined_alias = $qb->joinMetadataTable($alias, 'guid', 'status');
            return $qb->compare("$joined_alias.name", 'in', ['draft', 'unsaved_draft'],
↵ELGG_VALUE_STRING);
        }
    ]
]);
```

Les clauses `joins`, `order_by`, `group_by`, `selects` ne doivent pas utiliser de chaînes SQL brutes. Utilisez les Closures qui reçoivent une instance de `\Elgg\Database\QueryBuilder` et renvoyez une instruction (statement) préparée.

L'option `reverse_order_by` a été supprimée.

Les plugins ne devraient pas s'appuyer sur des alias de table joints et sélectionnés. Les Closures transmises au tableau d'options recevront un deuxième argument correspondant à l'alias de table sélectionné. Les plugins doivent effectuer leurs propres jointures et utiliser des alias de jointure en conséquence.

Notez que toute l'API privée autour de la construction de chaînes SQL brutes a également été supprimée. Si vous l'utilisez dans vos plugins, soyez informé que tout ce qui est marqué comme `@access private` ou `@internal` dans le noyau peut être modifié et supprimé à tout moment, et que nous ne garantissons aucune compatibilité arrière pour ces fonctions. NE LES UTILISEZ PAS. Si vous avez besoin de les utiliser, ouvrez un ticket sur Github et nous envisagerons d'ajouter un équivalent public.

Propriétés de l'entité booléenne

Le stockage des métadonnées, des annotations et des valeurs de paramètres privés a été aligné.

Les valeurs booléennes sont transformées en entiers lorsqu'elles sont enregistrées : `false` est stocké comme `0` et `true` est stocké comme `1`. Cela a des implications de rupture pour les paramètres privés, qui étaient précédemment stockés comme des chaînes vides pour les valeurs `false`. Les plugins doivent écrire leurs propres scripts de migration pour modifier les valeurs de la DB des chaînes vides à `0` (pour les paramètres privés qui sont censés stocker des valeurs booléennes) pour s'assurer que `elgg_get_entities()` peut récupérer ces valeurs avec `private_setting_name_value_pairs` contenant des valeurs `false`. Cela s'applique aux paramètres de plugin, ainsi qu'à tous les paramètres privés ajoutés aux entités.

Modifications des métadonnées

Les métadonnées ne sont plus contrôlées par accès. Si votre plugin a créé des métadonnées avec un accès restreint, ces restrictions ne seront pas respectées. Vous devez utiliser des annotations ou des entités à la place, qui fournissent un contrôle d'accès.

Ne lisez pas ou n'écrivez pas la propriété `access_id` sur les objets `ElggMetadata`.

Les métadonnées ne sont plus activées ou désactivées. Vous ne pouvez plus effectuer les appels API `enable` et `disable` sur les métadonnées.

Les métadonnées n'ont plus de `owner_guid`. Il n'est plus possible d'interroger les métadonnées sur la base de leur `owner_guids`. Par conséquent, `ElggMetadata::canEdit()` renvoie toujours `true` quel que soit l'utilisateur connecté, à moins que cela soit explicitement remplacé par un hook de plugin.

Permissions et droits d'accès

Le service des fonctionnalités utilisateur ne déclenche plus les hooks de vérification des autorisations lorsque :

- les autorisations sont vérifiées pour un utilisateur administrateur
- les autorisations sont vérifiées lorsque l'accès est ignoré avec `elgg_set_ignore_access()`

Cela signifie que les plugins ne peuvent plus modifier les autorisations dans les cas mentionnés ci-dessus.

`elgg_check_access_overrides()` a été supprimé, car les plugins n'auront plus besoin de valider les remplacements d'accès.

Les traductions des niveaux d'accès Elgg par défaut ont de nouvelles clefs de traduction.

Modifications multisites

Elgg avant la version 3.0 dispose d'une certaine prise en charge (partielle) pour avoir plusieurs sites dans la même base de données. Ce concept multisite a été complètement supprimé dans la version 3.0. Les entités n'ont plus l'attribut `site_guid`. Cela signifie qu'il n'y a plus la possibilité d'avoir des entités sur différents sites. Si vous avez actuellement plusieurs sites dans votre base de données, la mise à niveau d'Elgg vers 3.0 échouera. Vous devez séparer les différents sites en bases de données/tables distinctes.

En ce qui concerne la suppression du concept multisite de Elgg, il n'est plus nécessaire que les entités aient une relation "member_of_site" avec l'entité du site. Toutes les fonctions liées à l'ajout/suppression de cette relation ont été supprimées. Toutes les relations existantes seront supprimées dans le cadre de cette mise à niveau.

La définition de `ElggSite::$url` n'a aucun effet. La lecture de l'URL du site est toujours faite depuis la valeur définie par `$CONFIG->wwroot` dans `settings.php`, ou calculée par `Symphony Request`.

`ElggSite::save()` échouera si ce n'est pas le site principal.

Modifications des sous-tables des entités

La sous-table `sites_entity` pour `ElggSite` n'existe plus. Tous les attributs ont été déplacés vers les métadonnées. La sous-table `groups_entity` pour `ElggGroup` n'existe plus. Tous les attributs ont été déplacés vers les métadonnées. La sous-table `objects_entity` pour `ElggObject` n'existe plus. Tous les attributs ont été déplacés vers les métadonnées. La sous-table `users_entity` pour `ElggUser` n'existe plus. Tous les attributs ont été déplacés vers les métadonnées.

Si vous avez des requêtes personnalisées faisant référence à cette table, vous devez les mettre à jour. Si vous avez une fonction qui s'appuie sur `Entity->getOriginalAttributes()`, sachez que cela ne renvoie que les attributs de base d'une `ElggEntity` et ne contiendra plus les attributs secondaires.

Collection d'accès des Contacts et des Groupes

La table des collections d'accès a maintenant une colonne de sous-type. Ces données supplémentaires permettent d'identifier le but de l'ACL. Les collections d'accès appartenant à l'utilisateur sont supposées être utilisées comme collections de contacts et ont maintenant le sous-type `friends_collection`. Les informations d'accès aux groupes collectées étaient auparavant stockées dans la métadonnée `group_acl`. Avec l'introduction du sous-type ACL, ces informations ont été déplacées vers l'attribut de sous-type ACL.

L'`access_id` `ACCESS_FRIENDS` a été migré vers une collection d'accès réelle (avec le sous-type `friends`). Toutes les entités et annotations ont été mises à jour pour utiliser le nouvel id de collection d'accès. La collection d'accès est créée lors de la création d'un utilisateur. Lorsqu'une relation du type `friends` est créée, le guid lié sera également ajouté à la collection d'accès. Vous ne pouvez plus enregistrer ou mettre à jour les entités avec l'id d'accès `ACCESS_FRIENDS`.

Les sous-types n'ont plus d'ID

Les sous-types d'entités ont été dénormalisés. La table `entity_subtypes` a été supprimée et la colonne `subtype` dans la table entités contient simplement la représentation sous forme de chaîne du sous-type.

Par conséquent, toutes les API relatives à l'ajout/mise à jour des sous-types et des classes d'entités ont été supprimées.

Les plugins peuvent désormais utiliser `elgg_set_entity_class()` et `elgg_get_entity_class()` pour enregistrer une classe d'entité personnalisée au moment de l'exécution (par exemple dans le gestionnaire d'initialisation du système).

Toutes les entités **DOIVENT** maintenant avoir un sous-type. Par défaut, les sous-types suivants sont ajoutés et réservés :

- `user` pour les utilisateurs
- `group` pour les groupes
- `site` pour les sites

Chargement de classe personnalisée

Elgg ne fournit plus de fonctions API pour enregistrer des classes personnalisées. Si vous avez besoin de classes personnalisées, vous pouvez utiliser des classes PSR-0 dans le dossier `/classes` de votre plugin ou utiliser composer pour charger automatiquement des classes supplémentaires.

Les fonctions suivantes liées à l'enregistrement des classes ont été supprimées :

- `elgg_get_class_loader`
- `elgg_register_class`
- `elgg_register_classes`

Conteneur d'injection de dépendance (DI = Dependency Injection)

Les plugins peuvent maintenant définir leurs services et les attacher au conteneur public DI de Elgg en fournissant des définitions dans `elgg-services.php` dans la racine du répertoire de plugin.

`elgg()` ne renvoie plus une instance d'application Elgg, mais une instance de conteneur DI.

Modifications de la recherche

Nous avons ajouté un service de recherche dans le noyau, par conséquent, le plugin `search` fournit désormais uniquement une interface utilisateur pour afficher les formulaires et afficher les résultats de recherche. Bon nombre des vues dans le plugin de recherche ont été affectées par ce changement.

Les indices FULLTEXT ont été supprimés sur différentes tables. Le plugin de recherche utilise désormais toujours une requête LIKE lors de l'exécution d'une recherche.

Voyez la documentation du *Service de recherche* et des *Hooks de recherche* pour des informations détaillées sur les nouvelles fonctionnalités de recherche.

Modifications liées aux formulaires et aux champs

- `input/password` : par défaut, ce champ n'affichera plus la valeur qui lui est transmise ; cela peut être remplacé en passant à la vue la variable `always_empty` définie sur `false`
- `input/submit`, `input/reinit` et `input/button` sont maintenant rendus avec un `<button>` au lieu de la balise `<input>`. Ces vues d'entrée acceptent également les paramètres `text` et `icon`.
- `output/url` définit maintenant la classe `.elgg-anchor` sur les éléments d'ancre et accepte le paramètre `icon`. Si aucun `text` n'est défini, le paramètre `href` est utilisé comme étiquette et sera limité à 100 caractères.
- `output/url` prend désormais en charge un paramètre `badge`, qui peut être utilisé lorsqu'un compteur, un badge ou un autre est requis comme suffixe (principalement dans les éléments de menu qui ont des compteurs).
- `output/tags` n'utilise plus de balises `` avec des floats, et au lieu de cela s'appuie sur des éléments intrinsèquement en ligne `inline` tels que `` et `<a>`

Modifications du menu des entités et de la rivière

Le menu Entity et River affiche désormais tous les éléments d'une liste déroulante. Les actions sociales comme aimer ou commenter sont déplacées vers un menu alternatif appelé le menu social, qui est destiné aux actions sociales.

Bibliothèques supprimées

`elgg_register_library` et `elgg_load_library` ont été supprimées. Ces fonctions ont eu peu d'impact sur les performances (en particulier avec OPCache activé), et il a été difficile pour d'autres plugins de travailler avec les API contenues dans les bibliothèques. En outre, il était difficile pour les développeurs de savoir que les API étaient contenues dans une bibliothèque alors qu'il y avait l'autocomplétion par l'IDE.

Si vous êtes concerné par les performances, déplacez la logique vers les classes et laissez PHP les charger automatiquement au besoin, sinon utilisez `require_once` et exigez vos bibliothèques.

Gestionnaires de pages supprimés

- `file/download`
- `file/search`
- `groupicon`
- `twitterservice`
- `collections/pickercallback`
- `discussion/reply` : Voyez *Les réponses aux discussions ont été converties en commentaires*
- `expages`
- `invitefriends` : Utilisez `friends/{username}/invite`
- `messages/compose` : Utilisez `messages/add`

- reportedcontent

Actions supprimées

- file/download : Utilisez `elgg_get_inline_url` ou `elgg_get_download_url`
- file/delete : Utilisez l'action `entity/delete`
- import/opendd
- discussion/reply/save : Voyez *Les réponses aux discussions ont été converties en commentaires*
- discussion/reply/delete : Voyez *Les réponses aux discussions ont été converties en commentaires*
- comment/delete : Utilisez l'action `entity/delete`
- uservalidationbyemail/bulk_action : utilisez `admin/user/bulk/validate` ou `admin/user/bulk/delete`
- uservalidationbyemail/delete : utilisez `admin/user/bulk/delete`
- uservalidationbyemail/validate : utilisez `admin/user/bulk/validate`
- invitefriends/invite : utilisez `friends/invite`

Modifications de l'héritage

- ElggData (et donc la plupart des objets de domaine Elgg) n'implémente plus `Exportable`
- ElggEntity n'implémente plus `Importable`
- ElggGroup n'implémente plus `Frienable`
- ElggRelationship n'implémente plus `Importable`
- ElggSession n'implémente plus `ArrayAccess`
- Elgg\Application\Database n'étend plus Elgg\Database

APIs JavaScript supprimées

- `admin.js`
- `elgg.widgets` : Utilisez le module `elgg/widgets`. Les dispositions « widgets » chargent ce module automatiquement
- `lightbox.js` : Utilisez le module `elgg/lightbox` au besoin
- `lightbox/settings.js` : Utilisez le hook JS `getOptions`, `ui.lightbox` ou l'attribut `data-colorbox-opts`
- `elgg.ui.popupClose` : Utilisez le module `elgg/popup`
- `elgg.ui.popupOpen` : Utilisez le module `elgg/popup`
- `elgg.ui.initAccessInputs`
- `elgg.ui.river`
- `elgg.ui.initDatePicker` : Utilisez le module `input/date`
- `elgg.ui.likesPopupHandler`
- `elgg.embed` : Utilisez le module `elgg/embed`
- `elgg.discussion` : Utilisez le module `elgg/discussion`
- `embed/custom_insert_js` : Utilisez le hook JS `embed`, `editor`
- `elgg/ckeditor.js` : remplacé par `elgg-ckeditor.js`
- `elgg/ckeditor/set-basepath.js`
- `elgg/ckeditor/insert.js`
- `jQuery.cookie` : Utilisez `elgg.session.cookie`
- `jQuery.jeditable`
- `likes.js` : Le module `elgg/likes` est chargé automatiquement
- `messageboard.js`
- `elgg.autocomplete` n'est plus défini.
- `elgg.messageboard` n'est plus défini.

- `jQuery.fn.friendsPicker`
- `elgg.ui.toggleMenu` n'est plus défini
- `elgg.ui.toggleMenuItems` : Utilisez l'attribut `data-toggle` lors de l'enregistrement d'éléments de menu basculables
- `uservalidationbyemail/js.php` : Utilisez le module `elgg/uservalidationbyemail`
- `discussion.js` : Voyez *Les réponses aux discussions ont été converties en commentaires*

Hooks/événements retirés

- Événement **login, user** : Utilisez **login :before** ou **login :after**. Notez que l'utilisateur n'est pas connecté pendant l'événement **login :before**
- Événement **delete, annotations** : Utilisez **delete, annotation**
- Événement **pagesetup, system** : Utilisez plutôt les hook de menu ou de coquille de page
- Événement **upgrade, upgrade** : Utilisez **upgrade, system** à la place
- Hook **index, system** : Remplacez la vue `resources/index`
- Hook **object :notifications, <type>** : Utilisez le hook **send :before, notifications**
- Hook **output :before, layout** : Utilisez **view_vars, page/layout/<layout_name>**
- Hook **output :after, layout** : Utilisez **view, page/layout/<layout_name>**
- Hook **email, system** : Utilisez les hooks plus granulaires **<hook>, system :email**
- Hook **email :message, system** : Utilisez le hook **zend :message, system :email**
- Hook **members :list, <page>** : Utilisez votre propre gestionnaire de page ou hook de route
- Hook **members :config, <page>** : Utilisez **register, menu :filter :members**
- Hook **profile_buttons, group** : Utilisez **register, menu :title**

Formulaires/actions supprimés

- formulaire et action `notificationsettings/save`
- formulaire et action `notificationssettings/groupsave`
- formulaire et action `discussion/reply/save`

APIs qui n'acceptent plus qu'un tableau \$options

- `ElggEntity::getAnnotations`
- `ElggEntity::getEntitiesFromRelationship`
- `ElggEntity::getEntitiesFromRelationship`
- `ElggUser::getGroups`
- `ElggUser::getFriends` (dans le cadre de `Friendable`)
- `ElggUser::getFriendsOf` (dans le cadre de `Friendable`)
- `ElggUser::getFriendsObjects` (dans le cadre de `Friendable`)
- `ElggUser::getObjects` (dans le cadre de `Friendable`)
- `find_active_users`
- `elgg_get_admin_notices`

Fonctions plugin qui nécessitent maintenant un \$plugin_id explicite

- `elgg_get_all_plugin_user_settings`
- `elgg_set_plugin_user_setting`
- `elgg_unset_plugin_user_setting`
- `elgg_get_plugin_user_setting`
- `elgg_set_plugin_setting`
- `elgg_get_plugin_setting`
- `elgg_unset_plugin_setting`
- `elgg_unset_all_plugin_settings`

Constructeurs de classes qui n'acceptent désormais qu'un objet stdClass ou null

- `ElggAnnotation` : N'accepte plus un ID d'annotation
- `ElggGroup` : N'accepte plus un GUID
- `ElggMetadata` : N'accepte plus un ID de métadonnées
- `ElggObject` : N'accepte plus un GUID
- `ElggRelationship` : N'accepte plus un ID de relation ni null
- `ElggSite` : N'accepte plus un GUID ni une URL
- `ElggUser` : N'accepte plus un GUID ni un nom d'utilisateur
- `ElggPlugin` : N'accepte plus un GUID ni un chemin. Utilisez `ElggPlugin::fromId` pour construire un plugin à partir de son chemin

Modifications d'API diverses

- `ElggBatch` : Vous ne pouvez accéder qu'aux propriétés publiques
- `ElggEntity` : Les propriétés `tables_split` et `tables_loaded` ont été supprimées
- `ElggEntity` : Les URL vides ne seront plus normalisées. Cela signifie que les entités sans URL ne vont plus résulter en l'URL du site
- `ElggGroup::removeObjectFromGroup` nécessite de passer un `ElggObject` (n'accepte plus un GUID)
- `ElggUser::$salt` n'existe plus en tant qu'attribut, et n'est plus utilisé pour l'authentification
- `ElggUser::$password` n'existe plus en tant qu'attribut, et n'est plus utilisé pour l'authentification
- `elgg_get_widget_types` ne prend plus en charge `$exact` comme deuxième argument
- `elgg_instanceof` ne prend plus en charge le quatrième argument `class`
- `elgg_view` : Les 3ème et 4ème arguments (inutilisés) ont été supprimés. Si vous utilisez l'argument `$viewtype`, vous devez mettre à jour votre utilisation.
- `elgg_view_icon` ne prend plus en charge `true` comme 2e argument
- `elgg_list_entities` ne prend plus en charge l'option `view_type_toggle`
- `elgg_list_registered_entities` ne prend plus en charge l'option `view_type_toggle`
- `elgg_log` n'accepte plus le niveau "DEBUG"
- `elgg_dump` n'accepte plus d'argument `$to_screen`.
- `elgg_gatekeeper` et `elgg_admin_gatekeeper` ne signalent plus login ou admin comme motif de redirection, mais 403
- `Application::getDb()` ne renvoie plus une instance de `Elgg\Database`, mais plutôt `Elgg\Application\Database`
- `$CONFIG` n'est plus disponible en tant que variable locale à l'intérieur des fichiers `start.php` des plugins.
- `elgg_get_config('siteemail')` n'est plus disponible. Utilisez `elgg_get_site_entity()->email`.
- `ElggEntity::saveIconFromUploadedFile` enregistre uniquement la taille *master*, les autres tailles sont créées sur demande par `ElggEntity::getIcon()` à partir de la taille *master*
- `ElggEntity::saveIconFromLocalFile` enregistre uniquement la taille *master*, les autres tailles sont créées sur demande par `ElggEntity::getIcon()` à partir de la taille *master*

- `ElggEntity::saveIconFromElggFile` enregistre uniquement la taille *master*, les autres tailles sont créées sur demande par `ElggEntity::getIcon()` à partir de la taille *master*.
- Les entités de groupe n'ont plus l'attribut magique `username`.
- Le gestionnaire de page ne détecte plus `group:<guid>` dans l'URL.
- L'intervalle CRON `reboot` est supprimé.
- Les points de terminaison d'URL `js/` et `css/` ne sont plus pris en charge. Utilisez `elgg_get_simplecache_url()`.
- L'action d'enregistrement de commentaire générique n'envoie plus la notification directement, cela a été déchargé vers le système de notification.
- Le script `engine/start.php` est supprimé.
- Les fonctions `set_config`, `unset_config` et `get_config` ont été dépréciées et remplacées par `elgg_set_config`, `elgg_remove_config` et `elgg_get_config`.
- Les valeurs de configuration `path`, `wwwroot` et `dataroot` ne sont pas lues dans la base de données. Les valeurs de fichier `settings.php` sont toujours utilisées.
- Les fonctions de configuration comme `elgg_get_config` n'émondent (`trim`) plus les clefs.
- Si vous remplacez l'affichage `navigation/menu/user_hover/placeholder`, vous devez modifier la clef de configuration `lazy_hover:menus` en `elgg_lazy_hover_menus`.
- La valeur de configuration `entity_types` n'est plus présente ni utilisée.
- Les images téléchargées sont auto-pivotées en fonction de leurs métadonnées d'orientation.
- La vue `object/widget/edit/num_display` utilise désormais un champ `input/number` au lieu de `input/select`; vous devrez peut-être mettre à jour vos vues de configuration des widgets en conséquence.
- Les noms d'annotation ne sont plus émondés (`trim`) lors de l'enregistrement.

Le comportement d'extension des vues a été modifié

Une vue étendue recevra désormais tous les hooks classiques (comme le crochet `view_vars`). Il est désormais également possible d'étendre les extensions de vue. Avec ce changement de comportement, tous les rendus de vue se comporteront de la même façon. Cela n'a plus d'importance si elle est utilisée comme une extension ou non.

L'ordre d'appel des hooks JavaScript pourrait changer

Lors de l'inscription pour les hooks, le mot clef `all` pour la correspondance générique n'a plus aucun effet sur l'ordre d'appel des gestionnaires. Pour vous assurer que votre gestionnaire est appelé en dernier, vous devez lui donner la plus haute priorité de tous les gestionnaires correspondants, ou pour vous assurer que votre gestionnaire est appelé en premier, vous devez lui donner la plus faible priorité de tous les gestionnaires correspondants.

Si les gestionnaires ont été enregistrés avec la même priorité, ceux-ci sont appelés dans l'ordre où ils ont été enregistrés.

Pour émuler le comportement antérieur, les gestionnaires du noyau Elgg enregistrés avec le mot clé `all` ont été relevés en priorité. Certains de ces gestionnaires seront très probablement appelés dans un ordre différent.

Modifications liées à la disposition widget

L'utilisation de la disposition widget a été modifiée. Le contenu n'est plus dessiné dans le cadre de la mise en page. Vous devez envelopper votre contenu dans une autre mise en page et utiliser la disposition widget comme contenu. Si vous souhaitez que du contenu spécial s'affiche s'il n'y a pas de widgets dans la mise en page, vous pouvez maintenant passer un paramètre spécial `no_widgets` (sous forme de chaîne ou de Closure).

Lors de l'enregistrement des widgets, vous ne pouvez plus omettre de passer un contexte car le contexte `all` n'est plus pris en charge. Vous devez explicitement passer les contextes pour lesquels le widget est destiné.

Routage

La gestion des pages à l'aide de `elgg_register_page_handler()` est devenue obsolète.

Nous avons ajouté une nouvelle API de routage à l'aide de `elgg_register_route()`, qui permet aux plugins de définir des routes nommées, en utilisant par la suite des noms de routes pour générer des URLs à l'aide de `elgg_generate_url()`.

Pour plus d'informations, consultez les documentations sur le [routage](#).

À la suite de cette modification, tous les gestionnaires de pages de base ont été supprimés et toute la logique contenue dans ces gestionnaires de pages a été déplacée vers les vues de ressources respectives.

`elgg_generate_entity_url()` a été ajouté comme raccourci pour générer des URLs à partir de routes nommées qui dépendent du type d'entité et du sous-type.

L'utilisation du paramètre `handler` dans les menus des entités a été dépréciée au profit des routes d'entités nommées.

Les fonctions Gatekeeper ont été refactorisées pour servir de middleware dans le processus de routage, et en tant que tel, elles ne retournent plus de valeurs. Ces fonctions lancent des exceptions HTTP qui sont ensuite acheminées vers des pages d'erreur et peuvent être redirigées vers d'autres pages via des hooks.

Étiquetage

Les conventions de nommage des entités et des collections ont changé pour se conformer aux nouveaux schémas de routage :

```
return [
    'item:object:blog' => 'Blog',
    'collection:object:blog' => 'Blogs',
    'collection:object:blog:all' => 'All site blogs',
    'collection:object:blog:owner' => '%s\'s blogs',
    'collection:object:blog:group' => 'Group blogs',
    'collection:object:blog:friends' => 'Friends\' blogs',
    'add:object:blog' => 'Add blog post',
    'edit:object:blog' => 'Edit blog post',
];
```

Ces conventions sont utilisées dans les systèmes de routage et de navigation, de sorte que les plugins sont invités à les suivre.

Filtrage des valeurs de la requête

`set_input()` et `get_input()` n'émondent (trim) plus les valeurs.

Réponses des actions

Toutes les actions du noyau et des plugins du noyau utilisent maintenant toutes les nouvelles fonctions de Http Response comme *elgg_ok_response* et *elgg_error_response* au lieu de *forward()*. L'effet de ce changement est que dans la plupart des cas le hook “*forward*”, “*system*” n’est plus déclenché. Si vous souhaitez modifier les réponses, vous pouvez maintenant utiliser le hook “*response*”, “*action* :<name/of/action>”. Cela vous donne plus de contrôle sur la réponse et permet de cibler une action spécifique très facilement.

Htmlawed n'est plus un plugin

- N'appellez pas `elgg_load_library('htmlawed')`.
- Dans les paramètres du hook pour 'config', 'htmlawed', le nom de la fonction `hook_tag` a changé.

Nouvelle approche des mises en page (layouts)

Les dispositions `one_column`, `one_sidebar`, `two_sidebar` et `content` ont été supprimées - au lieu de cela, le rendu de la disposition a été centralisé dans `default`. La mise à jour de la disposition `default` permet de contrôler pleinement les éléments de mise en page via `$vars`. Pour obtenir une compatibilité descendante maximale, les appels à `elgg_view_layout()` avec ces noms de disposition donneront toujours la sortie attendue, mais les plugins doivent commencer à utiliser la disposition `default` avec un ensemble de paramètres mis à jour.

Les mises en page ont été décomposées en éléments plus petits, ce qui devrait faciliter le ciblage d'éléments de mise en page spécifiques par les thèmes sans avoir à remplacer les dispositions en général.

À la suite de ces changements :

- toutes les dispositions sont cohérentes dans la façon dont elles gèrent les menus de titre et de filtre, le fil d'Ariane et les sous-vues de mise en page
- toutes les dispositions peuvent maintenant être facilement étendues pour avoir plusieurs onglets. Les plugins peuvent passer le paramètre `filter_id` qui permettra à d'autres plugins de s'accrocher au hook `register`, `menu:filter:<filter_id>` et d'ajouter de nouveaux onglets. Si aucun `filter_id` n'est fourni, le hook par défaut `register`, `menu:filter` peut être utilisé.
- les vues de mise en page et les sous-vues reçoivent désormais les `identifier` et `segments` de la page en cours de rendu
- les paramètres de mise en page sont disponibles pour les hooks de menu de titre et de filtre, ce qui permet aux ressources de fournir des informations de contexte supplémentaires, par exemple, un `$entity` en cas de ressource pour le profil

Plugins et thèmes devraient :

- Mettez à jour les appels à `elgg_view_layout()` pour utiliser la disposition `default`
- Mettez à jour et remplacez le paramètre `nav` dans les vues de disposition par le paramètre `breadcrumbs`
- Mettez à jour leur utilisation du paramètre `filter` dans les vues de disposition (layouts), soit en fournissant un ensemble par défaut d'onglets de filtre, soit en définissant un paramètre `filter_id` et en utilisant des hooks
- Supprimez la vue `page/layouts/one_column`
- Supprimez la vue `page/layouts/one_sidebar`
- Supprimez la vue `page/layouts/two_sidebar`
- Supprimez la vue `page/layouts/content`
- Mettez à jour leur utilisation de `page/layouts/default`
- Mettez à jour leur utilisation de `page/layouts/error`
- Mettez à jour leur utilisation de `page/layouts/elements/filter`
- Mettez à jour leur utilisation de `page/layouts/elements/header`
- Mettez à jour leur utilisation de `page/layouts/elements/footer`
- Mettez à jour leur utilisation de `page/elements/title`
- Mettez à jour leur utilisation de la `navigation/breadcrumb` pour passer `$vars ['breadcrumb']` à `elgg_get_breadcrumbs()`

- Mettez à jour les enregistrements de hook pour `output:before`, layout en `view_vars`, `page/layout/<layout_name>`
- Mettez à jour les enregistrements de hooks pour `output:after`, layout en `view`, `page/layout/<layout_name>`

Plugin Likes

La fonctionnalité J'aime n'utiliser plus l'API de bascule d'Elgg, de sorte qu'un seul élément de menu `likes` est utilisé. Les actions `add/remove` ne renvoient plus directement les valeurs Ajax, car les données d'état des mentions J'aime sont maintenant renvoyées avec *chaque* requête Ajax qui envoie un « guid ». Lorsque le nombre de mentions J'aime est nul, l'élément de menu `likes_count` est maintenant masqué en ajoutant `.hidden` à l'élément LI, au lieu de le faire sur l'ancre. De plus, l'élément de menu `likes_count` est un lien classique, et n'est plus créé par la vue `likes/count`.

Plugin Notifications

Le plugin Notifications a été réécrit en abandonnant de nombreuses vues et actions. Le but de cette réécriture était de mettre en œuvre une interface plus efficace, extensible et évolutive pour la gestion des préférences de notifications. Nous avons mis en œuvre un balisage beaucoup plus simple, supprimé les styles en excès et le javascript qui était nécessaire pour faire fonctionner l'ancienne interface.

Si votre plugin étend l'une des vues ou s'appuie sur des actions du plugin notifications, il doit être mis à jour.

Plugin Pages

Le sous-type `page_top` a été migré dans le sous-type `page`. Le sous-type `page` dispose de sa propre classe, à savoir `ElggPage`. Pour vérifier si une `ElggPage` est une page de premier niveau, la fonction de classe `ElggPage->isTopPage()` a été ajoutée.

Toutes les pages ont une valeur de métadonnées pour `parent_guid`, pour les pages de premier niveau sa valeur est `0`.

Plugin Profile

Toutes les fonctionnalités liées au profil ont été déplacées hors du noyau dans ce plugin. Les plus remarquables sont l'utilitaire d'administration des champs de profil et le hook pour définir les données de configuration des champs de profil.

Plugin Data Views (vues de données)

Le plugin Data Views n'est plus livré avec le noyau.

Plugin Twitter API

Le plugin `twitter_api` a été retiré du noyau d'Elgg. Le plugin est toujours disponible sous forme de package Composer `elgg/twitter_api`. Pour l'installer, ajoutez ce qui suit à la section `require` de votre `composer.json` :

```
{
    "require": {
        "elgg/twitter_api": "~1.9"
    }
}
```

Plugin des URLs d'origine

Le plugin `legacy_urls` a été retiré du noyau Elgg. Le plugin est toujours disponible sous la forme du package Composer `elgg/legacy_urls`. Pour l'installer, ajoutez ce qui suit à la section `require` de votre `composer.json` :

```
{
    "require": {
        "elgg/legacy_urls": "~2.3"
    }
}
```

Plugin de validation de l'utilisateur par email

La vue qui liste les utilisateurs non validés a été déplacée du plugin vers le noyau Elgg. Certaines actions génériques (par exemple valider et supprimer) ont également été déplacées vers le noyau Elgg.

Livraison d'e-mail

Pour prévoir plus de granularité dans la gestion et la livraison des courriels, le hook **email, system** a été supprimé. Le nouveau service de messagerie fournit plusieurs autres hooks de remplacement qui permettent aux plugins de contrôler le contenu, le format et l'agent de transport de messagerie utilisés pour la livraison.

`elgg_set_email_transport()` peut désormais être utilisé pour remplacer l'agent de transport Sendmail par défaut par une autre instance de `\Zend\Mail\Transport\TransportInterface`, par exemple SMTP, in-memory, ou le transport de fichiers. Notez que cette fonction doit être appelée tôt dans le processus de démarrage. Notez que si vous appelez cette fonction sur chaque requête, l'utilisation des paramètres du plugin pour déterminer la configuration de transport peut ne pas être très efficace - stockez ces paramètres dans la liste de données (datalist) ou dans les valeurs de configuration du site, de sorte qu'ils soient chargés à partir du cache de démarrage.

Changements de thème et de style

Le thème Aalborg n'est plus livré avec Elgg. Le thème du noyau par défaut est maintenant basé sur Aalborg, mais avec des changements majeurs.

Évolutions notables dans les plugins :

- La barre supérieure, la barre de navigation et l'entête ont été combinés en un seul composant de barre supérieure réactive (responsive)
- La largeur intérieure par défaut est maintenant 1280px (80rem * 16px/1rem)
- L'unité de mesure préférée est maintenant *rem* et non *px*

- Le thème utilise un *système de grille à 8 points* <<https://builttoadapt.io/intro-to-the-8-point-grid-system-d2573cde8632>>
- Les menus, éléments de mise en page (layouts) et autres composants utilisent maintenant flexbox
- La réinitialisation se fait à l'aide d'un *système de grille à 8 points* <<https://necolas.github.io/normalize.css/>>
- Les requêtes multimédias ont été réécrites pour une expérience mobile-first
- Les éléments de formulaire (entrées de texte, boutons et sélecteurs) ont maintenant une hauteur égale de 2.5rem
- L'entête de la disposition (layout) est maintenant positionné à l'extérieur des colonnes de la disposition, qui ont été enveloppées dans `elgg-layout-columns`
- les propriétés `z-index` ont été examinés et empilés avec une itération simple au lieu de 9999999 <<https://hackernoon.com/my-approach-to-using-z-index-eca67feb079c>>.
- Le schéma de couleurs a été modifié pour mettre en évidence les éléments exploitables et réduire l'abondance des nuances de gris
- le plugin de recherche n'étend plus `page/elements/header`, et au lieu de cela `page/elements/topbar` affiche la vue `search/search_box`
- `.elgg-icon` n'a plus de `font-size`, `line-height` ou `color` global : ces valeurs seront héritées des éléments parent
- Le support de `.elgg-icon-hover` a été abandonné
- Les icônes « hover » de l'utilisateur ne sont plus recouvertes par une icône « caret »

Pour en savoir plus au sujet *Principes pour les thèmes*

Notez également que les vues CSS servies via les URL `/cache` sont pré-traitées à l'aide de *CSS Crush* <<http://the-echoplex.net/csscrush/>>. Si vous faites des références à des variables CSS ou à d'autres éléments, la définition doit être située dans la même vue d'affichage. Par exemple, une variable définie dans `elgg.css` ne peut pas être référencée dans un fichier CSS distinct comme `colorbox.css`.

Commentaires

L'envoi des commentaires est maintenant AJAXifié. Après un envoi réussi, la liste des commentaires sera mise à jour automatiquement.

Les modifications suivantes ont été apportées aux notifications des commentaires.

- Les clefs de traduction relatives aux notifications des commentaires ont changé. Vérifiez les clefs de traduction `generic_comment:notification:owner`:
- L'action de création d'un commentaire (`action/comment/save`) a été modifiée. Si votre plugin a remplacé cette action, vous devriez jeter un oeil dessus afin d'éviter les doubles notifications

Vues de listing d'objets

- `object/elements/full/body` enveloppe maintenant l'ensemble du corps du listing dans un emballage (wrapper) `.elgg-listing-full-body`
- `object/elements/full` supporte désormais `attachments` et `responses` qui sont affichés après le corps du listing
- Dans les plugins du noyau, les vues de ressources n'affichent plus `comments/replies` - au lieu de cela, elles transmettent un indicateur `show_responses` à la vue de l'entité, ce qui génère les réponses et les transmet à la vue de la liste complète. Les plugins tiers devront mettre à jour leurs utilisations des vues `object/<subtype>` et `ressources/<handler>/view`.
- La vue de la discussion complète est désormais affichée en utilisant la vue `object/elements/full`
- `object/file` passe désormais la vue image (`specialcontent`) sous la forme d'une pièce jointe `attachment` à la vue listing

Modifications des menus

Le tri par défaut des éléments de menu a été modifié de `text` à `priority`.

Notez que les hooks `register` et `prepare` utilisent maintenant l'API des collections. Dans la majorité des cas, tous les hooks devraient continuer à fonctionner, tant qu'ils n'effectuent pas d'opérations complexes avec des tableaux.

La prise en charge des paramètres `icon` et `badge` a été ajoutée. Les plugins doivent commencer à utiliser ces paramètres et les préférer à un seul paramètre `text`. CSS devrait être utilisé pour contrôler la visibilité de l'étiquette, de l'icône et du badge, au lieu de conditions dans la préparation des éléments de menu.

Tous les menus sont maintenant enveloppés par `nav.elgg-menu-container` pour s'assurer que plusieurs sections de menu ont un élément parent unique, et peuvent être stylisés à l'aide de flexbox ou de floats.

Tous les éléments de menu sont maintenant identifiés avec l'attribut `data-menu-item`, les sections avec `data-menu-section`, et les conteneurs avec les attributs `data-menu-name`.

menu `topbar` :

- l'élément de menu `account` avec la priorité `800` ajouté à la section `alt`
- l'élément de menu `site_notifications` est désormais un enfant de `account` avec pour priorité `100`
- l'élément de menu `usersettings` est désormais un enfant de `account` avec pour priorité `300`
- l'élément de menu `administration` est désormais un enfant de `account` avec pour priorité `800`
- l'élément de menu `logout` est désormais un enfant de `account` avec pour priorité `900`
- l'élément de menu `dashboard` est désormais un enfant de `account` avec pour priorité `100`
- Dans la section `default` (`profile`, `friends`, `messages`), les éléments de menu du noyau utilisent désormais le paramètre `icon` et utilisent CSS pour masquer l'étiquette. Les plugins qui enregistrent des éléments dans cette section et s'attendent à ce qu'une étiquette soit visible doivent mettre à jour leur CSS.
- l'élément de menu `profile` est désormais un enfant de `account`
- l'élément de menu `friends` est désormais un enfant de `account`

menu `entity` :

- l'élément de menu `access` a été supprimé. L'information sur les accès est désormais affichée dans la signature (byline) de l'entité.

menu `user_hover` :

- Tous les éléments utilisent le paramètre `icon`.
- La disposition de la liste déroulante a été modifiée. Si vous avez modifié l'apparence de cette liste déroulante, vous devrez peut-être mettre à jour votre code HTML/CSS.

menu `widget` :

- l'élément de menu `collapse` a été supprimé et les CSS ont été mis à jour en conséquence

menu `title` :

Le plugin `profile` n'utilise plus la section actions du menu `user_hover`, mais enregistre des éléments de menu `title` classiques.

menu `extras` :

Ce menu a été supprimé de la mise en page. Les éléments de menu enregistrés pour ce menu ont été déplacés vers d'autres menus.

menu `groups:my_status` :

Ce menu a été supprimé de la page de profil du groupe.

menu `site_notifications` :

Ce menu a été supprimé. Les objets de notification de site utilisent désormais le menu de l'entité pour les actions.

menu `site` :

L'enregistrement d'un élément de menu personnalisé défini dans l'interface d'administrateur a été déplacé vers le hook `register`, `menu:site`. La vue `navigation/menu/site` a été supprimée. Le menu du site ajoute maintenant un élément de menu `more` directement dans la section `default`.

Icônes des entités

Les fichiers images des icônes par défaut ont été déplacés et re-mappés comme suit :

- Icônes par défaut : `views/default/icon/default/$size.png`
- Icônes utilisateur : `views/default/icon/user/default/$size.gif`
- Icônes de groupe : `views/default/icon/group/default/$size.gif` dans le plugin groups

Les fichiers d'icônes de groupes ont été déplacés de `groups/<guid><size>.jpg` par rapport au répertoire du propriétaire du groupe dans le répertoire de fichiers, à un emplacement indiqué par le service d'icône d'entité. Les plugins doivent cesser d'accéder directement aux fichiers du répertoire de fichiers et utiliser l'API de l'icône de l'entité. Le script de mise à niveau est disponible via l'interface d'administration.

La génération d'icônes d'entité a été modifiée. Toutes les dimensions configurées ne seront plus générées lors de l'appel de l'une des fonctions de l'icône de l'entité (`ElggEntity::saveIconFromUploadedFile`, `ElggEntity::saveIconFromLocalFile` ou `ElggEntity::saveIconFromElggFile`), mais seulement la taille *master*. Les autres dimensions configurées seront générées à partir de l'icône *master* lors de la demande de cette dimension.

Glyphes d'icône

FontAwesome a été mis à niveau vers la version 5.0+. Il y a eu certains changements dans la façon dont les glyphes FontAwesome sont rendus. Le noyau s'occupera de la plupart des changements (par ex. mappage des anciens noms d'icônes à de nouveaux, et en utilisant le préfixe correct pour les icônes de marques et les icônes solides).

Autocomplétion / saisie automatique (sélecteurs d'utilisateurs et de contacts)

Le sélecteur de contacts (« Friends Picker ») est désormais affiché en utilisant `input/userpicker`.

Les plugins devraient :

- Mettez à jour le paramètre `input/userpicker` pour prendre en charge le nouveau paramètre `only_friends`
- Supprimez le CSS du sélecteur de contacts (`friends picker`) de leurs feuilles de style

Collections de Contacts

L'UI des collections de contacts a été déplacée vers son propre plugin - `friends_collections`.

Mise en page des éléments `.elgg-body`

Dans 3.0, ces éléments par défaut ne s'étirent plus pour remplir l'espace disponible dans un contexte de bloc. Ils continuent de supprimer les floats et permettent la césure des mots pour envelopper le texte.

Les modules du noyau et les mises en page qui reposaient sur le remplissage de l'espace ont été retravaillés pour Flexbox, et nous encourageons les développeurs à faire de même, plutôt que d'utiliser la technique problématique `overflow:hidden`.

Supprimer des éléments de la rivière

La fonction `elgg_delete_river()`, qui avait été dépréciée dans la version 2.3, a été rétablie. Des changements notables internes de cette fonction sont ;

- Elle accepte toutes les `$options` de `elgg_get_river()`, mais exige qu'au moins l'un des paramètres suivants soit défini : `id(s)`, `annotation_id(s)`, `subject_guid(s)`, `object_guid(s)`, `target_guid` ou `view(s)`
- Étant donné que `elgg_get_river` a par défaut une limite sur le nombre d'éléments de la rivière qu'elle récupère, si vous souhaitez supprimer tous les éléments de la rivière, vous devez définir `limit` à `false`
- L'accès est ignoré lors de la suppression d'éléments de la rivière
- Les événements sont déclenchés juste avant et après qu'un élément de la rivière a été effacé

Les réponses aux discussions ont été converties en commentaires

Dans la mesure où les réponses aux discussion étaient quasiment une copie des commentaires, toutes les réponses aux discussions ont été migrées vers les commentaires. Toutes les actions, hooks, événements et clefs de traduction associées ont été supprimés.

Note : Les commentaires des discussions apparaîtront désormais dans la section Commentaires de la recherche, et plus sous la section Discussion.

Nettoyage des traductions

Tous les plugins ont été scannés pour les clefs de traduction inutilisées. Les clefs inutilisées ont été supprimées. S'il existait une traduction générique disponible pour la clef de traduction personnalisée, celles-ci ont également été mises à jour.

Journal système

L'API du journal système a été déplacée hors du noyau dans un plugin `system_log`. Les plugins `logbrowser` et `logrotate` ont été fusionnés dans le plugin `system_log`.

Journalisation des erreurs

L'affichage des messages d'erreur de `elgg_log()` et de PHP n'est désormais possible que via le paramètre « Afficher à l'écran » du plugin des développeurs. Consultez le fichier `settings.example.php` pour plus d'informations sur l'utilisation de `$CONFIG->debug` dans votre fichier `settings.php`. Le débogage doit généralement se faire via l'extension `xdebug` ou via `tail -f /chemin/du/fichier/error.log` sur votre serveur.

Le plugin d'actifs de composer n'est plus requis

Les actifs sont maintenant chargés à partir de <https://asset-packagist.org>. Le plugin de composer FXP n'est plus nécessaire lors de l'installation de Elgg ou de la mise à jour des dépendances de composer.

Journaux du Cron

Les journaux du Cron ne sont plus conservés dans la base de données, mais sur le système de fichiers (dans la racine du répertoire des données). Ceci va permettre de conserver des sorties plus longues. Un script de migration a été ajouté pour migrer les anciens paramètres de la base de données vers le nouvel emplacement et supprimer les anciennes valeurs de la base de données.

Clefs de traduction supprimées / modifiées

- Les clefs de traduction relatives aux notifications des commentaires ont changé. Vérifiez les clefs de traduction `generic_comment:notification:owner:`

Les nouvelles fonctionnalités du schéma MySQL ne sont pas appliquées

Les nouvelles installations 3.0 nécessitent MySQL 5.5.3 (ou plus) et utilisent le jeu de caractères utf8mb4 et les colonnes de contenu LONGTEXT (ce qui permet notamment le stockage d'un contenu plus long et des caractères étendus comme emoji).

Changements divers

Les paramètres « Permettre aux visiteurs de s'inscrire » et « Restreindre les pages aux utilisateurs connectés » apparaissent désormais sur la page admin des paramètres basiques (« Basicv Settings »).

Plugin Twitter API

Le plugin `twitter_api` n'est plus livré avec Elgg.

Tests Unitaires et d'Intégration

L'amorçage du PHPUnit de Elgg peut désormais gérer à la fois les tests unitaires et d'intégration. Veuillez noter que **vous ne devriez pas exécuter de test sur un site de production**, car cela peut endommager l'intégrité des données. Pour éviter les pertes de données, vous devez spécifier les paramètres de la base de données via les variables d'environnement. Vous pouvez faire cela via le fichier d'amorçage `phpunit.xml`.

Les plugins peuvent désormais implémenter leurs propres tests PHPUnit en étendant les classes `\Elgg\UnitTestCase` et `\Elgg\IntegrationTestCase`. La suite de test `plugins` va automatiquement charger les tests PHPUnit depuis `mod/<plugin_id>/tests/phpunit/unit` et `mod/<plugin_id>/tests/phpunit/integration`.

Avant d'exécuter des tests d'intégration, vous devez activer les plugins que vous souhaitez tester en même temps que l'API du noyau.

`\Elgg\IntegrationTestCase` utilise le trait `\Elgg\Seeding`, qui peut être utilisé pour construire aisément de nouvelles entités et les enregistrer dans la base de données.

\Elgg\UnitTestCase n'utilise pas la base de données, mais fournit une interface de base de données factice, qui permet aux tests de définir des spécifications de requête avec des retours prédéfinis.

Par défaut, les tests unitaires et d'intégration seront exécutés à chaque fois que `phpunit` est appelé. Vous pouvez utiliser le drapeau `--testsuite` pour n'exécuter qu'une suite de tests spécifique : `phpunit --testsuite unit` ou `phpunit --testsuite integration` ou `phpunit --testsuite plugins`.

Pour que les tests d'intégration s'exécutent correctement, il est recommandé que les plugins ne placent aucune logique de fonctionnement dans la racine de `start.php`, et renvoient plutôt une Closure à la place. Ceci permet à la suite de test de construire une nouvelle instance Application sans perdre la logique d'initialisation du plugin.

Les plugins avec `simpletests` vont continuer à fonctionner comme précédemment. Cependant, les méthodes de signature dans la classe abstraite `ElggCoreUnitTest` ont changé et vous allez avoir besoin de mettre à jour vos tests en conséquence. A savoir, il est déconseillé d'utiliser les méthodes `__construct` et `__destruct`. `setUp` et `tearDown` ont été marquées comme privées et sont utilisées pour amorcer des tests cohérents et pour déclarer des conditions `pre` et `post` ; votre cas de test devrait utiliser les méthodes `up` et `down` à leur place.

`Simpletests` ne peut plus être exécuté depuis l'interface admin du plugin `developers`. Utilisez la commande `Elgg cli : elgg-cli simpletest`

De 2.2 à 2.3

Contenus

- *Version de PHP*
- *APIs dépréciées*
- *Vues obsolètes*
- *Nouvelle API pour la gestion des pages et des actions*
- *Nouvelle API pour travailler avec l'envoi de fichiers*
- *Nouvelle API pour la manipulation d'images*
- *Nouvelle API pour les événements*
- *Nouvelle API pour la signature des URLs*
- *Vues de formulaire extensibles*
- *Métadonnées `access_id`*
- *Nouvelle API pour extraire des noms de classes à partir des tableaux*
- *Notifications*
- *Les fonctions de liste d'entités peuvent générer des tableaux*
- *Composants d'onglets en ligne*
- *API pour modifier l'URL d'inscription et de connexion*
- *Prise en charge des jeux de champs (fieldsets) dans les formulaires*
- *Lightbox*

Version de PHP

PHP 5.5 a atteint la date de fin de support en Juillet 2016. Afin de s'assurer que les sites Elgg soient sécurisés, nous exigeons désormais PHP 5.6 pour les nouvelles installations.

Les installations existantes peuvent continuer à utiliser PHP 5.5 jusqu'à Elgg 3.0.

Afin de mettre à niveau Elgg vers la 2.3 en utilisant `composer` avec PHP 5.5, vous pouvez avoir besoin d'utiliser le drapeau `--ignore-platform-reqs`.

APIs dépréciées

- Enregistrement pour le hook `to:object` par le nom de l'extenseur : Utilisez plutôt les hooks `to:object`, `annotation` et `to:object`, `metadata` à la place.
- `ajax_forward_hook()` : N'est plus utilisé comme gestionnaire pour le hook `"forward", "all"`. La réponse Ajax est maintenant enveloppée par la `ResponseFactory`
- `ajax_action_hook()` : N'est plus utilisé comme gestionnaire pour le hook `"action", "all"`. La mise en mémoire tampon de la sortie commence maintenant avant que le hook ne soit déclenché dans `ActionsService`
- `elgg_error_page_handler()` : N'est plus utilisé comme gestionnaire pour les hooks `"forward", <error_code>`
- `get_uploaded_file()` : Utilisez la nouvelle API d'envoi de fichiers à la place
- `get_user_notification_settings()` : Utilisez `ElggUser::getNotificationSettings()`
- `set_user_notification_setting()` : Utilisez `ElggUser::setNotificationSetting()`
- événement `pagesetup`, `system` : Utilisez le menu ou les hooks de la coquille de la page (page shell) à la place.
- `elgg.walled_garden` ce JavaScript est obsolète : Utilisez le module AMD `elgg/walled_garden` à la place.
- `elgg()->getDb()->getTableprefix()` : Utilisez `elgg_get_config('dbprefix')`.
- `update_entity_last_action()` privée : Évitez de mettre à jour manuellement le timestamp de la dernière action.
- Définir un `access_id` non-public sur une métadonnée est obsolète. Voyez ci-dessous.
- `get_resized_image_from_existing_file()` : Utilisez `elgg_save_resized_image()`.
- `get_resized_image_from_uploaded_file()` : Utilisez `elgg_save_resized_image()` en combinaison avec l'API d'envoi de fichier (upload).
- `get_image_resize_parameters()` sera supprimé.
- `elgg_view_input()` : Utilisez `elgg_view_field()`. Toutes nos excuses pour le changement d'API.

Vues obsolètes

- `resources/file/world` : Utilisez la vue `resources/file/all` à la place.
- `resources/pages/world` : Utilisez la vue `resources/pages/all` à la place.
- `walled_garden.js` : Utilisez le module `elgg/walled_garden` à la place.

Nouvelle API pour la gestion des pages et des actions

Les gestionnaires de pages et les fichiers de script d'action doivent maintenant renvoyer une instance de `\Elgg\Http\ResponseBuilder`. Les plugins devraient utiliser les fonctions suivantes pour créer des réponses :

- `elgg_ok_response()` envoie une réponse 2xx avec des données HTML (gestionnaire de page) ou JSON (actions)
- `elgg_error_response()` envoie une réponse 4xx ou 5xx sans contenu/données
- `elgg_redirect_response()` redirige silencieusement la requête

Nouvelle API pour travailler avec l'envoi de fichiers

- `elgg_get_uploaded_files()` - renvoie un tableau d'objets `Symfony` de fichiers téléchargés
- `ElggFile::acceptUploadedFile()` - déplace un fichier téléchargé dans le répertoire de fichiers de Elgg

Nouvelle API pour la manipulation d'images

Le nouveau service de manipulation d'images met en œuvre une approche plus efficace pour recadrer et redimensionner les images.

- `elgg_save_resized_image()` - recadre et redimensionne une image aux dimensions choisies

Nouvelle API pour les événements

- `elgg_clear_event_handlers()` - similaire à `elgg_clear_plugin_hook_handlers` cette fonction supprime tous les gestionnaires d'événements enregistrés

Nouvelle API pour la signature des URLs

Les URL peuvent désormais être signées avec une clé HMAC SHA-256 et validées à tout moment avant l'expiration de l'URL. Cette fonctionnalité peut être utilisée pour tokeniser l'URL d'action dans les notifications par e-mail, ainsi que d'autres utilisations en dehors de l'installation Elgg.

- `elgg_http_get_signed_url()` - signe l'URL avec la clef HMAC
- `elgg_http_validate_signed_url()` - valide l'URL signée
- `elgg_signed_request_gatekeeper()` - gardien qui valide la signature de la requête en cours

Vues de formulaire extensibles

Le rendu du pied de page du formulaire peut maintenant être reporté jusqu'à ce que la vue du formulaire et ses extensions aient terminé le rendu. Cela permet aux plugins de collaborer sur les vues de formulaire sans casser la logique de balisage.

- `elgg_set_form_footer()` - définit le pied de page du formulaire pour le rendu différé
- `elgg_get_form_footer()` - renvoie le pied de page du formulaire défini

Métadonnées `access_id`

Il est maintenant obsolète de créer des métadonnées avec une valeur explicite de `access_id` autre que `ACCESS_PUBLIC`.

Dans Elgg 3.0, les métadonnées ne seront pas contrôlées et seront disponibles dans tous les contextes. Si votre plugin repose sur le contrôle d'accès des métadonnées, il serait sage de migrer le stockage vers des annotations ou des entités à la place.

Nouvelle API pour extraire des noms de classes à partir des tableaux

Semblable à `elgg_extract()`, `elgg_extract_class()` extrait la clef `class` (si elle est présente), la fusionne avec les noms de classe existants, et renvoie toujours un tableau.

Notifications

- Un hook de haut niveau 'prepare', 'notification' est maintenant déclenché pour les notifications instantanées et d'abonnement et peut être utilisé pour modifier les objets de notification quel que soit leur type.
- le hook 'format', 'notification:<method>' est maintenant déclenché pour les notifications instantanées et d'abonnement et peut être utilisé pour formater la notification (par ex. pour supprimer les balises HTML, envelopper le corps de notification dans un modèle, etc.).
- Les notifications instantanées sont désormais traitées par le service de notifications, de sorte que la quasi-totalité des hooks applicables aux notifications d'abonnement s'appliquent également aux notifications instantanées.
- `elgg_get_notification_methods()` peut être utilisé pour obtenir les méthodes de notification enregistrées
- Ajout de `ElggUser::getNotificationSettings()` et `ElggUser::setNotificationSetting()`

Les fonctions de liste d'entités peuvent générer des tableaux

Dans des fonctions telles que `elgg_list_entities($options)`, le rendu sous forme de tableau est possible en définissant `$options['list_type'] = 'table'` et en fournissant un tableau de colonnes de tableau dans `$options['columns']`. Chaque colonne est un objet `Elgg\Views\TableColumn`, généralement créé via des méthodes du service `elgg()->table_columns`.

Les plugins peuvent fournir ou modifier ces méthodes d'usine (voir `Elgg\Views\TableColumn\Columnfactory`). Pour un exemple d'utilisation, consultez la vue `admin/users/newest`.

Composants d'onglets en ligne

Le composant szq onglets en ligne peut désormais être rendu avec l'affichage `page/composants/onglets`. Les composants permettent de basculer entre pré-rempli et chargé-par-ajax. Voir `page/components/tabs` dans les vues du noyau et `theme_sandbox/components/tabs` dans le plugin `developers` pour les instructions et les exemples d'utilisation.

API pour modifier l'URL d'inscription et de connexion

- `elgg_get_registration_url()` doit être utilisé pour obtenir l'URL d'inscription sur le site
- `elgg_get_login_url()` doit être utilisé pour obtenir l'URL de connexion du site
- le hook `registration_url`, `site` peut être utilisé pour modifier l'URL d'inscription par défaut
- le hook `login_url`, `site` peut être utilisé pour modifier l'URL de connexion par défaut

Prise en charge des jeux de champs (fieldsets) dans les formulaires

- `elgg_view_field()` remplace `elgg_view_input()`. Elle dispose d'une API similaire, mais accepte un seul tableau.
- `elgg_view_field()` prend en charge `#type`, `#label`, `#help` et `#class`, permettant l'envoi de versions non préfixées à la vue d'entrée `$vars`.
- La nouvelle vue `input/fieldset` peut être utilisée pour rendre un ensemble de champs, chacun rendu avec `elgg_view_field()`.

Lightbox

- Le CSS de Lightbox n'est plus chargé en tant que fichier CSS externe. Le thème Lightbox étend maintenant `elgg.css` et `admin.css`
- La configuration de lightbox par défaut est maintenant définie via le hook '`elgg.data`', '`site`' côté serveur

De 2.1 à 2.2

Contenus

- *APIs dépréciées*
- *Vues obsolètes*
- *Ajout du module `elgg/popup`*
- *Ajout du module `elgg/lightbox`*
- *Ajout du module `elgg/embed`*
- *Nouvelle API pour la manipulation des icônes d'entités*
- *APIs supprimées*
- *Un module `elgg/ckeditor` amélioré*

APIs dépréciées

- Bibliothèque JavaScript `elgg.ui.river` : Supprimez les appels à `elgg_load_js('elgg.ui.river')` du code du plugin. Mettez à jour `core/river/filter` et `forms/comment/save`, s'ils sont remplacés, pour exiger des modules de composants AMD
- Méthodes `elgg.ui.popupOpen()` et `elgg.ui.popupClose()` dans la bibliothèque JS `elgg.ui` : Utilisez plutôt le module `elgg/popup`.
- Bibliothèque `lightbox.js` : n'utilisez pas `elgg_load_js('lightbox.js')`; à moins que votre code ne référence l'espace de nom déprécié `elgg.ui.lightbox`. Utilisez plutôt le module AMD `elgg/lightbox`.
- Bibliothèque `elgg.embed` et objet `elgg.embed` : N'utilisez pas `elgg_load_js('elgg.embed')`. Utilisez plutôt le module AMD `elgg/embed`
- Accéder directement à la valeur de configuration `icons_sizes` : Utilisez `elgg_get_icon_sizes()`
- `can_write_to_container()` : Utilisez `ElggEntity::canWriteToContainer()`

Vues obsolètes

- `elgg/ui.river.js` est déprécié : ne vous fiez pas aux URLs de simplecache pour travailler.
- `groups/js` est déprécié : utilisez à la place le module AMD `groups/navigation` comme dépendance d'élément du menu pour les éléments de menu « feature » (mettre en Une) et « un-feature » (retirer de la Une).
- `lightbox/settings.js` est déprécié : utilisez l'attribut `getOptions`, `ui.lightbox` ou `data-colorbox-opts` du hook plugin JS.
- `elgg/ckeditor/insert.js` est déprécié : vous n'avez plus besoin de l'inclure, l'enregistrement des hooks a lieu dans le module `elgg/ckeditor`
- `embed/embed.js` est déconseillé : utilisez le module AMD `elgg/embed`.

Ajout du module elgg/popup

Le nouveau *module elgg/popup* peut être utilisé pour créer des interactions plus complexes de déclenchement de popup, y compris la liaison des types d'ancrage personnalisés et l'ouverture/fermeture de fenêtres contextuelles programmatically.

Ajout du module elgg/lightbox

Le nouveau *module elgg/lightbox* peut être utilisé pour ouvrir et fermer la lightbox par programme.

Ajout du module elgg/embed

Même s'il est rarement nécessaire, le module AMD elgg/embed peut être utilisé pour accéder aux méthodes d'intégration par programme. Le module s'initialise lui-même si nécessaire et est peu susceptible de nécessiter une décoration supplémentaire.

Nouvelle API pour la manipulation des icônes d'entités

- ElggEntity implémente désormais l'interface \Elgg\EntityIcon
- elgg_get_icon_sizes() - renvoie les tailles d'icônes spécifiques au type/sous-type d'entité spécifié
- ElggEntity::saveIconFromUploadedFile() - crée des icônes à partir d'un fichier téléchargé
- ElggEntity::saveIconFromLocalFile() - crée des icônes à partir d'un fichier local
- ElggEntity::saveIconFromElggFile() - crée des icônes à partir d'une instance de ElggFile
- ElggEntity::getIcon() - renvoie une instance de ElggIcon qui pointe vers l'emplacement de l'icône de l'entité dans le répertoire de fichiers (il peut s'agir simplement d'un espace réservé, utilisez ElggEntity::hasIcon() pour vérifier si le fichier a été écrit)
- ElggEntity::deleteIcon() - supprime les icônes d'entité
- ElggEntity::getIconLastChange() - retourne la date de modification du fichier icône
- ElggEntity::hasIcon() - vérifie si une icône de cette dimension a bien été créé
- elgg_get_embed_url() - peut être utilisé pour retourner une URL pour intégrer l'icône d'une entité (serve via le gestionnaire /serve-icon)

Les avatars des utilisateurs sont désormais servis via le gestionnaire serve-file. Les plugins devraient commencer à utiliser elgg_get_inline_url() et noter que :

- le gestionnaire de page et la vue ressource ``avatar/view`` sont devenus obsolètes
- le fichier /mod/profile/icondirect.php est devenu obsolète
- profile_set_icon_url() n'est plus enregistré comme fonction de rappel (callback) pour le hook plugin "entity:icon:url", "user"

Les avatars de groupe sont désormais servis via le gestionnaire serve-file. Les plugins devraient commencer à utiliser elgg_get_inline_url() et noter que :

- le gestionnaire de page groupicon (groups_icon_handler()) est devenu obsolète
- le fichier /mod/groups/icon.php est devenu obsolète

Les miniatures et les téléchargements sont désormais servis via le gestionnaire serve-file. Les plugins devraient commencer à utiliser elgg_get_inline_url() et elgg_get_download_url() et noter que :

- le gestionnaire de page et la vue ressource ``file/download`` sont devenus obsolètes
- le fichier mod/file/thumbnail.php est devenu obsolète
- Plusieurs vues ont été mises à jour pour utiliser les nouvelles URLs de téléchargement, notamment :
 - mod/file/views/default/file/specialcontent/audio/default.php
 - mod/file/views/default/file/specialcontent/image/default.php
 - mod/file/views/default/resources/file/view.php
 - mod/file/views/rss/file/enclosure.php

APIs supprimées

Juste un avertissement pour indiquer que les fonctions de caches d'entité privée (par ex. `_elgg_retrieve_cached_entity`) ont été supprimées. Quelques plugins ont pu les utiliser. Les plugins ne devraient pas utiliser les API privées dans la mesure où elles seront plus souvent supprimées sans avertissement.

Un module `elgg/ckeditor` amélioré

le module `:doc:elgg/ckeditor` </guides/javascript> peut maintenant être utilisé pour ajouter un éditeur WYSIWYG à un textarea de manière programmatique avec `elgg/ckeditor#bind`.

De 2.0 à 2.1

Contenus

- *APIs dépréciées*
- *changements de `Application::getDb()`*
- *Ajout du module `elgg/widgets`*

APIs dépréciées

- `ElggFile::setFilestore`
- `get_default_filestore`
- `set_default_filestore`
- `elgg_get_config('siteemail')` : Utilisez `elgg_get_site_entity()->email`
- URLs commençant par `/css/` et `/js/` : Utilisez `elgg_get_simplecache_url()`
- `elgg.ui.widgets` L'objet JavaScript est déprécié par le module AMD `elgg/widgets`

changements de `Application::getDb()`

Si vous utilisez cette API bas niveau, ne vous attendez pas à ce qu'elle renvoie une instance `Elgg\Database` dans 3.0. Elle renvoie maintenant un `Elgg\Application\Database` avec de nombreuses alertes de dépréciation. Ces méthodes n'ont jamais été destinées à devenir une API publique, mais nous ferons de notre mieux pour les prendre en charge dans en 2.x.

Ajout du module `elgg/widgets`

Si le code de votre plugin appelle `elgg.ui.widgets.init()`, utilisez plutôt le *module `elgg/widgets`*.

De 1.x à 2.0

Contenus

- *Elgg peut maintenant être installé en tant que dépendance de composer au lieu d'être situé à la racine*
- *Les vues cacheables doivent avoir une extension de fichier dans leurs noms*
- *Abandon de jquery-migrate et mise à niveau de jquery vers ^2.1.4*
- *JS et CSS ont été déplacés hors des répertoires js/ et css/*
- *fxp/composer-asset-plugin est maintenant nécessaire pour installer Elgg à partir de la source*
- *Liste des vues dépréciées et des arguments des vues qui ont été supprimés*
- *Tous les scripts ont été déplacés vers le bas de la page*
- *Le formateur d'attributs supprime les clefs avec des tirets bas*
- *Fil d'Ariane*
- *Callbacks dans les Requêtes*
- *Hook plugin des commentaires*
- *Hook des permissions du conteneur*
- *La création ou la suppression d'une relation déclenche un seul événement*
- *La fonction de discussion a été extraite des groupes vers son propre plugin*
- *Fonctionnalité de connexion via https abandonnée*
- *Elgg a migré d'ext/mysql vers PDO MySQL*
- *L'ordre d'appel d'un événement/hook pourrait changer*
- *Les URL **export/** ne sont plus disponibles*
- *Icônes migrés vers Font Awesome*
- *Augmentation de la valeur de l'index z dans la classe elgg-menu-site*
- *vue input/autocomplete*
- *Introduction d'une bibliothèque tierce pour l'envoi de courriels*
- *Éléments d'étiquetage*
- *Plugin Aalborg Theme*
- *Plugin Likes*
- *Plugin Messages*
- *Plugin Blog*
- *Plugin Bookmarks*
- *Plugin File*
- *Classes supprimées*
- *Clefs retirées disponibles via elgg_get_config()*
- *Fonctions supprimées*
- *Méthodes supprimées*
- *Hooks des plugins supprimés*
- *Actions supprimées*
- *Vues supprimées*
- *Variables des vues supprimées*
- *Bibliothèques supprimées*
- *Spécifier des Vues via les propriétés*
- *Le type de vue est statique après l'appel initial elgg_get_viewtype()*
- *Dépréciation*

Elgg peut maintenant être installé en tant que dépendance de composer au lieu d'être situé à la racine

Cela signifie qu'un site Elgg peut ressembler à ceci :

```
settings.php
vendor/
  elgg/
    elgg/
      engine/
        start.php
      _graphics/
        elgg_sprites.png
mod/
  blog
  bookmarks
  ...
```

`elgg_get_root_path` et `$CONFIG->path` vont renvoyer le chemin vers le répertoire racine de l'application, qui n'est pas nécessairement le même que le répertoire racine d'Elgg core (qui dans ce cas est `vendor/elgg/elgg/`).

N'essayez pas d'accéder directement au noyau Elgg depuis votre plugin, car vous ne pouvez pas compter sur son emplacement dans le système de fichiers.

En particulier, n'essayez pas de charger `engine/start.php`.

```
// Don't do this!
dirname(__DIR__) . "/engine/start.php";
```

Pour démarrer Elgg manuellement, vous pouvez utiliser la classe `Elgg\Application`.

```
// boot Elgg in mod/myplugin/foo.php
require_once dirname(dirname(__DIR__)) . '/vendor/autoload.php';
\Elgg\Application::start();
```

Cependant, utilisez cette approche avec parcimonie. Préférez *Routage* dans la mesure du possible, car cela permet de découpler vos URLs publiques et votre mise en page du système de fichiers.

En outre, n'essayez pas d'accéder directement aux fichiers `_graphics`.

```
readfile(elgg_get_root_path() . "_graphics/elgg_sprites.png");
```

Utilisez *Vues* à la place :

```
echo elgg_view('elgg_sprites.png');
```

Les vues cacheables doivent avoir une extension de fichier dans leurs noms

Cette exigence nous permet de servir des actifs directement à partir du disque pour des raisons de performance, au lieu de les servir par PHP.

Il est également beaucoup plus facile d'explorer les ressources disponibles dans le cache en allant vers `data-root/views_simplecache` et en naviguant à partir de là.

- Mauvais : `my/cool/template`
- Bon : `my/cool/template.html`

Nous mettons maintenant en cache des ressources par `$viewtype/$view`, et plus `md5('$viewtype|$view')`, qui entraînait des conflits entre les vues pouvant être mises en cache qui n'avaient pas d'extension de fichier pour les distinguer des répertoires.

Abandon de jquery-migrate et mise à niveau de jquery vers ^2.1.4

l'API jQuery 2.x est compatible avec 1.x, mais abandonne la prise en charge de IE8-, que Elgg ne prend plus en charge depuis un certain temps de toutes façons.

Lisez <http://jquery.com/upgrade-guide/1.9/> pour savoir comment se passer de jquery-migrate.

Si vous préférez la conserver, vous pouvez utiliser ce code dans l'initialisation de votre plugin :

```
elgg_register_js('jquery-migrate', elgg_get_simplecache_url('jquery-migrate.js'), 'head
↪');
elgg_load_js('jquery-migrate');
```

Définissez également une vue `jquery-migrate.js` contenant le contenu du script.

JS et CSS ont été déplacés hors des répertoires js/ et css/

Ils ont également reçu des extensions `.js` et `.css` respectivement s'ils ne les avaient pas déjà :

Ancienne vue	Nouvelle vue
<code>js/view</code>	<code>view.js</code>
<code>js/other.js</code>	<code>other.js</code>
<code>css/view</code>	<code>view.css</code>
<code>css/other.css</code>	<code>other.css</code>
<code>js/img.png</code>	<code>img.png</code>

Le principal avantage que cela apporte est d'être en mesure de co-localiser les actifs connexes. Ainsi, un modèle (`view.php`) peut avoir ses dépendances CSS/JS juste à côté (`view.css`, `view.js`).

Un grand soin a été pris pour rendre cette modification aussi rétro-compatible que possible, de sorte que vous ne devriez pas avoir besoin de mettre à jour les références des vues immédiatement. Cependant, vous êtes certainement encouragés à déplacer vos vues JS et CSS vers leurs nouveaux emplacements canoniques.

En pratique, ceci introduit quelques pièges :

Les hooks `view_vars`, `$view_name` et `view`, `$view_name` fonctionneront sur le nom *canonique* de la vue :

```
elgg_register_plugin_hook_handler('view', 'css/elgg', function($hook, $view_name) {
    assert($view_name == 'elgg.css') // not "css/elgg"
});
```

L'utilisation du hook `view`, `all` et la vérification des vues individuelles peuvent ne pas fonctionner comme prévu :

```
elgg_register_plugin_hook_handler('view', 'all', function($hook, $view_name) {  
    // Won't work because "css/elgg" was aliased to "elgg.css"  
    if ($view_name == 'css/elgg') {  
        // Never executed...  
    }  
  
    // Won't work because no canonical views start with css/* anymore  
    if (strpos($view_name, 'css/') === 0) {  
        // Never executed...  
    }  
});
```

Merci de nous faire part de tous les autres problèmes de rétro-compatibilité que ce changement provoque. Nous aimerions en résoudre le plus grand nombre possible pour faciliter la transition.

fxp/composer-asset-plugin est maintenant nécessaire pour installer Elgg à partir de la source

Nous utilisons `fxp/composer-asset-plugin` pour gérer nos actifs de navigateur (js, css, html) avec Composer, mais il doit être installé globalement *avant d'installer Elgg* afin que les packages `bower-asset/*` soient reconnus. Pour l'installer, exécutez :

```
composer global require fxp/composer-asset-plugin
```

Si vous ne le faites pas avant d'exécuter `composer install` ou `composer create-project`, vous obtiendrez un message d'erreur :

```
[InvalidArgumentException]  
Package fxp/composer-asset-plugin not found
```

Liste des vues dépréciées et des arguments des vues qui ont été supprimés

Nous avons abandonné la prise en charge et/ou supprimé les vues suivantes :

- `canvas/layouts/*`
- `categories`
- `categories/view`
- `core/settings/tools`
- `embed/addcontentjs`
- `footer/analytics` (Utilisez `page/elements/foot` à la place)
- `groups/left_column`
- `groups/right_column`
- `groups/search/finishblurb`
- `groups/search/startblurb`
- `input/calendar` (Utilisez `input/date` à la place)
- `input/datepicker` (Utilisez `input/date` à la place)
- `input/pulldown` (Utilisez `input/select` à la place)
- `invitefriends/formitems`
- `js/admin` (Utilisez AMD et `elgg_require_js` au lieu d'étendre les vues JS existantes)
- `js/initialise_elgg` (Utilisez AMD et `elgg_require_js` au lieu d'étendre les vues JS)
- `members/nav`
- `metatags` (Utilisez le hook plugin "head", "page" à la place)

- navigation/topbar_tools
- navigation/viewtype
- notifications/subscriptions/groupsform
- object/groupforumtopic
- output/calendar (Utilisez output/date à la place)
- output/confirmlink (Utilisez output/url à la place)
- page_elements/contentwrapper
- page/elements/shortcut_icon (Utilisez le hook plugin “head”, “page” plugin hook à la place)
- page/elements/wrapper
- profile/icon (Utilisez elgg_get_entity_icon)
- river/object/groupforumtopic/create
- settings/{plugin}/edit (Utilisez plugins/{plugin}/settings à la place)
- user/search/finishblurb
- user/search/startblurb
- usersettings/{plugin}/edit (Utilisez plugins/{plugin}/usersettings à la place)
- widgets/{handler}/view (Utilisez widgets/{handler}/content à la place)

Nous avons également abandonné les arguments suivants des vues :

- « value » dans output/iframe (Utilisez plutôt « src »)
- « area2 » et « area3 » dans page/elements/sidebar (utilisez plutôt « sidebar » ou l’extension de vue)
- « js » dans les vues des icônes (par exemple icon/user/default)
- Les options pour input/radio et input/checkboxes qui ne sont pas des paires de clef-valeur ne seront plus acceptées.

Tous les scripts ont été déplacés vers le bas de la page

Vous devriez tester votre plugin **avec la console d’erreur JavaScript visible**. Pour des raisons de performance, Elgg ne prend plus en charge les éléments script dans l’élément head ou dans les vues HTML. elgg_register_js chargera désormais *tous les scripts* à la fin de l’élément body.

Vous devez convertir les scripts en ligne en [AMD](#) ou en scripts externes chargés avec elgg_load_js.

Au début de la page, Elgg fournit un shim de la fonction RequireJS `require()` qui met simplement le code en file d’attente jusqu’à ce que les modules AMD elgg et jQuery soient définis. Cela fournit un moyen simple de convertir de nombreux scripts en ligne pour utiliser `require()`.

Du code en ligne échouera car la pile n’est pas encore chargée :

```
<script>
$(function () {
    // code using $ and elgg
});
</script>
```

Ceci devrait fonctionner dans Elgg 2.0 :

```
<script>
require(['elgg', 'jquery'], function (elgg, $) {
    $(function () {
        // code using $ and elgg
    });
});
</script>
```

Le formateur d'attributs supprime les clefs avec des tirets bas

`elgg_format_attributes()` (et toutes les API qui l'utilisent) filtrent désormais les attributs dont le nom contient un tiret baswordre (underscore). Toutefois, si l'attribut commence par `data-`, il ne sera pas supprimé.

Fil d'Ariane

L'affichage du fil d'Ariane supprime maintenant le dernier élément s'il ne contient pas de lien. Pour restaurer le comportement précédent, remplacez le gestionnaire de hook plugin `elgg_prepare_breadcrumbs` par le vôtre :

```
elgg_unregister_plugin_hook_handler('prepare', 'breadcrumbs', 'elgg_prepare_breadcrumbs');
elgg_register_plugin_hook_handler('prepare', 'breadcrumbs', 'myplugin_prepare_breadcrumbs');

function myplugin_prepare_breadcrumbs($hook, $type, $breadcrumbs, $params) {
    // just apply excerpt to titles
    foreach (array_keys($breadcrumbs) as $i) {
        $breadcrumbs[$i]['title'] = elgg_get_excerpt($breadcrumbs[$i]['title'], 100);
    }
    return $breadcrumbs;
}
```

Callbacks dans les Requêtes

Assurez-vous d'utiliser uniquement des valeurs *appelables* (callable) valides pour l'argument/les options de la fonction de rappel dans l'API.

Les fonctions de requête lanceront désormais une `RuntimeException` si `is_callable()` renvoie `false` comme valeur de la fonction de rappel donnée. Cela comprend des fonctions telles que `elgg_get_entities()`, `get_data()`, et bien d'autres encore.

Hook plugin des commentaires

Les plugins peuvent désormais renvoyer une chaîne vide à partir du hook `'comments'`, `$entity_type` afin de remplacer la vue du composant de commentaires par défaut. Pour forcer le composant de commentaires par défaut, votre plugin doit renvoyer `false`. Si vous utilisiez des chaînes vides pour forcer l'affichage des commentaires par défaut, vous devez mettre à jour vos gestionnaires de crochet pour qu'ils renvoient `false`.

Hook des permissions du conteneur

Le comportement du hook `container_permissions_check` a changé lorsqu'une entité est créée : avant 2.0, le hook était appelé deux fois si le conteneur de l'entité n'était pas le propriétaire. Lors du premier appel, le propriétaire de l'entité était transmis en tant que `$params['container']`, ce qui pouvait créer des confusion dans les gestionnaires.

Dans 2.0, lorsqu'une entité est créée dans un conteneur comme un groupe, si le propriétaire est le même que l'utilisateur connecté (ce qui est presque toujours le cas), cette première vérification est contournée. Ainsi, le hook `container_permissions_check` sera presque toujours appelé une seule fois avec `$params['container']` comme conteneur correct de l'entité.

La création ou la suppression d'une relation déclenche un seul événement

Les événements de relation « create » et « delete » sont désormais déclenchés une seule fois, avec "relationship" comme type d'objet.

Par ex. Écouter l'événement "create", "member" ou "delete", "member" ne capturera plus les ajouts/suppressions d'appartenance au groupe. Utilisez les événements "create", "relationship" ou "delete", "relationship" events.

La fonction de discussion a été extraite des groupes vers son propre plugin

Le sous-type object, groupforumtopic a été remplacé par le sous-type object, discussion. Si votre plugin utilise ou modifie l'ancienne fonctionnalité de discussion, vous devez la mettre à niveau pour utiliser le nouveau sous-type.

Rien ne change du point de vue du propriétaire du groupe. La fonction de discussion est toujours disponible en tant qu'outil de groupe et toutes les discussions anciennes sont intactes.

Fonctionnalité de connexion via https abandonnée

Pour une meilleure sécurité et de meilleures performances, servez toutes les pages par HTTPS en changeant le schéma dans le wwwroot de votre site en https dans <http://votresite.tld/admin/settings/advanced>

Elgg a migré d'ext/mysql vers PDO MySQL

Elgg utilise désormais une connexion PDO_MYSQL et n'utilise plus de fonctions ext/mysql. Si vous utilisez les fonctions mysql_ *, en vous appuyant implicitement sur une connexion ouverte, celles-ci échoueront.

Si votre code utilise l'une des fonctions suivantes, lisez ce qui suit.

- execute_delayed_write_query()
- execute_delayed_read_query()

Si vous fournissez un objet \$handler qui doit être appelé avec les résultats, votre gestionnaire recevra désormais un objet \Doctrine\DBAL\Driver\Statement. Auparavant, il s'agissait d'une ressource ext/mysql result.

L'ordre d'appel d'un événement/hook pourrait changer

Lors de l'inscription aux événements/hooks, le mot clé all pour la correspondance générique n'a plus aucun effet sur l'ordre dans lequel les gestionnaires sont appelés. Pour vous assurer que votre gestionnaire est appelé en dernier, vous devez lui donner la plus haute priorité de tous les gestionnaires correspondants, ou pour vous assurer que votre gestionnaire est appelé en premier, vous devez lui donner la plus faible priorité de tous les gestionnaires correspondants.

Si les gestionnaires ont été enregistrés avec la même priorité, ceux-ci sont appelés dans l'ordre où ils ont été enregistrés.

Pour émuler le comportement antérieur, les gestionnaires du noyau Elgg enregistrés avec le mot clé all ont été relevés en priorité. Certains de ces gestionnaires seront très probablement appelés dans un ordre différent.

Les URL export/ ne sont plus disponibles

Elgg ne prend plus en charge ce point de terminaison pour exposer les données de ressources.

Icônes migrés vers Font Awesome

Les sprites d'Elgg et la plupart des classes CSS commençant par `elgg-icon-` ont été supprimées.

L'utilisation de `elgg_view_icon()` est rétro-compatible, mais le HTML statique utilisant les classes `elgg-icon` devra être mis à jour vers le nouveau balisage.

Augmentation de la valeur de l'index z dans la classe elgg-menu-site

La valeur de l'index z dans la classe `elgg-menu-site` est passée de 1 à 50 pour permettre aux éléments de page de la zone de contenu d'utiliser la propriété z-index sans que le menu du site « Plus » (More) ne s'affiche derrière ces éléments. Si votre plugin/thème remplace la classe `elgg-menu-site` ou les vues/default/elements/navigation.css veuillez ajuster la valeur de z-index dans votre fichier CSS modifié en conséquence.

vue input/autocomplete

Les plugins qui remplacent la vue `input/autocomplete` devront inclure l'URL source dans l'attribut `data-source` de l'élément d'entrée, exiger le nouveau module AMD `elgg/autocomplete` et appeler sa méthode `init`. La bibliothèque javascript 1.x `elgg.autocomplete` n'est plus utilisée.

Introduction d'une bibliothèque tierce pour l'envoi de courriels

Nous utilisons l'excellente bibliothèque `Zend\Mail` pour envoyer des courriels dans Elgg 2.0. Il y a probablement des cas particuliers que la bibliothèque gère différemment de Elgg 1.x. Prenez soin de tester attentivement vos notifications par e-mail lors de la mise à niveau vers 2.0.

Éléments d'étiquetage

Les vues suivantes ont reçu des éléments `label` autour de certains des champs d'entrée. Si votre plugin/thème remplace ces vues, veuillez vérifier le nouveau contenu.

- `views/default/core/river/filter.php`
- `views/default/forms/admin/plugins/filter.php`
- `views/default/forms/admin/plugins/sort.php`
- `views/default/forms/login.php`

Plugin Aalborg Theme

La vue `page/elements/navbar` utilise désormais une icône Font Awesome pour le sélecteur de menu mobile au lieu d'une image. L'image `bars.png` et la prise en charge de CSS pour le rendu 1.12 ont été supprimées, aussi mettez à jour votre thème en conséquence.

Plugin Likes

Les objets ne peuvent plus être aimés (« likés ») par défaut. Pour qu'ils puissent être aimés, vous pouvez enregistrer un gestionnaire pour permettre l'annotation, ou plus simplement vous inscrire au hook `["likes:is_likable", <type>":<subtype>"]` et renvoyer `true`. Par ex.

```
elgg_register_plugin_hook_handler('likes:is_likable', 'object:mysubtype', 'Elgg\
↳ Values::getTrue');
```

Tout comme auparavant, le hook `permissions_check:annotated` est toujours appelé et peut être utilisé pour remplacer le comportement par défaut.

Plugin Messages

Si vous avez supprimé ou remplacé la fonction de gestionnaire `messages_notifier` pour masquer/modifier l'icône de la boîte de réception, vous devrez plutôt le faire de même pour le gestionnaire de menu de la barre supérieure (topbar) `messages_register_topbar.messages_notifier` n'est plus utilisé pour ajouter le lien de menu.

Les messages ne recevront plus les métadonnées “msg” pour les messages nouvellement créés. Cela signifie que vous ne pouvez plus compter sur ces métadonnées.

Plugin Blog

Les pages de blog qui affichent les articles “Mine” (Moi) ou “Friends” (Contacts) ont été modifiées pour lister tous les articles de blog appartenant aux utilisateurs (y compris ceux créés dans les groupes).

Plugin Bookmarks

Les pages des signets qui affichent les listes de signets “Mine” (Moi) ou “Friends” (Contacts) ont été modifiées pour lister tous les signets appartenant aux utilisateurs (y compris ceux créés dans les groupes).

Plugin File

Les pages de fichiers qui affichent les listes de fichiers “Mine” (Moi) ou “Friends” (Contacts) ont été modifiées pour lister tous les fichiers appartenant aux utilisateurs (y compris ceux créés dans les groupes).

Classes supprimées

- ElggInspector
- Notable
- FilePluginFile : remplacez par ElggFile (ou chargez avec `get_entity()`)

Clefs retirées disponibles via `elgg_get_config()`

- `allowed_ajax_views`
- `dataroot_in_settings`
- `externals`
- `externals_map`
- `i18n_loaded_from_cache`
- `language_paths`
- `pagesetupdone`
- `registered_tag_metadata_names`
- `simplecache_enabled_in_settings`
- `translations`
- `viewpath`
- `views`
- `view_path`
- `viewtype`
- `wordblacklist`

Notez également que les plugins ne devraient pas accéder à la variable globale `$CONFIG` à l'exception de `settings.php`.

Fonctions supprimées

- `blog_get_page_content_friends`
- `blog_get_page_content_read`
- `count_unread_messages()`
- `delete_entities()`
- `delete_object_entity()`
- `delete_user_entity()`
- `elgg_get_view_location()`
- `elgg_validate_action_url()`
- `execute_delayed_query()`
- `extend_view()`
- `get_db_error()`
- `get_db_link()`
- `get_entities()`
- `get_entities_from_access_id()`
- `get_entities_from_access_collection()`
- `get_entities_from_annotations()`
- `get_entities_from_metadata()`
- `get_entities_from_metadata_multi()`
- `get_entities_from_relationship()`
- `get_filetype_cloud()`
- `get_library_files()`
- `get_views()`
- `is_ip_in_array()`

- `list_entities()`
- `list_entities_from_annotations()`
- `list_group_search()`
- `list_registered_entities()`
- `list_user_search()`
- `load_plugins()`
- `menu_item()`
- `make_register_object()`
- `mysql_*()` : Elgg *n'utilise plus ext/mysql*
- `remove_blacklist()`
- `search_for_group()`
- `search_for_object()`
- `search_for_site()`
- `search_for_user()`
- `search_list_objects_by_name()`
- `search_list_groups_by_name()`
- `search_list_users_by_name()`
- `set_template_handler()`
- `test_ip()`

Méthodes supprimées

- `ElggCache::set_variable()`
- `ElggCache::get_variable()`
- `ElggData::initialise_attributes()`
- `ElggData::getObjectOwnerGUID()`
- `ElggDiskFilestore::make_directory_root()`
- `ElggDiskFilestore::make_file_matrix()`
- `ElggDiskFilestore::user_file_matrix()`
- `ElggDiskFilestore::mb_str_split()`
- `ElggEntity::clearMetadata()`
- `ElggEntity::clearRelationships()`
- `ElggEntity::clearAnnotations()`
- `ElggEntity::getOwner()`
- `ElggEntity::setContainer()`
- `ElggEntity::getContainer()`
- `ElggEntity::getIcon()`
- `ElggEntity::setIcon()`
- `ElggExtender::getOwner()`
- `ElggFileCache::create_file()`
- `ElggObject::addToSite()` : la fonction parente est toujours disponible dans `ElggEntity`
- `ElggObject::getSites()` : la fonction parente est toujours disponible dans `ElggEntity`
- `ElggSite::getCollections()`
- `ElggUser::addToSite()` : la fonction parente est toujours disponible dans `ElggEntity`
- `ElggUser::getCollections()`
- `ElggUser::getOwner()`
- `ElggUser::getSites()` : la fonction parente est toujours disponible dans `ElggEntity`
- `ElggUser::listFriends()`
- `ElggUser::listGroups()`
- `ElggUser::removeFromSite()` : la fonction parente est toujours disponible dans `ElggEntity`

Les arguments suivants ont également été abandonnés :

- `ElggSite::getMembers() - 2 : $offset`

- `elgg_view_entity_list()` - 3 : `$offset` - 4 : `$limit` - 5 : `$full_view` - 6 : `$list_type_toggle` - 7 : `$pagination`

Hooks des plugins supprimés

- `[display, view]` : Voyez le *nouvel hook plugin*.

Actions supprimées

- `widgets/upgrade`

Vues supprimées

- `forms/admin/plugins/change_state`

Variables des vues supprimées

Pendant le rendu, le système de vue ne les injecte plus dans le champ (scope) :

- `$vars['url']` : remplacé par `elgg_get_site_url()`
- `$vars['user']` : remplacé par `elgg_get_logged_in_user_entity()`
- `$vars['config']` : utilisez `elgg_get_config()` et `elgg_set_config()`
- `$vars['config']` : utilisez `elgg_get_config()` et `elgg_set_config()`

Plusieurs solutions de contournement pour des vues très anciennes ne sont plus effectuées. Effectuez ces modifications :

- Définissez `$vars['full_view']` au lieu de `$vars['full']`.
- Définissez `$vars['name']` au lieu de `$vars['internalname']`.
- Définissez `$vars['id']` au lieu de `$vars['internalid']`.

Bibliothèques supprimées

- `elgg:markdown` : Elgg ne fournit plus d'implémentation de balisage (markdown). Vous devez fournir la vôtre.

Spécifier des Vues via les propriétés

La métadonnée `$entity->view` ne spécifie plus la vue utilisée pour l'affichage dans `elgg_view_entity()`.

De même, la propriété `$annotation->view` n'a plus d'effet dans `elgg_view_annotation()`.

Le type de vue est statique après l'appel initial `elgg_get_viewtype()`

`elgg_set_viewtype()` doit être utilisé pour définir le type de vue au moment de l'exécution. Bien que Elgg vérifie toujours l'entrée `view` et `$CONFIG->view` initialement, cela n'est fait qu'une fois par requête.

Dépréciation

Il est déprécié de lire ou d'écrire les clef de métadonnées commençant par les `filestore::` sur les objets `ElggFile`. Dans Elgg 3.0, ces métadonnées seront supprimées si elles indiquent le chemin d'accès racine des données actuel, aussi peu d'objets de fichiers devraient l'avoir. Les plugins ne doivent utiliser `ElggFile::setFilestore` que si les fichiers doivent être stockés dans un emplacement personnalisé.

Note : Ce n'est pas la seule dépréciation dans Elgg 2.0. Les développeurs de plugins devraient regarder leurs journaux d'erreurs de site.

De 1.10 à 1.11

Contenus

— *Mise en surbrillance des commentaires*

Mise en surbrillance des commentaires

Si votre thème utilise le fichier `views/default/css/elements/components.php`, vous devez y ajouter les définitions de style suivantes pour activer la mise en surbrillance pour les commentaires et les réponses de discussion :

```
.elgg-comments .elgg-state-highlight {
    -webkit-animation: comment-highlight 5s;
    animation: comment-highlight 5s;
}
@-webkit-keyframes comment-highlight {
    from {background: #dff2ff;}
    to {background: white;}
}
@keyframes comment-highlight {
    from {background: #dff2ff;}
    to {background: white;}
}
```

De 1.9 à 1.10

Contenus

— *Chargements de fichier*

Chargements de fichier

Si votre plugin utilise un bout de code copié à partir de l'action `file/upload` pour corriger les types mime détectés pour les formats zippés Microsoft, ce bout de code peut maintenant être supprimé en toute sécurité.

Si votre action de téléchargement effectue d'autres manipulations sur le type mime détecté et les types simples, il est recommandé de faire usage des hooks plugin disponibles :

- `'mime_type', 'file'` pour filtrer les types mime détectés
- `'simple_type', 'file'` pour filtrer les types simples analysés

De 1.8 à 1.9

Contenus

- *Le fichier manifest.xml*
- *\$CONFIG et \$vars["config"]*
- *Fichiers de langues*
- *Notifications*
- *Ajout d'éléments à la liste des Activités*
- *Gestionnaires d'URL des entités*
- *Web services*

Dans les exemples, nous mettons à niveau un plugin imaginaire Photos.

Seuls les changements de clefs sont inclus. Par exemple, certaines des fonctions dépréciées ne sont pas mentionnées ici séparément.

Chaque section comprendra des informations sur la compatibilité arrière avec Elgg 1.8.

Le fichier manifest.xml

Aucun changement n'est nécessaire si votre plugin est compatible avec 1.8.

Il est toutefois recommandé d'ajouter l'étiquette `<id>`. Sa valeur devrait être le nom du répertoire du plugin à l'intérieur du répertoire `mod/`.

Si vous effectuez des modifications qui cassent la compatibilité arrière (BC pour Backward Compatibility), vous devez mettre à jour les versions du plugin et de Elgg requises.

Exemple d'ancienne version (raccourcie) :

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>Photos</name>
  <author>John Doe</author>
  <version>1.0</version>
  <description>Adds possibility to upload photos and arrange them into albums.</
description>
  <requires>
    <type>elgg_release</type>
    <version>1.8</version>
  </requires>
</plugin_manifest>
```


Exemple de nouvelle version (raccourcie) :

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin_manifest xmlns="http://www.elgg.org/plugin_manifest/1.8">
  <name>Photos</name>
  <id>photos</id>
  <author>John Doe</author>
  <version>2.0</version>
  <description>Adds possibility to upload photos and arrange them into albums.</
  <description>
  <requires>
    <type>elgg_release</type>
    <version>1.9</version>
  </requires>
</plugin_manifest>
```

\$CONFIG et \$vars["config"]

La variable mondiale \$CONFIG et le paramètre \$vars ['config'] ont été dépréciés. Ils devraient être remplacés par la fonction elgg_get_config().

Exemple d'ancien code :

```
// Using the global $CONFIG variable:
global $CONFIG;
$plugins_path = $CONFIG->plugins_path

// Using the $vars view parameter:
$plugins_path = $vars['plugins_path'];
```

Exemple de nouveau code :

```
$plugins_path = elgg_get_config('plugins_path');
```

Note : Compatible avec 1.8

Note : Voyez comment le plugin community_plugins a été mis à jour : https://github.com/Elgg/community_plugins/commit/f233999bbd1478a200ee783679c2e2897c9a0483

Fichiers de langues

Dans Elgg 1.8, les fichiers linguistiques devaient utiliser la fonction add_translation(). En 1.9, il suffit de simplement renvoyer le tableau qui a été précédemment passé à la fonction en tant que paramètre. Le noyau Elgg utilisera le nom du fichier (par exemple fr.php) pour déterminer quelle langue contient le fichier.

Exemple d'ancienne manière de faire dans languages/en.php :

```
$english = array(
  'photos:all' => 'All photos',
```

(suite sur la page suivante)

(suite de la page précédente)

```
);
add_translation('en', $english);
```

Exemple de nouvelle manière de faire :

```
return array(
    'photos:all' => 'All photos',
);
```

Avertissement : Non compatible avec 1.8

Notifications

L'un des plus grands changements dans Elgg 1.9 est le système de notifications. Le nouveau système offre une manière plus flexible et évolutive d'envoyer des notifications.

Exemple d'ancienne manière de faire :

```
function photos_init() {
    // Tell core that we want to send notifications about new photos
    register_notification_object('object', 'photo', elgg_echo('photo:new'));

    // Register a handler that creates the notification message
    elgg_register_plugin_hook_handler('notify:entity:message', 'object', 'photos_notify_
↵message');
}

/**
 * Set the notification message body
 *
 * @param string $hook    Hook name
 * @param string $type    Hook type
 * @param string $message The current message body
 * @param array  $params  Parameters about the photo
 * @return string
 */
function photos_notify_message($hook, $type, $message, $params) {
    $entity = $params['entity'];
    $to_entity = $params['to_entity'];
    $method = $params['method'];
    if (elgg_instanceof($entity, 'object', 'photo')) {
        $descr = $entity->excerpt;
        $title = $entity->title;
        $owner = $entity->getOwnerEntity();
        return elgg_echo('photos:notification', array(
            $owner->name,
            $title,
            $descr,
            $entity->getURL()
        ));
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

    }
    return null;
}

```

Exemple de nouvelle manière de faire :

```

function photos_init() {
    elgg_register_notification_event('object', 'photo', array('create'));
    elgg_register_plugin_hook_handler('prepare', 'notification:publish:object:photo',
    ↪ 'photos_prepare_notification');
}

/**
 * Prepare a notification message about a new photo
 *
 * @param string           $hook           Hook name
 * @param string           $type           Hook type
 * @param Elgg_Notifications_Notification $notification The notification to prepare
 * @param array            $params         Hook parameters
 * @return Elgg_Notifications_Notification
 */
function photos_prepare_notification($hook, $type, $notification, $params) {
    $entity = $params['event']->getObject();
    $owner = $params['event']->getActor();
    $recipient = $params['recipient'];
    $language = $params['language'];
    $method = $params['method'];

    // Title for the notification
    $notification->subject = elgg_echo('photos:notify:subject', array($entity->title),
    ↪ $language);

    // Message body for the notification
    $notification->body = elgg_echo('photos:notify:body', array(
        $owner->name,
        $entity->title,
        $entity->getExcerpt(),
        $entity->getURL()
    ), $language);

    // The summary text is used e.g. by the site_notifications plugin
    $notification->summary = elgg_echo('photos:notify:summary', array($entity->title),
    ↪ $language);

    return $notification;
}

```

Avertissement : Non compatible avec 1.8

Note : Voyez comment le plugin community_plugins a été mis à jour pour utiliser le nouveau système : <https://github.com>.

[com/Elgg/community_plugins/commit/bfa356cfe8fb99ebbca4109a1b8a1383b70ff123](https://github.com/Elgg/community_plugins/commit/bfa356cfe8fb99ebbca4109a1b8a1383b70ff123)

Les notifications peuvent aussi être envoyées par la fonction `notify_user()`.

Il a toutefois été mis à jour pour prendre en charge trois nouveaux paramètres optionnels passés à l'intérieur d'un tableau comme cinquième paramètre.

Les paramètres donnent aux plugins de notification plus de contrôle sur les notifications, de sorte qu'ils doivent être inclus dans la mesure du possible. Par exemple, le plugin `site_notifications` ne fonctionnera pas correctement si les paramètres sont manquants.

Paramètres :

- **object** L'objet pour lequel nous envoyons une notification (par ex. `ElggEntity` ou `ElggAnnotation`). Ceci est nécessaire pour que les plugins de notification puissent fournir un lien vers l'objet.
- **action** Chaîne qui décrit l'action qui a déclenché la notification (par ex. « create », « update », etc.).
- **summary** Chaîne qui contient un résumé de la notification. (Elle devrait être plus informative que le sujet de notification, mais moins informative que le corps de la notification.)

Exemple d'ancienne manière de faire :

```
// Notify $owner that $user has added a $rating to an $entity created by him

$subject = elgg_echo('rating:notify:subject');
$body = elgg_echo('rating:notify:body', array(
    $owner->name,
    $user->name,
    $entity->title,
    $entity->getURL(),
));

notify_user($owner->guid,
            $user->guid,
            $subject,
            $body
        );
```

Exemple de nouvelle manière de faire :

```
// Notify $owner that $user has added a $rating to an $entity created by him

$subject = elgg_echo('rating:notify:subject');
$summary = elgg_echo('rating:notify:summary', array($entity->title));
$body = elgg_echo('rating:notify:body', array(
    $owner->name,
    $user->name,
    $entity->title,
    $entity->getURL(),
));

$params = array(
    'object' => $rating,
    'action' => 'create',
    'summary' => $summary,
);

notify_user($owner->guid,
```

(suite sur la page suivante)

(suite de la page précédente)

```

        $user->guid,
        $subject,
        $body,
        $params
    );

```

Note : Compatible avec 1.8

Ajout d'éléments à la liste des Activités

```
add_to_river('river/object/photo/create', 'create', $user_guid, $photo_guid);
```

```

elgg_create_river_item(array(
    'view' => 'river/object/photo/create',
    'action_type' => 'create',
    'subject_guid' => $user_guid,
    'object_guid' => $photo_guid,
));

```

Vous pouvez également ajouter le paramètre facultatif `target_guid` qui indique la cible de l'action create.

Si la photo avait été ajoutée par exemple dans un album photo, nous pourrions l'ajouter en passant aussi :

```
'target_guid' => $album_guid,
```

Avertissement : Non compatible avec 1.8

Gestionnaires d'URL des entités

La fonction `elgg_register_entity_url_handler()` a été dépréciée. Dans 1.9, vous devez utiliser le hook plugin `'entity:url', 'object'`.

Exemple d'ancienne manière de faire :

```

/**
 * Initialize the photo plugin
 */
my_plugin_init() {
    elgg_register_entity_url_handler('object', 'photo', 'photo_url_handler');
}

/**
 * Returns the URL from a photo entity
 *
 * @param ElggEntity $entity
 * @return string
 */

```

(suite sur la page suivante)

(suite de la page précédente)

```
function photo_url_handler($entity) {  
    return "photo/view/{$entity->guid}";  
}
```

Exemple de nouvelle manière de faire :

```
/**  
 * Initialize the photo plugin  
 */  
my_plugin_init() {  
    elgg_register_plugin_hook_handler('entity:url', 'object', 'photo_url_handler');  
}  
  
/**  
 * Returns the URL from a photo entity  
 *  
 * @param string $hook 'entity:url'  
 * @param string $type 'object'  
 * @param string $url The current URL  
 * @param array $params Hook parameters  
 * @return string  
 */  
function photo_url_handler($hook, $type, $url, $params) {  
    $entity = $params['entity'];  
  
    // Check that the entity is a photo object  
    if ($entity->getSubtype() !== 'photo') {  
        // This is not a photo object, so there's no need to go further  
        return;  
    }  
  
    return "photo/view/{$entity->guid}";  
}
```

Avvertissement : Non compatible avec 1.8

Web services

Dans Elgg 1.8, l'API des services Web a été incluse dans le noyau et les méthodes ont été exposées à l'aide de `expose_function()`. Pour activer la même fonctionnalité pour Elgg 1.9, activez le plugin `Web services 1.9` et remplacez tous les appels à `expose_function()` par `elgg_ws_expose_function()`.

1.7 vers 1.8

Contenus

- *Mettre à niveau le cœur*
- *Mettre à niveau les plugins*

Elgg 1.8 est le plus grand bond en avant dans le développement d'Elgg depuis la version 1.0. Pour cette raison, cela demande plus de travail pour mettre à jour le noyau et les plugins qu'avec les mises à niveau précédentes. Il y a eu un petit nombre de changements d'API et, suivant notre pratique standard, les méthodes que nous avons dépréciées ont été mises à jour pour travailler avec la nouvelle API. Les plus grands changements sont dans la normalisation des plugins et dans le système de vues.

Mettre à niveau le cœur

Supprimez les répertoire du noyau suivants (même niveau que `_graphics` et `engine`) :

- `_css`
- `account`
- `admin`
- `dashboard`
- `entities`
- `friends`
- `search`
- `settings`
- `simplecache`
- `views`

Avertissement : Si vous ne supprimez pas ces répertoires avant une mise à niveau, vous allez avoir des ennuis !

Mettre à niveau les plugins

Utilisez le routage standard avec vos gestionnaires de pages

- Tout : `/page_handler/all`
- Publications de l'utilisateur : `/page_handler/owner/ :username`
- Publications des contacts de l'utilisateur : `/page_handler/friends/ :username`
- Entité seule : `/page_handler/view/ :guid/ :title`
- Ajout : `/page_handler/add/ :container_guid`
- Modification : `/page_handler/edit/ :guid`
- Liste du groupe : `/page_handler/group/ :guid/all`

Inclure des scripts de gestionnaire de page à partir du gestionnaire de page

Presque tous les gestionnaires de page doivent avoir un script de gestionnaire de page. (Exemple : `bookmarks/all => mod/bookmarks/pages/bookmarks/all.php`)

- Appelez `set_input()` pour les guids d'entité dans le gestionnaire de page et utilisez `get_input()` dans les scripts du gestionnaire de page.
- Appelez `gatekeeper()` et `admin_gatekeeper()` dans la fonction du gestionnaire de page si nécessaire.
- L'URL d'un groupe doit utiliser le script `pages/:handler/owner.php`.
- Les gestionnaires de pages ne devraient pas contenir de HTML.
- Mettez à jour les URL dans tout le plugin. (N'oubliez pas d'enlever `/pg/` !)

Utilisez les gestionnaires de pages standardisés et les scripts

- Stockez les scripts du gestionnaire de page dans `mod/:plugin/pages/:page_handler/:page_name.php`
- Utilisez la disposition de page de contenu dans les scripts du gestionnaire de pages :

```
$content = elgg_view_layout('content', $options);
```

- Les scripts de gestionnaires de pages ne devraient pas contenir de HTML.
- Appelez `elgg_push_breadcrumb()` dans les scripts du gestionnaire de pages.
- Pas besoin de définir le propriétaire de la page si les URLs sont au format standard.
- Pour le contenu du groupe, vérifiez le conteneur `container_guid` en utilisant `elgg_get_page_owner_entity()`.

La vue `object/:subtype`

- Assurez-vous qu'il existe bien des vues pour `$vars['full_view'] == true` et `$vars['full_view'] == false`. `$vars['full_view']` a remplacé `$vars['full']`.
- Vérifiez l'objet dans `$vars['entity']`. Utilisez `elgg_instance_of()` pour vous assurer qu'il s'agit du type d'entité que vous souhaitez.
- Renvoyez `true` pour court-circuiter la vue si l'entité est manquante ou erronée.
- Utilisez `elgg_view('object/elements/summary', array('entity' => $entity));` et `elgg_view_menu('entity', array('entity' => $entity));` pour faciliter le formatage. Vous devriez utiliser très peu de balisage dans ces vues.

Mettre à jour la structure de l'action

- Fichiers d'action et noms d'action de l'espace de noms (exemple : `mod/blog/actions/blog/save.php => action/blog/save`)
- Utilisez les URLs d'action suivantes :
 - Ajoutez : `action/:plugin/save`
 - Modifiez : `action/:plugin/save`
 - Suppression : `action/:plugin/delete`
- Faites que l'action de suppression accepte `action/:handler/delete?guid=:guid` de sorte que le menu des métadonnées de l'entité ait la bonne URL par défaut.

Mettez à jour les fonctions obsolètes

- Les fonctions dépréciées en 1.7 produiront des erreurs visibles en 1.8.
- Vous pouvez également mettre à jour les fonctions dépréciées dans 1.8.
 - De nombreuses fonctions d’enregistrement ont simplement ajouté un préfixe `elgg_` pour la cohérence, et devraient être faciles à mettre à jour.
 - Voyez `/engine/lib/deprecated-1.8.php` pour la liste complète.
 - Vous pouvez définir le niveau de débogage sur “warning” pour obtenir des rappels visuels des fonctions obsolètes.

Mettez à jour les vues widget

Voyez les widgets des plugins blog ou file pour des exemples.

Mettez à jour le module de profil de groupe

Utilisez les plugins blog ou file pour des exemples. Cela vous aidera à pouvoir appliquer un thème à votre plugin avec le nouveau framework CSS.

Mettre à jour les formulaires

- Déplacez les corps de formulaire vers la vue `forms/:action` pour utiliser le nouveau `elgg_view_form` d’Evan.
- Utilisez les vues d’entrée dans les corps de formulaire plutôt que du html. Cela aide à utiliser des thèmes et permet de préparer l’avenir.
- Ajoutez une fonction qui prépare le formulaire (voyez `mod/file/lib/file.php` pour un exemple)
- Rendre vos formulaires persistants (voir l’action de téléchargement du plugin file et la fonction de préparation du formulaire).

L’API des formulaires est discutée plus en détail dans *Formulaires + Actions*.

Nettoyez le CSS/HTML

Nous avons ajouté de nombreux modèles CSS au fichier CSS du noyau (modules, bloc avec une image, primitives d’espace). Nous vous encourageons à utiliser ces modèles et ces classes dans la mesure du possible. Faire cela devrait :

1. Réduire les coûts de maintenance, car vous pouvez supprimer la plupart des CSS personnalisés.
2. Rendre votre plugin plus compatible avec les thèmes de la communauté.

Recherchez des modèles qui peuvent être déplacés dans le noyau si vous avez besoin de CSS significatif.

Nous utilisons des traits d’union plutôt que des soulignements dans les classes/ids et vous encourageons à faire de même pour maintenir la cohérence.

Si vous avez besoin de votre propre CSS, vous devez utiliser votre propre espace de noms, plutôt que `elgg-`.

Mettre à jour manifest.xml

- Utilisez <http://el.gg/manifest17to18> pour automatiser cette étape.
- N'utilisez pas la catégorie `bundled` avec vos plugins. C'est réservé aux plugins distribués avec Elgg.

Mettez à jour les paramètres et les vues des paramètres utilisateur

- La vue pour les paramètres est maintenant `plugins/:plugin/settings` (auparavant `settings/:plugin/edit`).
- La vue pour les paramètres utilisateur est maintenant `plugins/:plugin/usersettings` (auparavant `usersettings/:plugin/edit`).

3.7.2 FAQs et autres Aides au dépannage

Ci-dessous quelques questions fréquemment posées à propos de Elgg.

Contenus

- *Autres types de fichiers*
 - « Le plugin ne peut pas démarrer et a été désactivé » (« *Plugin cannot start and has been deactivated* ») ou « Ce plugin est invalide » (« *This plugin is invalid* »)
- *Page Blanche (WSOD = White Screen Of Death, l'Ecran Blanc De la Mort)*
- *Page non trouvée*
- *Erreur de correspondance des jetons de connexion (« login token mismatch »)*
- *Il manque les champs `__token` ou `__ts` dans le formulaire. Veuillez recharger la page pour continuer*
- *Mode de maintenance*
- *Email manquant*
- *Journaux du serveur*
- *Comment fonctionne l'inscription ?*
- *Validation de l'utilisateur*
- *Ajouter manuellement un utilisateur*
- *Je suis en train de créer ou viens d'installer un nouveau thème, mais les images ou d'autres éléments ne fonctionnent pas*
- *Modifier les champs du profil*
- *Modifier l'inscription*
- *Comment puis-je modifier les paramètres PHP en utilisant `.htaccess` ?*
- *Connexion HTTPS activée par erreur*
- *Utiliser un site de test*
- *500 - Internal Server Error*
- *Lorsque je télécharge une photo ou que je modifie ma photo de profil, j'obtiens un écran blanc*
- *CSS manquant*
- *Devrais-je modifier la base de données manuellement ?*
- *Problème de connexion d'Internet Explorer (IE)*
- *Les emails ne supportent pas les caractères non-latins*
- *Durée de session*
- *Un fichier n'a pas de propriétaire*
- *Pas d'image*
- *Avertissements d'obsolescence*
- *JavaScript ne fonctionne pas*
- *Les adresses IP dans les journaux sont incorrectes*

- *Sécurité*
 - *Est-ce upgrade.php pose des soucis de sécurité ?*
 - *Devrais-je supprimer install.php ?*
 - *Filtrage*
- *Développement*
 - *Que dois-je utiliser pour modifier le code php ?*
 - *Je n'aime pas certaines traductions dans Elgg. Comment puis-je les changer ?*
 - *Comment puis-je trouver le code qui fait X ?*
 - *Mode de débogage*
 - *Quels événements sont déclenchés sur chaque chargement de page ?*
 - *Copier un plugin*

Autres types de fichiers

Voir aussi :

Trouver de l'aide

« Le plugin ne peut pas démarrer et a été désactivé » (« Plugin cannot start and has been deactivated ») ou « Ce plugin est invalide » (« This plugin is invalid »)

Cette erreur est habituellement accompagnée par plus de détails qui expliquent pourquoi le plugin est invalide. Ceci est généralement causé par un plugin mal installé.

Si vous installez un plugin appelé « test », il y aura un répertoire test dans le dossier mod. Dans ce dossier test il doit y avoir un fichier manifest.xml /mod/test/manifest.xml.

Si ce fichier n'existe pas, cela peut être causé par :

- l'installation d'un plugin dans le mauvais répertoire
- la création d'un répertoire dans /mod qui ne contient pas un plugin
- un mauvais transfert FTP
- l'extraction d'un plugin dans un niveau de répertoire supplémentaire (myplugin.zip est extrait dans myplugin/myplugin)

Si vous utilisez un hôte de type Unix et que els fichiers existent dans le bon répertoire, vérifiez les permissions. Elgg doit avoir accès en lecture + exécution sur els répertoires.

Page Blanche (WSOD = White Screen Of Death, l'Ecran Blanc De la Mort)

Une page vide, blanche (souvent appelé « écran blanc de la mort » - « white screen of death » ou WSOD) signifie qu'il y a une

- fichier corrompu - essayez de transférer à nouveau le code vers votre serveur
- un appel à un module php qui n'a pas été chargé - ceci peut se produire après que vous ayez installé un plugin qui requiert un module spécifique.
- mauvais plugin - tous les plugins n'ont pas été écrits avec le même niveau de qualité aussi vous devriez faire attention à ceux que vous installez.

Pour trouver l'origine de l'erreur, modifiez le fichier .htaccess pour afficher les erreurs dans le navigateur. Définissez display_errors à 1 et chargez la même page à nouveau. Vous devriez voir les erreurs PHP dans votre navigateur. Modifiez à nouveau ce paramètre une fois que vous avez résolu le problème.

Note : Si vous utilisez le plugin Developer's Tools, allez dans sa page de configuration et assurez-vous que l'option « Afficher les erreurs PHP fatales » (« Display fatal PHP errors ») est bien activé.

Si l'écran blanc est dû à un mauvais plugin, retirez les derniers plugins que vous avez installés en supprimant leurs répertoires puis rechargez la page.

Note : Vous pouvez désactiver temporairement tous les plugins en créant un fichier vide dans `mod/mod_disabled`. Vous pouvez ensuite désactiver le plugin responsable via les outils du panneau d'administration.

Si vous avez un WSOD quand vous effectuez une action, comme vous identifier ou publier un article de blog, mais qu'il n'y a pas de message d'erreur, le plus probable est que ce soit causé par des caractères non-imprimables dans le code du plugin. Vérifiez le plugin et supprimez les espaces vides et nouvelles lignes situés après le tag de fin php (`?>`).

Page non trouvée

Si vous avez récemment installé votre site Elgg, la cause la plus probable d'une erreur de page non trouvée est que `mod_rewrite` n'est pas configuré correctement sur votre serveur. Il y a des informations dans la page de dépannage [Install Troubleshooting](#) sur la manière de résoudre cela. La deuxième cause la plus probable est que l'URL de votre site dans votre base de données est incorrecte.

Si vous exécutez votre site depuis un certain temps et commencez soudainement à obtenir des erreurs de page non trouvées, vous devez vous demander ce qui a changé. Avez-vous ajouté des plugins ? Avez-vous modifié la configuration de votre serveur ?

Pour déboguer une erreur page non trouvée (« page not found ») :

- Confirmez que le lien qui a mené à la page manquante est correct. S'il ne l'est pas, comment ce lien a-t-il été généré ?
- Confirmez que les règles de réécriture du `.htaccess` sont bien prises en compte.

Erreur de correspondance des jetons de connexion (« login token mismatch »)

Si vous devez vous connecter deux fois à votre site et que le message d'erreur après la première tentative indique qu'il y a eu une erreur de décalage de jeton, l'URL dans les paramètres d'Elgg ne correspond pas à l'URL utilisée pour y accéder. La cause la plus fréquente est d'ajouter ou de supprimer le « `www` » lors de l'accès au site. Par exemple, `www.elgg.org` vs `elgg.org`. Cela pose un problème avec la gestion des sessions en raison de la façon dont les navigateurs Web enregistrent les cookies.

Pour régler cela, vous pouvez ajouter des règles de réécriture (« rewrite rules »). Pour rediriger de `www.elgg.org` vers `elgg.org` dans Apache, les règles peuvent ressembler à

```
RewriteCond %{HTTP_HOST} .
RewriteCond %{HTTP_HOST} !^elgg\.org
RewriteRule (.*) http://elgg.org/$1 [R=301,L]
```

Rediriger depuis une adresse non-`www` vers `www` peut ressembler à

```
RewriteCond %{HTTP_HOST} ^elgg\.org
RewriteRule ^(.*)$ http://www.elgg.org/$1 [R=301,L]
```

Si vous ne savez pas comment configurer les règles de réécriture, demandez à votre hébergeur pour plus d'informations.

Il manque les champs __token ou __ts dans le formulaire. Veuillez recharger la page pour continuer

Toutes les actions Elgg requièrent un jeton de sécurité, et cette erreur se produit quand ce jeton est manquant. C'est soit un problème avec la configuration de votre serveur ou avec un plugin tierce-partie.

Si vous rencontrez ce cas lors d'une nouvelle installation, assurez-vous que votre serveur est correctement configuré et que vos règles de réécriture sont correctes. Si vous en faites l'expérience sur une mise à niveau, assurez-vous d'avoir mis à jour vos règles de réécriture dans .htaccess (Apache) ou dans la configuration du serveur.

Si vous rencontrez ce cas, désactivez tous les plugins tiers et réessayez. Les très vieux plugins pour Elgg n'utilisent pas de jeton de sécurité. Si le problème disparaît lorsque les plugins sont désactivés, c'est dû à un plugin qui devrait être mis à jour par son auteur.

Mode de maintenance

Pour mettre temporairement votre site hors-ligne, allez dans Administration -> Utilitaires (Utilities) -> Mode de Maintenance (Maintenance Mode). Complétez le formulaire et cliquez sur Enregistrer pour désactiver votre site pour tout le monde à l'exception des utilisateurs admin.

Email manquant

Si vos utilisateurs signalent que les e-mails de validation ne s'affichent pas, demandez-leur de vérifier leur dossier de spam. Il est possible que les e-mails provenant de votre serveur sont marqués comme spam. Cela dépend de nombreux facteurs : si votre fournisseur d'hébergement a un problème avec les spammeurs, la façon dont votre configuration de messagerie PHP est configurée, quel agent de transport de messagerie votre serveur utilise, ou si votre hébergement limite le nombre de courriels que vous pouvez envoyer en une heure.

Si personne ne reçoit d'email du tout, il est très probable que votre serveur n'est pas configuré correctement pour le courrier électronique. Votre serveur a besoin d'un programme pour envoyer un e-mail (appelé agent de transfert de messagerie - MTA) et PHP doit être configuré pour utiliser le MTA.

Pour vérifier rapidement si PHP et un MTA (Mail Transfer Agent) sont correctement configurés, créez un fichier sur votre serveur avec le contenu suivant :

```
<?php
$address = "your_email@your_host.com";

$subject = 'Test email.';

$body = 'If you can read this, your email is working.';

echo "Attempting to email $address...<br />";

if (mail($address, $subject, $body)) {
    echo 'SUCCESS! PHP successfully delivered email to your MTA. If you don\'t see
    ↳ the email in your inbox in a few minutes, there is a problem with your MTA.';
} else {
    echo 'ERROR! PHP could not deliver email to your MTA. Check that your PHP
    ↳ settings are correct for your MTA and your MTA will deliver email.';
}
```

Assurez-vous de remplacer « `your_email@your_host.com` » par votre adresse e-mail réelle. Prenez soin de garder des guillemets autour d'elle ! Lorsque vous accédez à cette page via votre navigateur Web, il tentera d'envoyer un e-mail de test. Ce test vous permettra de savoir si PHP et votre MTA sont correctement configurés. Si elle échoue - soit vous

obtenez une erreur, soit vous ne recevez jamais l'e-mail - vous aurez besoin de faire plus de recherches et éventuellement de contacter votre fournisseur de services.

La configuration complète des fonctionnalités de messagerie d'un MTA et de PHP dépasse le cadre de cette FAQ, et vous devez rechercher plus de ressources sur Internet à ce sujet. Quelques informations de base sur les paramètres php peuvent être trouvées sur le [PHP's site](#)

Journaux du serveur

Très probablement vous utilisez Apache comme serveur Web. Les avertissements et les erreurs sont écrits dans un journal par le serveur Web et peuvent être utiles pour les problèmes de débogage. Vous verrez généralement deux types de fichiers journaux : les journaux d'accès et les journaux d'erreurs. Les informations de PHP et Elgg sont inscrites dans le journal des erreurs du serveur.

- Linux – Le journal d'erreur est probablement dans `/var/log/httpd` ou `/var/log/apache2`.
- Windows - Il est probablement dans votre répertoire Apache.
- Mac OS - Le journal d'erreur est probablement dans `/var/log/apache2/error_log`

Si vous utilisez un hébergement partagé sans accès ssh, votre fournisseur d'hébergement peut fournir un mécanisme pour obtenir l'accès à vos journaux de serveur. Vous devrez leur poser des questions à ce sujet.

Comment fonctionne l'inscription ?

Avec une installation par défaut, voici comment fonctionne l'inscription :

1. L'utilisateur renseigne le formulaire d'inscription et l'envoie
2. Le compte utilisateur est créé et désactivé jusqu'à sa validation
3. Un email est envoyé à l'utilisateur afin de valider son compte
4. Quand l'utilisateur clique sur le lien, le compte est validé
5. L'utilisateur peut maintenant s'identifier

Les échecs durant ce processus comprennent l'utilisateur entrant une adresse e-mail incorrecte, l'e-mail de validation marqué comme spam, ou un utilisateur qui ne prend jamais la peine de valider le compte.

Validation de l'utilisateur

Par défaut, tous les utilisateurs qui s'inscrivent doivent valider leur compte par e-mail. Si un utilisateur a des difficultés pour valider son compte, vous pouvez valider manuellement les utilisateurs en allant dans Administration -> Utilisateurs -> Non validé.

Vous pouvez supprimer cette exigence en désactivant le plugin User Validation by Email.

Note : La suppression de la validation a certaines conséquences : il n'y a aucun moyen de savoir qu'un utilisateur inscrit à une adresse e-mail fonctionnelle, et cela peut ouvrir le système aux spammeurs.

Ajouter manuellement un utilisateur

Pour ajouter manuellement un utilisateur, dans la partie Administrer, accédez aux Utilisateurs. Là, vous verrez un titre intitulé « Ajouter un nouvel utilisateur ». Après avoir rempli les informations et soumis le formulaire, le nouvel utilisateur recevra un e-mail avec son nom d'utilisateur et son mot de passe, et un rappel pour changer le mot de passe.

Note : Elgg ne force pas l'utilisateur à changer le mot de passe.

Je suis en train de créer ou viens d'installer un nouveau thème, mais les images ou d'autres éléments ne fonctionnent pas

Assurez-vous que le thème est placé en tout dernier sur la liste des plugins.

Effacez le cache de votre navigateur et rechargez la page. Pour alléger la charge sur le serveur, Elgg demande au navigateur de charger rarement le fichier CSS. Un nouveau thème modifiera complètement le fichier CSS et un rafraîchissement devrait inciter le navigateur à demander à nouveau le fichier CSS.

Si vous créez ou modifiez un thème, assurez-vous d'avoir désactivé les caches simples et système. Cela peut être fait en activant le plugin Developer Tools, puis en naviguant vers Administration -> Outils de développement > Paramètres. Une fois que vous êtes satisfait des modifications, activez les caches ou les performances en souffriront.

Modifier les champs du profil

Dans les paramètres d'administration d'Elgg se trouve une page pour remplacer les champs de profil par défaut. Elgg donne par défaut à l'administrateur deux choix :

- Utilisez les champs de profil par défaut
- Remplacez les champs par défaut par un jeu de champs de profil personnalisés

Vous ne pouvez pas ajouter de nouveaux champs de profil aux champs par défaut. L'ajout d'un nouveau champ de profil via l'option remplacer les champs de profil efface ceux par défaut. Avant de laisser les utilisateurs s'inscrire, il est préférable de déterminer quels champs de profil vous voulez, quels types de champ ils devraient être, et l'ordre dans lequel ils devraient apparaître. Vous ne pouvez pas modifier le type de champ ou réordonner ou supprimer les champs après leur création sans effacer l'intégralité du profil.

Plus de flexibilité peut être obtenue grâce à des plugins. Il y a au moins deux plugins sur le site communautaire qui vous permettent d'avoir plus de contrôle sur les champs de profil. Le plugin [Profile Manager](#) est devenu très populaire dans la communauté Elgg. Il vous permet d'ajouter de nouveaux champs de profil quand vous le souhaitez, de modifier leur ordre, de regrouper les champs de profil et de les ajouter à l'inscription.

Modifier l'inscription

Le processus d'inscription peut être modifié par un plugin. Tout ce qui concerne l'inscription peut être modifié : l'apparence du formulaire, des champs d'inscription différents, des validations additionnelles des champs, des étapes additionnelles et ainsi de suite. Ces types de changements demandent quelques connaissances de base en HTML, CSS, PHP.

Une autre option est d'utiliser le plugin Profile Manager`_ qui vous permet d'ajouter des champs à la fois aux profils des utilisateurs et au formulaire d'inscription.

Créer le squelette du plugin *Squelette du plugin*

Modifier l'apparence de l'inscription Surchargez la vue account/forms/register

Modifiez le gestionnaire de l'action d'inscription Vous pouvez écrire votre propre action pour créer le compte utilisateur

Comment puis-je modifier les paramètres PHP en utilisant .htaccess ?

Vous pouvez modifier les paramètres php dans votre fichier `.htaccess`. Cela est particulièrement vrai si votre fournisseur d'hébergement ne vous donne pas accès au fichier `php.ini` du serveur. Les variables peuvent être liées aux limites de taille du téléchargement de fichiers, à la sécurité, à la longueur de la session ou à n'importe quel nombre d'autres attributs php. Pour des exemples sur comment faire cela, voyez la [documentation PHP](#) à ce sujet.

Connexion HTTPS activée par erreur

Si vous avez activé la connexion HTTPS mais que vous n'avez pas configuré SSL, vous êtes maintenant bloqué hors de votre installation Elgg. Pour désactiver ce paramètre de configuration, vous devrez modifier votre base de données. Utilisez un outil comme phpMyAdmin pour afficher votre base de données. Sélectionnez la table `config` et supprimez la ligne qui a le nom `https_login`.

Utiliser un site de test

Il est recommandé de toujours tester les nouvelles versions ou les nouveaux plugins sur un site de test avant de les exécuter sur un site de production (un site avec des utilisateurs réels). La façon la plus simple de le faire est de maintenir une installation séparée d'Elgg avec des comptes factices. Lors des tests, il est important d'utiliser des comptes factices qui ne sont pas des administrateurs pour tester ce que vos utilisateurs verront.

Un test plus réaliste consiste à refléter le contenu de votre site de production sur votre site de test. Suivez les instructions pour [dupliquer un site](#). Assurez-vous ensuite d'empêcher l'envoi d'e-mails à vos utilisateurs. Vous pouvez écrire un petit plugin qui redirige tous les e-mails vers votre propre compte (soyez conscient des plugins qui comprennent leur propre code d'envoi d'email personnalisé, car vous devrez modifier ces plugins). Une fois cela fait, vous pouvez afficher tout le contenu pour s'assurer que la mise à niveau ou le nouveau plugin fonctionne comme vous le souhaitez et ne casse rien. Si ce processus vous semble trop lourd, veuillez vous en tenir à l'exécution d'un site de test simple.

500 - Internal Server Error

Qu'est-ce que c'est ?

Une erreur interne du serveur ou **500 - Internal Server Error** signifie que le serveur web a rencontré un problème en servant une requête.

Voir aussi :

[La page Wikipédia sur les codes de statut HTTP](#)

Causes possibles

Configuraiton du serveur web La cause la plus fréquente pour cela est un serveur mal configuré. Si vous avez modifié le fichier `.htaccess` et ajouté quelque chose d'incorrect, Apache renverra une erreur 500.

Permissions sur les fichiers Il peut également s'agir d'un problème de permissions sur un fichier. Apache doit pouvoir lire les fichiers d'Elgg. L'utilisation des autorisations 755 sur les répertoires et 644 sur les fichiers permettra à Apache de lire les fichiers.

Lorsque je télécharge une photo ou que je modifie ma photo de profil, j'obtiens un écran blanc

Il est fort probable que la bibliothèque PHP GD n'est pas installée ou configurée correctement. Vous pourriez avoir besoin de l'aide de l'administrateur de votre serveur.

CSS manquant

Mauvaise URL

Parfois, les gens installent Elgg de sorte que l'URL de base est `localhost` et essaient ensuite d'afficher le site à l'aide d'un nom d'hôte. Dans ce cas, le navigateur ne pourra pas charger le fichier CSS. Essayez d'afficher la source de la page Web et de copier le lien du fichier CSS. Collez-le dans votre navigateur. Si vous obtenez une erreur 404, il est probable que c'est votre problème. Vous devrez modifier l'URL de base de votre site.

Erreur de syntaxe

Elgg stocke ses CSS sous forme de code PHP pour fournir flexibilité et puissance. S'il y a une erreur de syntaxe, le fichier CSS servi au navigateur peut être vide. La désactivation des plugins non groupés est la première étape recommandée.

Erreurs de règles de réécriture d'URL (Rewrite rules)

Un mauvais fichier `.htaccess` peut également entraîner une erreur 404 lors de la requête du fichier CSS. Cela pourrait se produire lors d'une mise à niveau si vous oubliez de mettre également à niveau le fichier `.htaccess`.

Devrais-je modifier la base de données manuellement ?

Avertissement : Non, vous ne devriez jamais modifier manuellement la base de données !

Est-ce que modifier manuellement la base de données va casser mon site ?

Oui.

Est-ce que je peux ajouter des champs supplémentaires aux tables dans la base de données ?

(AKA : Je ne comprends pas le modèle de données de Elgg *data model* donc je vais ajouter des colonnes. Est-ce que vous allez m'aider?)

Non, c'est une mauvaise idée. Étudiez le modèle de données *data model* et vous verrez qu'à moins que ce soit une installation très spécifique et fortement personnalisée, vous pouvez faire tout ce dont vous avez besoin avec le modèle de données actuel d'Elgg.

Je veux supprimer des utilisateurs. Est-ce que je ne peux pas les supprimer directement de la table `elgg_entities` ?

Non, cela va corrompre votre base de données. Supprimez-les via le site.

Je veux supprimer du spam. Est-ce que je ne peux pas juste le rechercher et le supprimer dans la table `elgg_entities` ?

Non, cela va corrompre votre base de données. Supprimez-le via le site.

Quelqu'un sur le site communautaire m'a dit de modifier manuellement la base de données. Est-ce que je devrais le faire ?

Qui étais-ce ? Est-ce quelqu'un d'expérimenté avec Elgg, comme l'un des développeurs du noyau ou un auteur de plugins bien connu ? Est-ce qu'il ou elle vous a donné des instructions claires sur quoi modifier ? Si vous ne savez pas qui c'est, ou si vous ne comprenez pas ou n'êtes pas à l'aise avec ces instructions, ne modifiez pas la base de données manuellement.

Je connais PHP et MySQL et ai une raison légitime de modifier la base de données. Est-ce OK de modifier manuellement la base de données ?

Assurez-vous d'abord de bien comprendre le modèle de données d'Elgg *data model* et le schéma. Faites un backup, modifiez avec soin, puis testez abondamment.

Problème de connexion d'Internet Explorer (IE)

URL canonique

IE n'aime pas travailler avec des sites qui utilisent à la fois <http://exemple.org> et <http://www.exemple.org>. Il stocke plusieurs cookies et cela cause des problèmes. Il est préférable d'utiliser une seule URL de base. Pour plus d'informations sur la façon de le faire, voyez l'erreur « Login token mismatch error » (erreur de correspondance des jetons de connexion).

Cadre Chrome

Utiliser le cadre Chrome au sein de IE peut rompre le processus d'authentification.

Les emails ne supportent pas les caractères non-latins

Pour prendre en charge les caractères non-latins, (tels que les caractères cyrilliques ou chinois) Elgg a besoin que le support des chaînes de caractère multibyte (*»multibyte string support«*) soit compilé avec PHP.

Sur de nombreuses installations (par exemple Debian & Ubuntu), cela est activé par défaut. Si ce n'est pas le cas, vous devez l'activer (ou recompiler PHP pour l'inclure). Pour vérifier si votre serveur prend en charge les chaînes multioctets (multibyte), vérifiez phpinfo.

Durée de session

La durée de la session est contrôlée par votre configuration php. Vous devrez d'abord localiser votre fichier `php.ini`. Dans ce fichier vous trouverez plusieurs variables de session. Une liste complète et ce qu'elles font peuvent être trouvés dans le [manuel de php](#).

Un fichier n'a pas de propriétaire

Il y a trois causes à cette erreur. Vous pourriez avoir une entité dans votre base de données qui a un `owner_guid` de 0. Cela devrait être extrêmement rare et ne peut se produire que si votre base de données/serveur plante lors d'une opération d'écriture.

La deuxième cause serait une entité où le propriétaire n'existe plus. Cela peut se produire si un plugin est désactivé qui a été impliqué dans la création de l'entité, puis le propriétaire est supprimé, mais l'opération de suppression a échoué (parce que le plugin est désactivé). Si vous pouvez comprendre quelle entité est à l'origine de cette situation, regardez dans votre table `entities` et modifiez le `owner_guid` pour le vôtre, puis vous pouvez supprimer l'entité via Elgg.

Avertissement : Lisez la section « Dois-je modifier la base de données manuellement ? ». Soyez très prudent lors de l'édition directe de la base de données. Cela peut briser votre site. Faites **toujours** une sauvegarde avant de faire cela.

Corrections

Le plugin [Database Validator](#) vérifiera votre base de données pour ces causes et vous fournira une option pour les corriger. Assurez-vous de sauvegarder la base de données avant d'essayer l'option de correction.

Pas d'image

Si des images de profil, des images de groupe ou d'autres fichiers ont cessé de fonctionner sur votre site, il est probable qu'il s'agit d'une mauvaise configuration, surtout si vous avez migré vers un nouveau serveur.

Ce sont les erreurs de configuration les plus courantes qui font arrêter le fonctionnement des images et d'autres fichiers.

Mauvais chemin pour le répertoire de données

Assurez-vous que le chemin d'accès du répertoire de données est correct dans la zone d'administration du site. Il devrait avoir un slash à la fin.

Mauvaises permissions sur le répertoire de données

Vérifiez les permissions du répertoire de données. Le répertoire de données doit être lisible et inscriptible par l'utilisateur du serveur web.

Installation migrée avec un nouvel emplacement du répertoire de données

Si vous avez migré une installation et que vous devez modifier votre chemin d'accès au répertoire de données, assurez-vous de mettre à jour le SQL pour l'emplacement des données tel que documenté dans les [Dupliquer une installation](#) instructions.

Avertissements d'obsolescence

Si vous voyez de nombreux avertissements de dépréciation qui disent des choses comme

Deprecated in 1.7: `extend_view()` was deprecated by `elgg_extend_view()`!

alors vous utilisez un plugin qui a été écrit pour une ancienne version d'Elgg. Cela signifie que le plugin utilise des fonctions qui sont prévues pour être supprimées dans une future version d'Elgg. Vous pouvez demander au développeur de plugin si le plugin sera mis à jour ou vous pouvez mettre à jour le plugin vous-même. Si ni l'un ni l'autre de ceux-ci sont susceptibles de se produire, vous ne devriez pas utiliser ce plugin.

JavaScript ne fonctionne pas

Si le menu utilisateur qui apparaît au survol cesse de fonctionner ou que vous ne pouvez pas supprimer les messages système, cela signifie que JavaScript est cassé sur votre site. Ceci est généralement dû à un plugin ayant un mauvais code JavaScript. Vous devriez trouver le plugin à l'origine du problème et le désactiver. Vous pouvez le faire en désactivant les plugins non groupés un par un jusqu'à ce que le problème disparaisse. Une autre approche consiste à désactiver tous les plugins non groupés, puis à les activer un par un jusqu'à ce que le problème se reproduise.

La plupart des navigateurs Web vous donneront un indice quant à ce qui casse le code JavaScript. Ils disposent souvent d'une console pour les erreurs JavaScript ou un mode avancé pour afficher les erreurs. Une fois que vous voyez le message d'erreur, il est plus facile de localiser le problème.

Les adresses IP dans les journaux sont incorrectes

Si votre installation Elgg est derrière un serveur proxy ou un équilibreur de charge, les adresses IP enregistrées dans les journaux système (System Log) peuvent être erronées. Elles pourraient n'indiquer que les adresses IP du serveur proxy.

Afin de résoudre ceci, vous pouvez configurer les adresses IP du serveur proxy comme adresses IP de confiance, ce qui permettra au système d'accéder aux bonnes IP de vos utilisateurs.

Dans le fichier `settings.php`, vous pouvez configurer des paramètres pour `$CONFIG->http_request_trusted_proxy_ips` et `$CONFIG->http_request_trusted_proxy_headers`. Regardez dans le fichier `settings.php` pour plus d'information.

Sécurité

Est-ce `upgrade.php` pose des soucis de sécurité ?

`Upgrade.php` est un fichier utilisé pour exécuter des mises à niveau du code et de la base de données. Il est placé à la racine de l'installation et il n'y a pas besoin d'un compte identifié pour y accéder. Sur un site totalement mis à niveau, l'exécution du fichier va seulement réinitialiser les caches et quitter, de sorte que ceci ne constitue pas un problème de sécurité.

SI cela vous préoccupe tout de même, vous pouvez supprimer, déplacer ou modifier les permissions sur ce fichier jusqu'à ce que vous ayez besoin d'effectuer une mise à niveau.

Devrais-je supprimer install.php ?

Ce fichier est utilisé pour installer Elgg et n'a pas besoin d'être supprimé. Le fichier vérifie si Elgg est déjà installé et redirige l'utilisateur vers la première page si c'est le cas.

Filtrage

Le filtrage est utilisé dans Elgg pour rendre les [attaques XSS](#) plus difficiles. L'objectif du filtrage est de supprimer les JavaScript et autres entrées dangereuses envoyées par les utilisateurs.

Le filtrage est effectué par l'intermédiaire de la fonction `filter_tags()`. Cette fonction prend une chaîne et renvoie une chaîne filtrée. Elle déclenche un `:ref:hook plugin <design/events#plugin-hooks>` validate, input`. Par défaut Elgg est fourni avec le code de filtrage `htmlawed` sous la forme d'un plugin. Les développeurs peuvent intégrer tout autre code de filtrage supplémentaire ou en remplacement sous forme de plugin.

La fonction `filter_tags()` est appelée pour chaque saisie utilisateur dès lors que la saisie est obtenue à travers un appel à `get_input()`. Si pour quelque raison un développeur souhaite ne pas appliquer le filtrage par défaut sur certaines saisies utilisateur, la fonction `get_input()` a un paramètre pour désactiver le filtrage.

Développement

Que dois-je utiliser pour modifier le code php ?

Il existe deux options principales : l'éditeur de texte ou *environnement de développement intégré* (IDE).

Éditeur de texte

Si vous débutez dans le développement de logiciels ou n'avez pas beaucoup d'expérience avec les IDE, l'utilisation d'un éditeur de texte vous permettra d'être opérationnel le plus rapidement. Au minimum, vous voudrez un éditeur avec coloration syntaxique, qui rend le code plus facile à lire. Si vous pensez que vous pouvez soumettre des correctifs pour le suivi de bogues, vous voudrez vous assurer que votre éditeur de texte ne modifie pas les terminaisons de ligne. Si vous utilisez Windows, [Notepad++](#) est un bon choix. Si vous êtes sur un Mac, [TextWrangler](#) est un choix populaire. Vous pouvez également essayer [TextMate](#).

Environnement de développement intégré

Un EDI (IDE en anglais) fait exactement ce que son nom implique : il comprend un ensemble d'outils que vous utiliserez normalement séparément. La plupart des IDE incluront le contrôle de code source qui vous permettra de valider et de mettre à jour directement votre code à partir de votre référentiel cvs. Il peut avoir un client FTP intégré pour faciliter le transfert de fichiers sur un serveur distant. Il disposera d'une vérification de syntaxe pour attraper les erreurs avant que vous essayiez d'exécuter le code sur un serveur.

Les deux IDE gratuites les plus populaires pour les développeurs PHP sont [Eclipse](#) et [NetBeans](#). Eclipse dispose de deux plugins différents pour travailler avec le code PHP : [PDT](#) et [PHPEclipse](#).

Je n'aime pas certaines traductions dans Elgg. Comment puis-je les changer ?

La meilleure façon de faire ceci est avec un plugin.

Créer le squelette du plugin

Squelette du plugin

Localisez la chaîne que vous voulez modifier

Toutes les chaînes qu'un utilisateur voit doivent être dans le répertoire `/languages` ou dans le répertoire des langues d'un plugin (`/mod/<plugin name>/languages`). Ceci est fait de sorte qu'il soit facile de changer la langue utilisée par Elgg. Pour plus d'informations à ce sujet, consultez la documentation développeur sur [Internationalisation](#).

Pour trouver la chaîne, utilisez `grep` ou un éditeur de texte qui fournit la recherche dans les fichiers pour localiser la chaîne. (Un bon éditeur de texte pour Windows est [Notepad++](#)) Disons que nous voulons modifier la chaîne `Ajouter un contact` en `Se faire un nouvel ami`. La commande `grep` qui permet de trouver cette chaîne serait `grep -r "Add friend" *`. Avec [Notepad++](#), vous utiliseriez la commande « Trouver dans les fichiers ». Vous recherchez la chaîne, définissez le filtre sur `*.php`, définissez le répertoire de recherche sur le répertoire racine de Elgg, et vous assurez qu'il recherche dans tous les sous-répertoires. Vous voudrez peut-être aussi définir que la recherche soit sensible à la casse.

Vous devriez placer la chaîne « Ajouter un contact » dans `/languages/fr.php`. Vous devriez voir quelque chose comme ceci dans le fichier :

```
'friend:add' => "Add friend",
```

Cela signifie que chaque fois qu'Elgg voit `friend:add` il le remplace par `Ajouter un contact`. Nous voulons changer la définition de `friend:add`.

Remplacer la chaîne

Pour remplacer cette définition, nous ajouterons un fichier de langues au plugin que nous avons construit dans la première étape.

1. Créer un nouveau répertoire : `/mod/<nom du plugin>/languages`
2. Créer un fichier dans ce répertoire appelé `fr.php`
3. Ajoutez ces lignes à ce fichier

```
<?php
return array(
    'friend:add' => 'Make a new friend',
);
```

Assurez-vous que vous n'avez pas d'espaces ou de nouvelles lignes avant `<?php`.

Vous avez terminé maintenant et devriez être en mesure d'activer le plugin et de voir le changement. Si vous remplacez la langue d'un plugin, assurez-vous que votre plugin est chargé après celui que vous essayez de modifier. L'ordre de chargement est déterminé dans la page Administration des outils de la section Administrateur. Au fur et à mesure que vous trouverez d'autres choses à changer, vous pouvez continuer à les ajouter à ce plugin.

Comment puis-je trouver le code qui fait X ?

La meilleure façon de trouver le code qui fait quelque chose que vous souhaitez changer est d'utiliser `grep` ou un outil de recherche similaire. Si `grep` n'est pas intégré à votre système d'exploitation, vous voudrez installer un outil `grep` ou utiliser un éditeur de texte/IDE qui dispose d'une bonne recherche dans les fichiers. `Notepad++` est un bon choix pour les utilisateurs de Windows. `Eclipse` avec PHP et `NetBeans` sont de bons choix pour n'importe quelle plateforme.

Exemple de chaîne (String)

Supposons que vous souhaitez trouver où se trouve le code de la boîte de *Connexion*. Une chaîne de la zone *Connexion* qui devrait être assez unique est `Se souvenir de moi` (« Remember me »). Faites un `grep` pour cette chaîne. Vous constaterez qu'elle n'est utilisée que dans le fichier `fr.php` (ou `en.php` pour la version d'origine) dans le répertoire `/languages`. Là, il est utilisé pour définir la chaîne *Internationalisation* `user:persistent`. Faites maintenant un `grep` pour cette chaîne. Vous la trouverez en deux endroits : le même fichier linguistique `fr.php` et dans `/views/default/forms/login.php`. Ce dernier emplacement définit le code html qui constitue la boîte *Connexion*.

Exemple d'Action

Supposons que vous souhaitez trouver le code qui s'exécute lorsqu'un utilisateur clique sur le bouton *Enregistrer* après avoir agencé les widgets sur une page de profil. Consultez la page du Profil d'un utilisateur de test. Utilisez Firebug pour inspecter le html de la page jusqu'à ce que vous trouviez l'action du formulaire de widgets de modification. Vous verrez que l'URL depuis la racine est `action/widgets/move`.

Faites un `grep` sur `widgets/move` et deux fichiers sont retournés. L'un est le code JavaScript pour les widgets : `/js/lib/ui.widgets.js`. L'autre, `/engine/lib/widgets.php`, est l'endroit où l'action est enregistrée à l'aide de `elgg_register_action('widgets/reorder')`. Vous ne connaissez peut-être pas cette fonction : dans ce cas, vous devriez consulter sa page dans la référence de l'API. Effectuez une recherche sur la fonction et cela renverra la documentation de la fonction. Cela vous indique que l'action se trouve à l'emplacement par défaut puisqu'un emplacement de fichier n'a pas été spécifié. L'emplacement par défaut des actions est `/actions` de sorte que vous trouverez le fichier à `/actions/widgets/move.php`.

Mode de débogage

Durant le processus d'installation, vous avez peut-être remarqué une case à cocher qui contrôlait si le mode débogage était activé ou désactivé. Ce paramètre peut également être modifié dans la page Administration du site. Le mode de débogage écrit beaucoup de données supplémentaires dans votre journal php. Par exemple, lorsque vous utilisez ce mode, chaque requête dans la base de données est écrite dans vos journaux. Cela peut être utile pour le débogage d'un problème, mais peut produire une quantité énorme de données qui peuvent ne pas être liées du tout au problème recherché. Vous pouvez expérimenter avec ce mode pour comprendre ce qu'il fait, mais assurez-vous d'exécuter Elgg en mode normal sur un serveur de production.

Avertissement : En raison de la quantité de données enregistrées, il est préférable de ne pas l'activer sur un serveur de production car cela peut remplir les fichiers journaux très rapidement.

Que contient le journal en mode débogage ?

- Toutes les requêtes de base de données
- Profilage des requêtes de base de données
- Durée de génération de page
- Nombre de requêtes par page
- Liste des fichiers linguistiques du plugin
- Des erreurs/avertissements supplémentaires par rapport au mode normal (il est très rare que ces types d'erreurs soient liés à tout problème que vous pourriez avoir)

À quoi ressemblent les données ?

```
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and
↳enabled='yes') results cached
[07-Mar-2009 14:27:20] SELECT guid from elggsites_entity where guid = 1 results cached
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and
↳enabled='yes') results cached
[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and
↳enabled='yes') results returned from cache
[07-Mar-2009 14:27:20] ** Sub part of GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggsites_entity where guid=1 results cached
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] DEBUG: 2009-03-07 14:27:20 (MST): "Undefined index: user" in
↳file /var/www/elgg/engine/lib/elgglib.php (line 62)
[07-Mar-2009 14:27:20] DEBUG: 2009-03-07 14:27:20 (MST): "Undefined index: pass" in
↳file /var/www/elgg/engine/lib/elgglib.php (line 62)
[07-Mar-2009 14:27:20] ***** DB PROFILING *****
[07-Mar-2009 14:27:20] 1 times: 'SELECT * from elggentities where guid=1 and ( (access_
↳id in (2) or (owner_guid = -1) or (access_id = 0 and owner_guid = -1)) and enabled='yes
↳')'
...
[07-Mar-2009 14:27:20] 2 times: 'update elggmetadata set access_id = 2 where entity_guid
↳= 1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggentities set owner_guid='0', access_id='2',
↳container_guid='0', time_updated='1236461868' WHERE guid=1'
[07-Mar-2009 14:27:20] 1 times: 'SELECT guid from elggsites_entity where guid = 1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggsites_entity set name='3124/944',
↳description='', url='http://example.org/' where guid=1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggusers_entity set prev_last_action = last_
↳action, last_action = 1236461868 where guid = 2'
[07-Mar-2009 14:27:20] DB Queries for this page: 56
[07-Mar-2009 14:27:20] *****
[07-Mar-2009 14:27:20] Page /action/admin/site/update_basic generated in 0.36997294426
↳seconds
```


Quels événements sont déclenchés sur chaque chargement de page ?

Il y a 4 *Elgg events* qui sont déclenchés sur chaque charge de page :

1. `plugins_boot`, `system`
2. `init`, `system`
3. `ready`, `system`
4. `shutdown`, `system`

Les trois premiers sont déclenchés dans `Elgg\Application::bootCore`. **`shutdown`**, **`system`** est déclenché dans `\Elgg\Application\ShutdownHandler` après que la réponse a été envoyée au client. Ils sont tous *documentés*.

Il existe d'autres événements déclenchés par Elgg occasionnellement (par exemple lorsqu'un utilisateur se connecte).

Copier un plugin

Il y a beaucoup de questions posées sur la façon de copier un plugin. Disons que vous voulez copier le plugin `blog` afin d'exécuter un plugin appelé `blog` et un autre appelé `poésie`. Ce n'est pas difficile, mais cela demande beaucoup de travail. Vous auriez besoin de

- modifier le nom du répertoire
- modifier les noms de toutes les fonctions (avoir deux fonctions avec le même nom provoque un plantage de PHP)
- modifier le nom de chaque vue (afin de ne pas remplacer les vues sur le plugin d'origine)
- modifier tous les sous-types de modèles de données
- modifier le fichier linguistique
- changer tout le reste de ce qui était spécifique au plugin d'origine

Note : Si vous essayez de cloner le plugin `groupes`, vous aurez la difficulté supplémentaire que le plugin de groupe ne définit pas de sous-type.

Autres types de fichiers

Voir aussi :

Trouver de l'aide

« Le plugin ne peut pas démarrer et a été désactivé » (« Plugin cannot start and has been deactivated ») ou « Ce plugin est invalide » (« This plugin is invalid »)

Cette erreur est habituellement accompagnée par plus de détails qui expliquent pourquoi le plugin est invalide. Ceci est généralement causé par un plugin mal installé.

Si vous installez un plugin appelé « test », il y aura un répertoire `test` dans le dossier `mod`. Dans ce dossier `test` il doit y avoir un fichier `manifest.xml` `/mod/test/manifest.xml`.

Si ce fichier n'existe pas, cela peut être causé par :

- l'installation d'un plugin dans le mauvais répertoire
- la création d'un répertoire dans `/mod` qui ne contient pas un plugin
- un mauvais transfert FTP
- l'extraction d'un plugin dans un niveau de répertoire supplémentaire (`myplugin.zip` est extrait dans `myplugin/myplugin`)

Si vous utilisez un hôte de type Unix et que els fichiers existent dans le bon répertoire, vérifiez les permissions. Elgg doit avoir accès en lecture + exécution sur els répertoires.

Page Blanche (WSOD = White Screen Of Death, l'Ecran Blanc De la Mort)

Une page vide, blanche (souvent appelé « écran blanc de la mort » - « white screen of death » ou WSOD) signifie qu'il y a un problème.

- fichier corrompu - essayez de transférer à nouveau le code vers votre serveur
- un appel à un module php qui n'a pas été chargé - ceci peut se produire après que vous ayez installé un plugin qui requiert un module spécifique.
- mauvais plugin - tous les plugins n'ont pas été écrits avec le même niveau de qualité aussi vous devriez faire attention à ceux que vous installez.

Pour trouver l'origine de l'erreur, modifiez le fichier .htaccess pour afficher les erreurs dans le navigateur. Définissez `display_errors` à 1 et chargez la même page à nouveau. Vous devriez voir les erreurs PHP dans votre navigateur. Modifiez à nouveau ce paramètre une fois que vous avez résolu le problème.

Note : Si vous utilisez le plugin Developer's Tools, allez dans sa page de configuration et assurez-vous que l'option « Afficher les erreurs PHP fatales » (« Display fatal PHP errors ») est bien activé.

Si l'écran blanc est dû à un mauvais plugin, retirez les derniers plugins que vous avez installé en supprimant leurs répertoires puis rechargez la page.

Note : Vous pouvez désactiver temporairement tous les plugins en créant un fichier vide dans `mod/mod_disabled`. Vous pouvez ensuite désactiver le plugin responsable via les outils du panneau d'administration.

Si vous avez un WSOD quand vous effectuez une action, comme vous identifier ou publier un article de blog, mais qu'il n'y a pas de message d'erreur, le plus probable est que ce soit causé par des caractères non-imprimables dans le code du plugin. Vérifiez le plugin et supprimez les espaces vides et nouvelles lignes situés après le tag de fin php (`?>`).

Page non trouvée

Si vous avez récemment installé votre site Elgg, la cause la plus probable d'une erreur de page non trouvée est que `mod_rewrite` n'est pas configuré correctement sur votre serveur. Il y a des informations dans la page de dépannage [Install Troubleshooting](#) sur la manière de résoudre cela. La deuxième cause la plus probable est que l'URL de votre site dans votre base de données est incorrecte.

Si vous exécutez votre site depuis un certain temps et commencez soudainement à obtenir des erreurs de page non trouvées, vous devez vous demander ce qui a changé. Avez-vous ajouté des plugins ? Avez-vous modifié la configuration de votre serveur ?

Pour déboguer une erreur page non trouvée (« page not found ») :

- Confirmez que le lien qui a mené à la page manquante est correct. S'il ne l'est pas, comment ce lien a-t-il été généré ?
- Confirmez que les règles de réécriture du .htaccess sont bien prises en compte.

Erreur de correspondance des jetons de connexion (« login token mismatch »)

Si vous devez vous connecter deux fois à votre site et que le message d'erreur après la première tentative indique qu'il y a eu une erreur de décalage de jeton, l'URL dans les paramètres d'Elgg ne correspond pas à l'URL utilisée pour y accéder. La cause la plus fréquente est d'ajouter ou de supprimer le « www » lors de l'accès au site. Par exemple, `www.elgg.org` vs `elgg.org`. Cela pose un problème avec la gestion des sessions en raison de la façon dont les navigateurs Web enregistrent les cookies.

Pour régler cela, vous pouvez ajouter des règles de réécriture (« rewrite rules »). Pour rediriger de `www.elgg.org` vers `elgg.org` dans Apache, les règles peuvent ressembler à

```
RewriteCond %{HTTP_HOST} .
RewriteCond %{HTTP_HOST} !^elgg\.org
RewriteRule (.*) http://elgg.org/$1 [R=301,L]
```

Rediriger depuis une adresse non-www vers www peut ressembler à

```
RewriteCond %{HTTP_HOST} ^elgg\.org
RewriteRule ^(.*)$ http://www.elgg.org/$1 [R=301,L]
```

Si vous ne savez pas comment configurer les règles de réécriture, demandez à votre hébergeur pour plus d'informations.

Il manque les champs __token ou __ts dans le formulaire. Veuillez recharger la page pour continuer

Toutes les actions Elgg requièrent un jeton de sécurité, et cette erreur se produit quand ce jeton est manquant. C'est soit un problème avec la configuration de votre serveur ou avec un plugin tierce-partie.

Si vous rencontrez ce cas lors d'une nouvelle installation, assurez-vous que votre serveur est correctement configuré et que vos règles de réécriture sont correctes. Si vous en faites l'expérience sur une mise à niveau, assurez-vous d'avoir mis à jour vos règles de réécriture dans `.htaccess` (Apache) ou dans la configuration du serveur.

Si vous rencontrez ce cas, désactivez tous les plugins tiers et réessayez. Les très vieux plugins pour Elgg n'utilisent pas de jeton de sécurité. Si le problème disparaît lorsque les plugins sont désactivés, c'est dû à un plugin qui devrait être mis à jour par son auteur.

Mode de maintenance

Pour mettre temporairement votre site hors-ligne, allez dans Administration -> Utilitaires (Utilities) -> Mode de Maintenance (Maintenance Mode). Complétez le formulaire et cliquez sur Enregistrer pour désactiver votre site pour tout le monde à l'exception des utilisateurs admin.

Email manquant

Si vos utilisateurs signalent que les e-mails de validation ne s'affichent pas, demandez-leur de vérifier leur dossier de spam. Il est possible que les e-mails provenant de votre serveur sont marqués comme spam. Cela dépend de nombreux facteurs : si votre fournisseur d'hébergement a un problème avec les spammeurs, la façon dont votre configuration de messagerie PHP est configurée, quel agent de transport de messagerie votre serveur utilise, ou si votre hébergement limite le nombre de courriels que vous pouvez envoyer en une heure.

Si personne ne reçoit d'email du tout, il est très probable que votre serveur n'est pas configuré correctement pour le courrier électronique. Votre serveur a besoin d'un programme pour envoyer un e-mail (appelé agent de transfert de messagerie - MTA) et PHP doit être configuré pour utiliser le MTA.

Pour vérifier rapidement si PHP et un MTA (Mail Transfer Agent) sont correctement configurés, créez un fichier sur votre serveur avec le contenu suivant :

```
<?php
$address = "your_email@your_host.com";

$subject = 'Test email.';

$body = 'If you can read this, your email is working.';

echo "Attempting to email $address...<br />";

if (mail($address, $subject, $body)) {
    echo 'SUCCESS! PHP successfully delivered email to your MTA. If you don\'t see
↳the email in your inbox in a few minutes, there is a problem with your MTA.';
} else {
    echo 'ERROR! PHP could not deliver email to your MTA. Check that your PHP
↳settings are correct for your MTA and your MTA will deliver email.';
}
```

Assurez-vous de remplacer « [votre_email@votre_hote.com](#) » par votre adresse e-mail réelle. Prenez soin de garder des guillemets autour d'elle ! Lorsque vous accédez à cette page via votre navigateur Web, il tentera d'envoyer un e-mail de test. Ce test vous permettra de savoir si PHP et votre MTA sont correctement configurés. Si elle échoue - soit vous obtenez une erreur, soit vous ne recevez jamais l'e-mail - vous aurez besoin de faire plus de recherches et éventuellement de contacter votre fournisseur de services.

La configuration complète des fonctionnalités de messagerie d'un MTA et de PHP dépasse le cadre de cette FAQ, et vous devez rechercher plus de ressources sur Internet à ce sujet. Quelques informations de base sur les paramètres php peuvent être trouvées sur le [PHP's site](#)

Journaux du serveur

Très probablement vous utilisez Apache comme serveur Web. Les avertissements et les erreurs sont écrits dans un journal par le serveur Web et peuvent être utiles pour les problèmes de débogage. Vous verrez généralement deux types de fichiers journaux : les journaux d'accès et les journaux d'erreurs. Les informations de PHP et Elgg sont inscrites dans le journal des erreurs du serveur.

- Linux – Le journal d'erreur est probablement dans /var/log/httpd ou /var/log/apache2.
- Windows - Il est probablement dans votre répertoire Apache.
- Mac OS - Le journal d'erreur est probablement dans /var/log/apache2/error_log

Si vous utilisez un hébergement partagé sans accès ssh, votre fournisseur d'hébergement peut fournir un mécanisme pour obtenir l'accès à vos journaux de serveur. Vous devrez leur poser des questions à ce sujet.

Comment fonctionne l'inscription ?

Avec une installation par défaut, voici comment fonctionne l'inscription :

1. L'utilisateur renseigne le formulaire d'inscription et l'envoie
2. Le compte utilisateur est créé et désactivé jusqu'à sa validation
3. Un email est envoyé à l'utilisateur afin de valider son compte
4. Quand l'utilisateur clique sur le lien, le compte est validé
5. L'utilisateur peut maintenant s'identifier

Les échecs durant ce processus comprennent l'utilisateur entrant une adresse e-mail incorrecte, l'e-mail de validation marqué comme spam, ou un utilisateur qui ne prend jamais la peine de valider le compte.

Validation de l'utilisateur

Par défaut, tous les utilisateurs qui s'inscrivent doivent valider leur compte par e-mail. Si un utilisateur a des difficultés pour valider son compte, vous pouvez valider manuellement les utilisateurs en allant dans Administration -> Utilisateurs -> Non validé.

Vous pouvez supprimer cette exigence en désactivant le plugin User Validation by Email.

Note : La suppression de la validation a certaines conséquences : il n'y a aucun moyen de savoir qu'un utilisateur inscrit à une adresse e-mail fonctionnelle, et cela peut ouvrir le système aux spammeurs.

Ajouter manuellement un utilisateur

Pour ajouter manuellement un utilisateur, dans la partie Administrer, accédez aux Utilisateurs. Là, vous verrez un titre intitulé « Ajouter un nouvel utilisateur ». Après avoir rempli les informations et soumis le formulaire, le nouvel utilisateur recevra un e-mail avec son nom d'utilisateur et son mot de passe, et un rappel pour changer le mot de passe.

Note : Elgg ne force pas l'utilisateur à changer le mot de passe.

Je suis en train de créer ou viens d'installer un nouveau thème, mais les images ou d'autres éléments ne fonctionnent pas

Assurez-vous que le thème est placé en tout dernier sur la liste des plugins.

Effacez le cache de votre navigateur et rechargez la page. Pour alléger la charge sur le serveur, Elgg demande au navigateur de charger rarement le fichier CSS. Un nouveau thème modifiera complètement le fichier CSS et un rafraîchissement devrait inciter le navigateur à demander à nouveau le fichier CSS.

Si vous créez ou modifiez un thème, assurez-vous d'avoir désactivé les caches simples et système. Cela peut être fait en activant le plugin Developer Tools, puis en naviguant vers Administration -> Outils de développement > Paramètres. Une fois que vous êtes satisfait des modifications, activez les caches ou les performances en souffriront.

Modifier les champs du profil

Dans les paramètres d'administration d'Elgg se trouve une page pour remplacer les champs de profil par défaut. Elgg donne par défaut à l'administrateur deux choix :

- Utilisez les champs de profil par défaut
- Remplacez les champs par défaut par un jeu de champs de profil personnalisés

Vous ne pouvez pas ajouter de nouveaux champs de profil aux champs par défaut. L'ajout d'un nouveau champ de profil via l'option remplacer les champs de profil efface ceux par défaut. Avant de laisser les utilisateurs s'inscrire, il est préférable de déterminer quels champs de profil vous voulez, quels types de champ ils devraient être, et l'ordre dans lequel ils devraient apparaître. Vous ne pouvez pas modifier le type de champ ou réordonner ou supprimer les champs après leur création sans effacer l'intégralité du profil.

Plus de flexibilité peut être obtenue grâce à des plugins. Il y a au moins deux plugins sur le site communautaire qui vous permettent d'avoir plus de contrôle sur les champs de profil. Le plugin [Profile Manager](#) est devenu très populaire dans la communauté Elgg. Il vous permet d'ajouter de nouveaux champs de profil quand vous le souhaitez, de modifier leur ordre, de regrouper les champs de profil et de les ajouter à l'inscription.

Modifier l'inscription

Le processus d'inscription peut être modifié par un plugin. Tout ce qui concerne l'inscription peut être modifié : l'apparence du formulaire, des champs d'inscription différents, des validations additionnelles des champs, des étapes additionnelles et ainsi de suite. Ces types de changements demandent quelques connaissances de base en HTML, CSS, PHP.

Une autre option est d'utiliser le plugin Profile Manager`_ qui vous permet d'ajouter des champs à la fois aux profils des utilisateurs et au formulaire d'inscription.

Créer le squelette du plugin *Squelette du plugin*

Modifier l'apparence de l'inscription Surchargez la vue `account/forms/register`

Modifiez le gestionnaire de l'action d'inscription Vous pouvez écrire votre propre action pour créer le compte utilisateur

Comment puis-je modifier les paramètres PHP en utilisant .htaccess ?

Vous pouvez modifier les paramètres php dans votre fichier `.htaccess`. Cela est particulièrement vrai si votre fournisseur d'hébergement ne vous donne pas accès au fichier `php.ini` du serveur. Les variables peuvent être liées aux limites de taille du téléchargement de fichiers, à la sécurité, à la longueur de la session ou à n'importe quel nombre d'autres attributs php. Pour des exemples sur comment faire cela, voyez la [documentation PHP](#) à ce sujet.

Connexion HTTPS activée par erreur

Si vous avez activé la connexion HTTPS mais que vous n'avez pas configuré SSL, vous êtes maintenant bloqué hors de votre installation Elgg. Pour désactiver ce paramètre de configuration, vous devrez modifier votre base de données. Utilisez un outil comme phpMyAdmin pour afficher votre base de données. Sélectionnez la table `config` et supprimez la ligne qui a le nom `https_login`.

Utiliser un site de test

Il est recommandé de toujours tester les nouvelles versions ou les nouveaux plugins sur un site de test avant de les exécuter sur un site de production (un site avec des utilisateurs réels). La façon la plus simple de le faire est de maintenir une installation séparée d'Elgg avec des comptes factices. Lors des tests, il est important d'utiliser des comptes factices qui ne sont pas des administrateurs pour tester ce que vos utilisateurs verront.

Un test plus réaliste consiste à refléter le contenu de votre site de production sur votre site de test. Suivez les instructions pour [dupliquer un site](#). Assurez-vous ensuite d'empêcher l'envoi d'e-mails à vos utilisateurs. Vous pouvez écrire un petit plugin qui redirige tous les e-mails vers votre propre compte (soyez conscient des plugins qui comprennent leur propre code d'envoi d'email personnalisé, car vous devrez modifier ces plugins). Une fois cela fait, vous pouvez afficher tout le contenu pour s'assurer que la mise à niveau ou le nouveau plugin fonctionne comme vous le souhaitez et ne casse rien. Si ce processus vous semble trop lourd, veuillez vous en tenir à l'exécution d'un site de test simple.

500 - Internal Server Error

Qu'est-ce que c'est ?

Une erreur interne du serveur ou **500 - Internal Server Error** signifie que le serveur web a rencontré un problème en servant une requête.

Voir aussi :

[La page Wikipédia sur les codes de statut HTTP](#)

Causes possibles

Configuraiton du serveur web La cause la plus fréquente pour cela est un serveur mal configuré. Si vous avez modifié le fichier `.htaccess` et ajouté quelque chose d'incorrect, Apache renverra une erreur 500.

Permissions sur les fichiers Il peut également s'agir d'un problème de permissions sur un fichier. Apache doit pouvoir lire les fichiers d'Elgg. L'utilisation des autorisations 755 sur les répertoires et 644 sur les fichiers permettra à Apache de lire les fichiers.

Lorsque je télécharge une photo ou que je modifie ma photo de profil, j'obtiens un écran blanc

Il est fort probable que la bibliothèque PHP GD n'est pas installée ou configurée correctement. Vous pourriez avoir besoin de l'aide de l'administrateur de votre serveur.

CSS manquant

Mauvaise URL

Parfois, les gens installent Elgg de sorte que l'URL de base est `localhost` et essaient ensuite d'afficher le site à l'aide d'un nom d'hôte. Dans ce cas, le navigateur ne pourra pas charger le fichier CSS. Essayez d'afficher la source de la page Web et de copier le lien du fichier CSS. Collez-le dans votre navigateur. Si vous obtenez une erreur 404, il est probable que c'est votre problème. Vous devrez modifier l'URL de base de votre site.

Erreur de syntaxe

Elgg stocke ses CSS sous forme de code PHP pour fournir flexibilité et puissance. S'il y a une erreur de syntaxe, le fichier CSS servi au navigateur peut être vide. La désactivation des plugins non groupés est la première étape recommandée.

Erreurs de règles de réécriture d'URL (Rewrite rules)

Un mauvais fichier `.htaccess` peut également entraîner une erreur 404 lors de la requête du fichier CSS. Cela pourrait se produire lors d'une mise à niveau si vous oubliez de mettre également à niveau le fichier `.htaccess`.

Devrais-je modifier la base de données manuellement ?

Avertissement : Non, vous ne devriez jamais modifier manuellement la base de données !

Est-ce que modifier manuellement la base de données va casser mon site ?

Oui.

Est-ce que je peux ajouter des champs supplémentaires aux tables dans la base de données ?

(AKA : Je ne comprends pas le modèle de données de Elgg *data model* donc je vais ajouter des colonnes. Est-ce que vous allez m'aider ?)

Non, c'est une mauvaise idée. Etudiez le modèle de données *data model* et vous verrez qu'à moins que ce soit une installation très spécifique et fortement personnalisée, vous pouvez faire tout ce dont vous avez besoin avec le modèle de données actuel d'Elgg.

Je veux supprimer des utilisateurs. Est-ce que je ne peux pas les supprimer directement de la table `elgg_entities` ?

Non, cela va corrompre votre base de données. Supprimez-les via le site.

Je veux supprimer du spam. Est-ce que je ne peux pas juste le rechercher et le supprimer dans la table `elgg_entities` ?

Non, cela va corrompre votre base de données. Supprimez-le via le site.

Quelqu'un sur le site communautaire m'a dit de modifier manuellement la base de données. Est-ce que je devrais le faire ?

Qui étais-ce ? Est-ce quelqu'un d'expérimenté avec Elgg, comme l'un des développeurs du noyau ou un auteur de plugins bien connu ? Est-ce qu'il ou elle vous a donné des instructions claires sur quoi modifier ? Si vous ne savez pas qui c'est, ou si vous ne comprenez pas ou n'êtes pas à l'aise avec ces instructions, ne modifiez pas la base de données manuellement.

Je connais PHP et MySQL et ai une raison légitime de modifier la base de données. Est-ce OK de modifier manuellement la base de données ?

Assurez-vous d'abord de bien comprendre le modèle de données d'Elgg *data model* et le schéma. Faites un backup, modifiez avec soin, puis testez abondamment.

Problème de connexion d'Internet Explorer (IE)

URL canonique

IE n'aime pas travailler avec des sites qui utilisent à la fois <http://exemple.org> et <http://www.exemple.org>. Il stocke plusieurs cookies et cela cause des problèmes. Il est préférable d'utiliser une seule URL de base. Pour plus d'informations sur la façon de le faire, voyez l'erreur « Login token mismatch error » (erreur de correspondance des jetons de connexion).

Cadre Chrome

Utiliser le cadre Chrome au sein de IE peut rompre le processus d'authentification.

Les emails ne supportent pas les caractères non-latins

Pour prendre en charge les caractères non-latins, (tels que les caractères cyrilliques ou chinois) Elgg a besoin que le support des chaînes de caractère multibyte ([»multibyte string support](#) ») soit compilé avec PHP.

Sur de nombreuses installations (par exemple Debian & Ubuntu), cela est activé par défaut. Si ce n'est pas le cas, vous devez l'activer (ou recompiler PHP pour l'inclure). Pour vérifier si votre serveur prend en charge les chaînes multioctets (multibyte), vérifiez [phpinfo](#).

Durée de session

La durée de la session est contrôlée par votre configuration php. Vous devrez d'abord localiser votre fichier `php.ini`. Dans ce fichier vous trouverez plusieurs variables de session. Une liste complète et ce qu'elles font peuvent être trouvés dans le [manuel de php](#).

Un fichier n'a pas de propriétaire

Il y a trois causes à cette erreur. Vous pourriez avoir une entité dans votre base de données qui a un `owner_guid` de `0`. Cela devrait être extrêmement rare et ne peut se produire que si votre base de données/serveur plante lors d'une opération d'écriture.

La deuxième cause serait une entité où le propriétaire n'existe plus. Cela peut se produire si un plugin est désactivé qui a été impliqué dans la création de l'entité, puis le propriétaire est supprimé, mais l'opération de suppression a échoué (parce que le plugin est désactivé). Si vous pouvez comprendre quelle entité est à l'origine de cette situation, regardez dans votre table `entities` et modifiez le `owner_guid` pour le vôtre, puis vous pouvez supprimer l'entité via Elgg.

Avertissement : Lisez la section « Dois-je modifier la base de données manuellement ? ». Soyez très prudent lors de l'édition directe de la base de données. Cela peut briser votre site. Faites **toujours** une sauvegarde avant de faire cela.

Corrections

Le plugin [Database Validator](#) vérifiera votre base de données pour ces causes et vous fournira une option pour les corriger. Assurez-vous de sauvegarder la base de données avant d'essayer l'option de correction.

Pas d'image

Si des images de profil, des images de groupe ou d'autres fichiers ont cessé de fonctionner sur votre site, il est probable qu'il s'agit d'une mauvaise configuration, surtout si vous avez migré vers un nouveau serveur.

Ce sont les erreurs de configuration les plus courantes qui font arrêter le fonctionnement des images et d'autres fichiers.

Mauvais chemin pour le répertoire de données

Assurez-vous que le chemin d'accès du répertoire de données est correct dans la zone d'administration du site. Il devrait avoir un slash à la fin.

Mauvaises permissions sur le répertoire de données

Vérifiez les permissions du répertoire de données. Le répertoire de données doit être lisible et inscriptible par l'utilisateur du serveur web.

Installation migrée avec un nouvel emplacement du répertoire de données

Si vous avez migré une installation et que vous devez modifier votre chemin d'accès au répertoire de données, assurez-vous de mettre à jour le SQL pour l'emplacement des données tel que documenté dans les [Dupliquer une installation](#) instructions.

Avertissements d'obsolescence

Si vous voyez de nombreux avertissements de dépréciation qui disent des choses comme

Deprecated in 1.7: `extend_view()` was deprecated by `elgg_extend_view()`!

alors vous utilisez un plugin qui a été écrit pour une ancienne version d'Elgg. Cela signifie que le plugin utilise des fonctions qui sont prévues pour être supprimés dans une future version d'Elgg. Vous pouvez demander au développeur de plugin si le plugin sera mis à jour ou vous pouvez mettre à jour le plugin vous-même. Si ni l'un ni l'autre de ceux-ci sont susceptibles de se produire, vous ne devriez pas utiliser ce plugin.

JavaScript ne fonctionne pas

Si le menu utilisateur qui apparaît au survol cesse de fonctionner ou que vous ne pouvez pas supprimer les messages système, cela signifie que JavaScript est cassé sur votre site. Ceci est généralement dû à un plugin ayant un mauvais code JavaScript. Vous devriez trouver le plugin à l'origine du problème et le désactiver. Vous pouvez le faire en désactivant les plugins non groupés un par un jusqu'à ce que le problème disparaisse. Une autre approche consiste à désactiver tous les plugins non groupés, puis à les activer un par un jusqu'à ce que le problème se reproduise.

La plupart des navigateurs Web vous donneront un indice quant à ce qui casse le code JavaScript. Ils disposent souvent d'une console pour les erreurs JavaScript ou un mode avancé pour afficher les erreurs. Une fois que vous voyez le message d'erreur, il est plus facile de localiser le problème.

Les adresses IP dans les journaux sont incorrectes

Si votre installation Elgg est derrière un serveur proxy ou un équilibreur de charge, les adresses IP enregistrées dans les journaux système (System Log) peuvent être erronées. Elles pourraient n'indiquer que les adresses IP du serveur proxy.

Afin de résoudre ceci, vous pouvez configurer les adresses IP du serveur proxy comme adresses IP de confiance, ce qui permettra au système d'accéder aux bonnes IP de vos utilisateurs.

Dans le fichier `settings.php`, vous pouvez configurer des paramètres pour `$CONFIG->http_request_trusted_proxy_ips` et `$CONFIG->http_request_trusted_proxy_headers`. Regardez dans le fichier `settings.php` pour plus d'information.

Sécurité

Est-ce `upgrade.php` pose des soucis de sécurité ?

`Upgrade.php` est un fichier utilisé pour exécuter des mises à niveau du code et de la base de données. Il est placé à la racine de l'installation et il n'y a pas besoin d'un compte identifié pour y accéder. Sur un site totalement mis à niveau, l'exécution du fichier va seulement réinitialiser les caches et quitter, de sorte que ceci ne constitue pas un problème de sécurité.

SI cela vous préoccupe tout de même, vous pouvez supprimer, déplacer ou modifier les permissions sur ce fichier jusqu'à ce que vous ayez besoin d'effectuer une mise à niveau.

Devrais-je supprimer `install.php` ?

Ce fichier est utilisé pour installer Elgg et n'a pas besoin d'être supprimé. Le fichier vérifie si Elgg est déjà installé et redirige l'utilisateur vers la première page si c'est le cas.

Filtrage

Le filtrage est utilisé dans Elgg pour rendre les attaques XSS plus difficiles. L'objectif du filtrage est de supprimer les JavaScript et autres entrées dangereuses envoyées par les utilisateurs.

Le filtrage est effectué par l'intermédiaire de la fonction `filter_tags()`. Cette fonction prend une chaîne et renvoie une chaîne filtrée. Elle déclenche un `:ref:hook plugin <design/events#plugin-hooks>` validate, input`. Par défaut Elgg est fourni avec le code de filtrage `htmlawed` sous la forme d'un plugin. Les développeurs peuvent intégrer tout autre code de filtrage supplémentaire ou en remplacement sous forme de plugin.

La fonction `filter_tags()` est appelée pour chaque saisie utilisateur dès lors que la saisie est obtenue à travers un appel à `get_input()`. Si pour quelque raison un développeur souhaite ne pas appliquer le filtrage par défaut sur certaines saisies utilisateur, la fonction `get_input()` a un paramètre pour désactiver le filtrage.

Développement

Que dois-je utiliser pour modifier le code php ?

Il existe deux options principales : l'éditeur de texte ou *environnement de développement intégré* (IDE).

Éditeur de texte

Si vous débutez dans le développement de logiciels ou n'avez pas beaucoup d'expérience avec les IDE, l'utilisation d'un éditeur de texte vous permettra d'être opérationnel le plus rapidement. Au minimum, vous voudrez un éditeur avec coloration syntaxique, qui rend le code plus facile à lire. Si vous pensez que vous pouvez soumettre des correctifs pour le suivi de bogues, vous voudrez vous assurer que votre éditeur de texte ne modifie pas les terminaisons de ligne. Si vous utilisez Windows, [Notepad++](#) est un bon choix. Si vous êtes sur un Mac, [TextWrangler](#) est un choix populaire. Vous pouvez également essayer [TextMate](#).

Environnement de développement intégré

Un EDI (IDE en anglais) fait exactement ce que son nom implique : il comprend un ensemble d'outils que vous utiliserez normalement séparément. La plupart des IDE incluront le contrôle de code source qui vous permettra de valider et de mettre à jour directement votre code à partir de votre référentiel cvs. Il peut avoir un client FTP intégré pour faciliter le transfert de fichiers sur un serveur distant. Il disposera d'une vérification de syntaxe pour attraper les erreurs avant que vous essayiez d'exécuter le code sur un serveur.

Les deux IDE gratuites les plus populaires pour les développeurs PHP sont [Eclipse](#) et [NetBeans](#). Eclipse dispose de deux plugins différents pour travailler avec le code PHP : [PDT](#) et [PHPEclipse](#).

Je n'aime pas certaines traductions dans Elgg. Comment puis-je les changer ?

La meilleure façon de faire ceci est avec un plugin.

Créer le squelette du plugin

Squelette du plugin

Localisez la chaîne que vous voulez modifier

Toutes les chaînes qu'un utilisateur voit doivent être dans le répertoire `/languages` ou dans le répertoire des langues d'un plugin (`/mod/<plugin name>/languages`). Ceci est fait de sorte qu'il soit facile de changer la langue utilisée par Elgg. Pour plus d'informations à ce sujet, consultez la documentation développeur sur [Internationalisation](#).

Pour trouver la chaîne, utilisez `grep` ou un éditeur de texte qui fournit la recherche dans les fichiers pour localiser la chaîne. (Un bon éditeur de texte pour Windows est [Notepad++](#)) Disons que nous voulons modifier la chaîne `Ajouter un contact en Se faire un nouvel ami`. La commande `grep` qui permet de trouver cette chaîne serait `grep -r "Add friend" *`. Avec [Notepad++](#), vous utiliseriez la commande « Trouver dans les fichiers ». Vous recherchez la chaîne, définissez le filtre sur `*.php`, définissez le répertoire de recherche sur le répertoire racine de Elgg, et vous assurez qu'il recherche dans tous les sous-répertoires. Vous voudrez peut-être aussi définir que la recherche soit sensible à la casse.

Vous devriez placer la chaîne « Ajouter un contact » dans `/languages/fr.php`. Vous devriez voir quelque chose comme ceci dans le fichier :

```
'friend:add' => "Add friend",
```

Cela signifie que chaque fois qu'Elgg voit `friend:add` il le remplace par `Ajouter un contact`. Nous voulons changer la définition de `friend:add`.

Remplacer la chaîne

Pour remplacer cette définition, nous ajouterons un fichier de langues au plugin que nous avons construit dans la première étape.

1. Créer un nouveau répertoire : `/mod/<nom du plugin>/languages`
2. Créer un fichier dans ce répertoire appelé `fr.php`
3. Ajoutez ces lignes à ce fichier

```
<?php

return array(
    'friend:add' => 'Make a new friend',
);
```

Assurez-vous que vous n'avez pas d'espaces ou de nouvelles lignes avant `<?php`.

Vous avez terminé maintenant et devriez être en mesure d'activer le plugin et de voir le changement. Si vous remplacez la langue d'un plugin, assurez-vous que votre plugin est chargé après celui que vous essayez de modifier. L'ordre de chargement est déterminé dans la page Administration des outils de la section Administrateur. Au fur et à mesure que vous trouverez d'autres choses à changer, vous pouvez continuer à les ajouter à ce plugin.

Comment puis-je trouver le code qui fait X ?

La meilleure façon de trouver le code qui fait quelque chose que vous souhaitez changer est d'utiliser `grep` ou un outil de recherche similaire. Si `grep` n'est pas intégré à votre système d'exploitation, vous voudrez installer un outil `grep` ou utiliser un éditeur de texte/IDE qui dispose d'une bonne recherche dans les fichiers. `Notepad++` est un bon choix pour les utilisateurs de Windows. `Eclipse` avec PHP et `NetBeans` sont de bons choix pour n'importe quelle plateforme.

Exemple de chaîne (String)

Supposons que vous souhaitez trouver où se trouve le code de la boîte de *Connexion*. Une chaîne de la zone *Connexion* qui devrait être assez unique est `Se souvenir de moi` (« Remember me »). Faites un `grep` pour cette chaîne. Vous constaterez qu'elle n'est utilisée que dans le fichier `fr.php` (ou `en.php` pour la version d'origine) dans le répertoire `/languages`. Là, il est utilisé pour définir la chaîne `Internationalisation user:persistent`. Faites maintenant un `grep` pour cette chaîne. Vous la trouverez en deux endroits : le même fichier linguistique `fr.php` et dans `/views/default/forms/login.php`. Ce dernier emplacement définit le code html qui constitue la boîte *Connexion*.

Exemple d'Action

Supposons que vous souhaitez trouver le code qui s'exécute lorsqu'un utilisateur clique sur le bouton *Enregistrer* après avoir agencé les widgets sur une page de profil. Consultez la page du Profil d'un utilisateur de test. Utilisez Firebug pour inspecter le html de la page jusqu'à ce que vous trouviez l'action du formulaire de widgets de modification. Vous verrez que l'URL depuis la racine est `action/widgets/move`.

Faites un `grep` sur `widgets/move` et deux fichiers sont retournés. L'un est le code JavaScript pour les widgets : `/js/lib/ui.widgets.js`. L'autre, `/engine/lib/widgets.php`, est l'endroit où l'action est enregistrée à l'aide de `elgg_register_action('widgets/reorder')`. Vous ne connaissez peut-être pas cette fonction : dans ce cas, vous devriez consulter sa page dans la référence de l'API. Effectuez une recherche sur la fonction et cela renverra la documentation de la fonction. Cela vous indique que l'action se trouve à l'emplacement par défaut puisqu'un emplacement de fichier n'a pas été spécifié. L'emplacement par défaut des actions est `/actions` de sorte que vous trouverez le fichier à `/actions/widgets/move.php`.

Mode de débogage

Durant le processus d'installation, vous avez peut-être remarqué une case à cocher qui contrôlait si le mode débogage était activé ou désactivé. Ce paramètre peut également être modifié dans la page Administration du site. Le mode de débogage écrit beaucoup de données supplémentaires dans votre journal php. Par exemple, lorsque vous utilisez ce mode, chaque requête dans la base de données est écrite dans vos journaux. Cela peut être utile pour le débogage d'un problème, mais peut produire une quantité énorme de données qui peuvent ne pas être liées du tout au problème recherché. Vous pouvez expérimenter avec ce mode pour comprendre ce qu'il fait, mais assurez-vous d'exécuter Elgg en mode normal sur un serveur de production.

Avertissement : En raison de la quantité de données enregistrées, il est préférable de ne pas l'activer sur un serveur de production car cela peut remplir les fichiers journaux très rapidement.

Que contient le journal en mode débogage ?

- Toutes les requêtes de base de données
- Profilage des requêtes de base de données
- Durée de génération de page
- Nombre de requêtes par page
- Liste des fichiers linguistiques du plugin
- Des erreurs/avertissements supplémentaires par rapport au mode normal (il est très rare que ces types d'erreurs soient liés à tout problème que vous pourriez avoir)

À quoi ressemblent les données ?

```
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and ↵
↵enabled='yes') results cached
[07-Mar-2009 14:27:20] SELECT guid from elggsites_entity where guid = 1 results cached
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and ↵
↵enabled='yes') results cached
```

(suite sur la page suivante)

(suite de la page précédente)

```

[07-Mar-2009 14:27:20] ** GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggentities where guid=1 and ( (1 = 1) and_
↳enabled='yes') results returned from cache
[07-Mar-2009 14:27:20] ** Sub part of GUID:1 loaded from DB
[07-Mar-2009 14:27:20] SELECT * from elggsites_entity where guid=1 results cached
[07-Mar-2009 14:27:20] Query cache invalidated
[07-Mar-2009 14:27:20] DEBUG: 2009-03-07 14:27:20 (MST): "Undefined index: user" in_
↳file /var/www/elgg/engine/lib/elgglib.php (line 62)
[07-Mar-2009 14:27:20] DEBUG: 2009-03-07 14:27:20 (MST): "Undefined index: pass" in_
↳file /var/www/elgg/engine/lib/elgglib.php (line 62)
[07-Mar-2009 14:27:20] ***** DB PROFILING *****
[07-Mar-2009 14:27:20] 1 times: 'SELECT * from elggentities where guid=1 and ( (access_
↳id in (2) or (owner_guid = -1) or (access_id = 0 and owner_guid = -1)) and enabled='yes
↳')'
...
[07-Mar-2009 14:27:20] 2 times: 'update elggmetadata set access_id = 2 where entity_guid_
↳= 1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggentities set owner_guid='0', access_id='2',_
↳container_guid='0', time_updated='1236461868' WHERE guid=1'
[07-Mar-2009 14:27:20] 1 times: 'SELECT guid from elggsites_entity where guid = 1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggsites_entity set name='3124/944',_
↳description='', url='http://example.org/' where guid=1'
[07-Mar-2009 14:27:20] 1 times: 'UPDATE elggusers_entity set prev_last_action = last_
↳action, last_action = 1236461868 where guid = 2'
[07-Mar-2009 14:27:20] DB Queries for this page: 56
[07-Mar-2009 14:27:20] *****
[07-Mar-2009 14:27:20] Page /action/admin/site/update_basic generated in 0.36997294426_
↳seconds

```

Quels événements sont déclenchés sur chaque chargement de page ?

Il y a 4 *Elgg events* qui sont déclenchés sur chaque charge de page :

1. plugins_boot, system
2. init, system
3. ready, system
4. shutdown, system

Les trois premiers sont déclenchés dans `Elgg\Application::bootCore`. **shutdown, system** est déclenché dans `Elgg\Application\ShutdownHandler` après que la réponse a été envoyée au client. Ils sont tous *documentés*.

Il existe d'autres événements déclenchés par Elgg occasionnellement (par exemple lorsqu'un utilisateur se connecte).

Copier un plugin

Il y a beaucoup de questions posées sur la façon de copier un plugin. Disons que vous voulez copier le plugin `blog` afin d'exécuter un plugin appelé `blog` et un autre appelé `poésie`. Ce n'est pas difficile, mais cela demande beaucoup de travail. Vous auriez besoin de

- modifier le nom du répertoire
- modifier les noms de toutes les fonctions (avoir deux fonctions avec le même nom provoque un plantage de PHP)
- modifiez le nom de chaque vue (afin de ne pas remplacer les vues sur le plugin d'origine)
- modifier tous les sous-types de modèles de données
- modifier le fichier linguistique
- changer tout le reste de ce qui était spécifique au plugin d'origine

Note : Si vous essayez de cloner le plugin `groupes`, vous aurez la difficulté supplémentaire que le plugin de groupe ne définit pas de sous-type.

3.7.3 Feuille de route

Dans quelle direction va le projet ? Quelles nouvelles fonctionnalités passionnantes sont à venir bientôt ?

Nous ne publions pas de feuilles de route détaillées, mais il est possible d'avoir une idée de notre orientation générale en utilisant les ressources suivantes :

- Notre *groupe de feedback et de planification* ([feedback and planning group](#)) est utilisé pour accueillir des discussions préliminaires sur les travaux à venir.
- Nos *jalons Github* (milestones) représentent une orientation générale pour les futures versions d'Elgg. C'est la chose la plus proche d'une feuille de route traditionnelle que nous avons.
- Les `pull requests` sur Github vous donneront une bonne idée de ce qui est en cours de développement, mais rien n'est sûr tant que la PR n'est pas effectivement enregistrée.
- Nous utilisons le [developer blog](#) pour publier les annonces de fonctionnalités qui ont récemment été ajoutées dans notre branche de développement, ce qui donne l'indication la plus sûre sur quelles fonctionnalités seront disponibles dans la prochaine version.

Valeurs

Nous avons plusieurs objectifs/valeurs généraux qui affectent les orientations prises par Elgg. Les améliorations doivent généralement promouvoir ces valeurs afin d'être acceptées.

Accessibilité

Les sites basés sur Elgg devraient être utilisables par n'importe qui n'importe où. Cela signifie que nous nous efforcerons toujours de rendre Elgg :

- Compatible sur tous types d'appareils - mobile, tablette, ordinateur de bureau, etc.
- Multilingue – i18n, RTL, etc.
- Capacité-agnostique – compatible tactile, clavier, lecteur d'écran

Testabilité

Nous voulons **rendre les tests manuels inutiles** pour les développeurs du noyau, les auteurs de plugins, et les administrateurs de sites en faisant la promotion et en permettant des tests rapides et automatisés à tous les niveaux de la pile Elgg.

Nous pensons que les API sont cassées si elles exigent des auteurs de plugins d'écrire du code non testable. Nous savons qu'il y a beaucoup de violations de ce principe dans le noyau actuellement et nous travaillons à les corriger.

Nous avons hâte d'un monde dans lequel les développeurs du noyau n'ont pas besoin de faire de tests manuels pour vérifier la correction du code contribué à Elgg. De manière similaire, nous envisageons un monde dans lequel les administrateurs de site peuvent mettre à niveau et installer de nouveaux plugins avec la certitude que tout fonctionne bien ensemble.

TODO : d'autres objectifs/valeurs ?

FAQ

Quand la fonctionnalité X sera-t-elle implémentée ?

Nous ne pouvons pas promettre quand les fonctionnalités seront mises en œuvre parce que les nouvelles fonctionnalités sont inscrites dans Elgg seulement lorsque quelqu'un est suffisamment motivé pour implémenter la fonctionnalité et soumettre une demande de fusion. Le mieux que nous puissions faire est de vous dire de regarder pour quelles fonctionnalités les développeurs existants ont exprimé leur intérêt pour travailler dessus.

La meilleure façon de s'assurer qu'une fonctionnalité est implémentée est d'en discuter avec l'équipe du noyau et de la mettre en œuvre vous-même. Consultez notre guide *Guides du contributeur* si vous êtes intéressé. Nous adorons les nouveaux contributeurs !

Ne comptez pas sur les améliorations futures si vous êtes hésitez à savoir s'il faut utiliser Elgg. Évaluez-le compte tenu de l'ensemble de fonctionnalités existantes. Les fonctionnalités à venir ne se réaliseront certainement pas à temps dans votre calendrier.

Quand la version X.Y.Z sortira-t-elle ?

La prochaine version sera publiée lorsque l'équipe du noyau se sentira prête et aura le temps de publier la version. <http://github.com/Elgg/Elgg/issues/milestones> vous donnera une idée générale de la chronologie.

3.7.4 Politique de versions

À quoi s'attendre lors de la mise à niveau de Elgg.

Nous adhérons au "semantic versioning" <http://semver.org/>.

Suivez le blog pour [rester à jour sur les dernières versions](#).

Contenus

- *Sorties de correctifs/bugfix (2.1.x)*
- *Versions mineures/fonctionnalités (2.x.0)*
- *Révisions Majeures (x.0.0)*
- *Versions Alphas, Betas, et Candidates (RC = Release Candidates)*
- *Rétrocompatibilité*

Sorties de correctifs/bugfix (2.1.x)

Toutes les deux semaines.

Les versions de correctifs sont effectuées régulièrement pour s'assurer qu'Elgg reste stable, sécurisé et exempt de bogues. Plus le troisième chiffre est élevé, plus la version est testée et stable.

Dans la mesure où les versions correctives de bogues se concentrent sur la correction des bogues et évitent d'apporter des changements majeurs, les thèmes et les plugins doivent fonctionner de version corrective en version corrective.

Versions mineures/fonctionnalités (2.x.0)

Tous les trois mois.

Chaque fois que nous introduisons de nouvelles fonctionnalités, nous incrémentons le numéro de version du milieu. Ces versions ne sont pas aussi matures que les correctifs, mais sont considérées comme stables et utilisables.

Nous faisons tous les efforts possibles pour être rétrocompatible dans ces versions, de sorte que les plugins devraient fonctionner de version mineure en version mineure.

Toutefois, les plugins pourraient devoir être mis à jour pour utiliser les nouvelles fonctionnalités.

Révisions Majeures (x.0.0)

Chaque année.

Inévitablement, l'amélioration d'Elgg nécessite des changements non rétro-compatibles et une nouvelle version majeure est alors publiée. Ces versions sont des occasions pour l'équipe de base d'apporter des changements stratégiques et de rupture à la plateforme sous-jacente. Les thèmes et les plugins des anciennes versions ne sont pas censés fonctionner sans modification sur les différentes versions majeures.

Nous pouvons supprimer des API obsolètes, mais nous ne supprimerons pas les API sans les avoir d'abord rendues obsolètes (deprecated).

Les dépendances d'Elgg peuvent être mises à niveau par leur version majeure ou supprimées entièrement. Nous ne supprimerons pas de dépendance avant une version majeure, mais nous ne rendons pas obsolète (« deprecate ») les dépendances ou nous n'émettons pas d'avertissement avant de les supprimer.

Votre package, plugin ou application doit déclarer ses propres dépendances directement afin que cela ne cause pas de problème.

Versions Alphas, Betas, et Candidates (RC = Release Candidates)

Avant les versions majeures (et parfois avant les versions de fonctionnalités), l'équipe de base offrira une version pré-version d'Elgg pour obtenir des tests et des commentaires dans le monde réel sur la version. Ceux-ci sont destinés à des tests seulement et ne doivent pas être utilisés sur un site en direct.

SemVer 2.0 ne définit pas de signification particulière pour les pré-versions, mais nous abordons les versions alpha, bêta et rc avec ces indications générales :

Une pré-version -alpha-X signifie qu'il y a encore des modifications de rupture prévues, mais l'ensemble des fonctionnalités de la version est gelé. Aucune nouvelle fonctionnalité ou modification de rupture ne peuvent être proposées pour cette version.

Une pré-version -beta.X signifie qu'il n'y a plus de modification de rupture connue à inclure, mais il reste des régressions connues ou des bogues critiques à corriger.

Une pré-version `-rc.X` signifie qu'il n'y a plus de régression connue ou de bogue critique à corriger. Cette version pourrait devenir la version stable finale d'Elgg si aucun nouveau blocage n'est signalé.

Rétrocompatibilité

Certaines parties du système ont besoin de clarifications supplémentaires si nous parlons d'être rétro-compatibles. Tout ce qui est considéré comme une API publique doit respecter les règles de rétro-compatibilité qui font partie du [semantic versioning](#).

Classes et fonctions

Les classes et fonctions marquées `@internal` ne sont pas considérées comme faisant partie de l'API publique et peuvent être modifiées/supprimées à tout moment. Si une classe est marquée `@internal`, toutes les propriétés et méthodes de cette classe sont considérées comme une API privée et peuvent être modifiées/supprimées à tout moment.

Fonctions de rappel des événements et des hooks de plugin

Toutes les fonctions de rappel des événements et des hooks plugin ne doivent jamais être appelés directement, mais seulement être appelés par leur événement / hook plugin respectif.

Le nom de la fonction de rappel est considéré comme une API dans le sens où les développeurs de plugins doivent être en mesure de compter sur le fait qu'ils peuvent (dé-)enregistrer une fonction de rappel. Ceci ne s'applique que si la fonction de rappel sert toujours le même but. Si une fonction de rappel devient obsolète, elle peut être supprimée du système.

Avertissement : Les exceptions à ces règles sont les fonctions de rappel liées aux événements `system`, ces fonctions de rappel peuvent être renommées / supprimées à tout moment.

- `plugins_load`
- `plugins_boot`
- `init`
- `ready`
- `shutdown`
- `upgrade`

Vues

- Les noms de vue sont des API.
- Les arguments de la vue (tableau `$vars`) sont des API.
- La suppression de vues ou le changement de nom de vues suit les stratégies de dépréciation de l'API.
- L'ajout de nouvelles vues nécessite une modification mineure de la version.
- La sortie des vues n'est pas une API et peut être modifiée entre les versions de correctifs.

3.7.5 Politique de suport

À partir d'Elgg 2.0, chaque version mineure reçoit des corrections de bogue et de sécurité jusqu'à la prochaine version mineure.

Contenus

- *Versions supportées sur le long terme (LTS)*
- *Bugs*
- *Problèmes de sécurité*
- *Chronologie*

Versions supportées sur le long terme (LTS)

Dans chaque version majeure, la dernière version mineure est désignée pour le support à long terme (« LTS ») et recevra des corrections de bogues jusqu'à 1 an après la sortie de la prochaine version majeure, et les correctifs de sécurité jusqu'à la 2ème version majeure suivante.

Par exemple, 2.3 est la dernière version mineure dans les 2.x. Elle recevra des corrections de bogues jusqu'à 1 après que la version 3.0 a été publiée, et les correctifs de sécurité jusqu'à ce que la version 4.0 soit publiée.

Voir aussi :

- *Politique de versions*
- doc :/contribute/issues

Bugs

Lorsque des bogues sont trouvés, un effort de bonne foi sera fait pour corriger la publication de la version LTS, mais **tous les correctifs ne seront pas rétro-portés**. Par exemple, certains correctifs peuvent dépendre de nouvelles API, rompre la rétro-compatibilité, ou nécessitent une refactorisation significative.

Important : Si un correctif risque de compromettre la stabilité de la branche LTS, il ne sera pas inclus.

Problèmes de sécurité

Lorsqu'un problème de sécurité est constaté, tous les efforts seront faits pour corriger la version LTS.

Attention : Veuillez signaler tout problème de sécurité à [security @ elgg . org](mailto:security@elgg.org)

Chronologie

Vous trouverez ci-dessous un tableau décrivant les détails de chaque version (les dates futures sont provisoires) :

Version	Première révision stable	Corrections de bogues à travers	Correctifs de sécurité à travers
1.12	Juillet 2015	Avril 2019	Avril 2019
2.0	Décembre 2015	Mars 2016	
2.1	Mars 2016	Juin 2016	
2.2	Juin 2016	Novembre 2016	
2.3 LTS	Novembre 2016	Avril 2020	Jusqu'à la 4.0
3.0	Avril 2019	Juillet 2019	
3.1	Juillet 2019	Octobre 2019	
3.2	Octobre 2019	Janvier 2020	
3.3	Janvier 2019		
4.0	A FAIRE		

3.7.6 Historique

Le nom vient d'une ville en Suisse. Cela signifie aussi « élan » ou « orignal » en Danois.

Le financement initial d'Elgg a été initié par une société appelée Curverider Ltd, qui a été lancée par David Tosh et Ben Werdmuller. En 2010, Curverider a été acquise par Thematic Networks et le contrôle du projet open-source a été confié à [The Elgg Foundation](#). Aujourd'hui, Elgg est un projet open source communautaire avec une variété de contributeurs et de soutiens.