31 août 2008

Pear et les librairies PHP

Catégorie : Web Tags : LP

~~Retrouvez cet article dans : Linux Pratique 32~~

Quel que soit le langage utilisé, tout programme peut faire appel à une ou plusieurs libraires. Celles-ci intègrent un ensemble de fonctions spécialisées dont le programme est susceptible de se servir. PEAR est un projet qui met à disposition des développeurs des librairies pouvant être utilisées en langage PHP. Comment récupérer ces librairies et comment s'en servir ?

Qu'est-ce que PEAR ?

PEAR est l'acronyme de PHP Extension and Application Repository. C'est un projet communautaire dont la fonction est de fournir aux utilisateurs de PHP une librairie structurée de code source libre. Le site officiel de PEAR se trouve à l'adresse suivante : http://pear.php.net/
Dans PEAR, le code est sous forme de packages. Chaque package est un projet séparé, mais peut inclure des dépendances avec d'autres packages (tout comme les paquets Debian par exemple). Les packages sont distribués sous forme d'une archive tar, compressée, contenant entre autres, un fichier XML de description. Tous les packages PEAR sont enregistrés et classés dans une base de données centrale se trouvant sur http://pear.php.net/ (Fig.1). Vous pourrez y trouver aussi des packages d'autres origines, mais toujours de source libre.

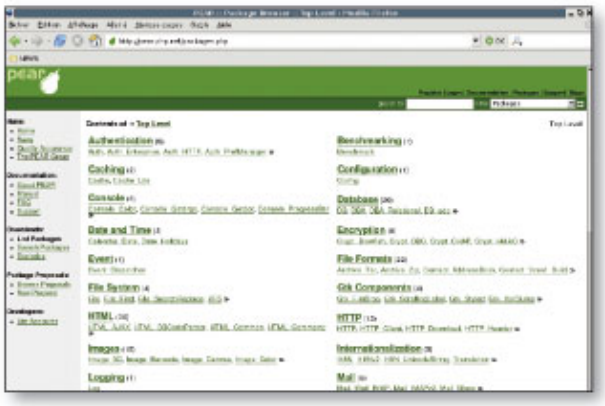


Fig. 1

Comment installer les librairies PEAR ?

Installation du gestionnaire

Si vous disposez d'une version récente de PHP (> 4.3.0), vous avez certainement déjà installé le paquet correspondant au gestionnaire. Par exemple, sous Debian, il s'agit du paquet php4-pear. Si vous utilisez une version PHP 4.2.x ou plus ancienne, l'installation du gestionnaire est différente, je vous invite à consulter la documentation sur le site officiel.
Si votre site est hébergé sur un serveur mutualisé et que vous ne disposez d'aucun accès au système (via les protocoles Telnet, SSH, etc.), vous ne pourrez pas utiliser le gestionnaire de PEAR. Mais en général, les hébergeurs font en sorte d'installer les packages PEAR les plus courants.

Installation des packages

Installation automatique

L'installation par ligne de commande est la procédure la plus simple pour ajouter des packages PEAR sur votre système : une connexion est initiée vers le serveur de paquetages de PEAR via le protocole HTTP, le package est téléchargé sur votre système et installé à l'endroit désiré. L'installation en ligne de commande est très simple à utiliser. Lancez simplement cette commande :

```
$ pear install nom_package
```

Une liste des packages disponibles se trouve à l'adresse http://pear.php.net /packages.php. Mais vous pouvez également afficher cette liste via la commande :

```
$ pear remote-list
```

Si vous n'avez pas installé le gestionnaire de librairies, il vous faudra effectuer une installation manuelle.

Installation manuelle

L'installation manuelle est très simple :

- Repérez le package de la librairie qui vous intéresse, sur le site officiel. Cliquez sur la librairie en question, puis lorsque vous vous trouvez sur la fiche descriptive, cliquez sur le lien de la dernière version (rubrique Current Release).
- Téléchargez le fichier .tgz (dans un dossier temporaire).
- Veillez à contrôler si la librairie qui vous intéresse dépend d'autres librairies, auquel cas, il faudra également télécharger les .tgz des dépendances. Les dépendances d'une librairie sont indiquées dans l'onglet Download, colonne Informations, rubrique Dependencies.
- Il vous faut ensuite extraire le contenu des archives (toujours dans le répertoire temporaire).
- Déplacez ensuite les fichiers obtenus sur votre serveur web (via FTP ou autre), dans un répertoire que vous aurez dédié à ces librairies.
- Modifier la directive ~~include_path~~, afin qu'elle contienne le chemin d'accès aux fichiers que vous venez de déplacer.

Si vous avez accès au fichier de configuration ~~php.ini~~, il vous suffira d'y ajouter directement le chemin vers les librairies, à la directive ~~include_path~~. Si ce n'est pas le cas, vous devrez modifier la directive ~~include_path~~ dans le script où vous souhaitez utiliser la librairie en question :

```
<?php
ini_set("include_path", '/mon_site.com/mes_librairies/' . élément_separateur . ini_get("include_path"));
?>
```

Une fois l'installation terminée, les fonctions propres à la librairie peuvent être utilisées dans vos scripts PHP.

Utiliser une librairie : exemple de PEAR::DB

À quoi sert PEAR::DB ?

La librairie ~~PEAR::DB~~ est une librairie permettant de gérer l'utilisation de bases de données. Mais elle permet aussi (et c'est ce qui nous intéresse principalement) de faire abstraction du type de serveur de base de données (Database Abstraction

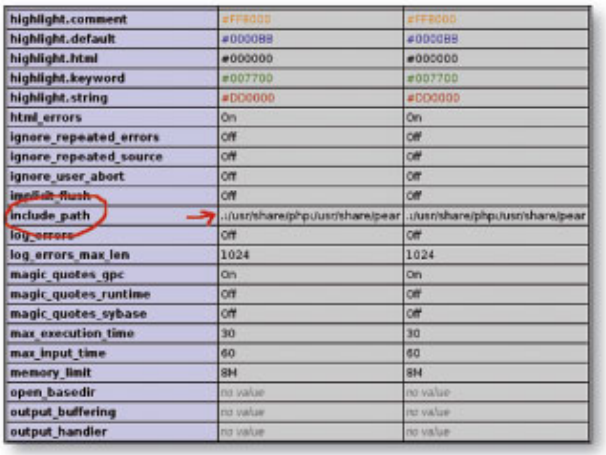
Layer).
Qu'est-ce que cela signifie exactement ? Et bien, en PHP, lorsque vous utilisez un serveur de type mysql, les fonctions d'utilisation du serveur sont de type mysql_nomdelafonction ; si vous utilisez postgresql, les fonctions ont toutes le préfixe pg_, etc.
Or, PEAR::DB permet de s'affranchir de cela. Ainsi les noms de fonctions sont les mêmes pour tous les serveurs supportés par PEAR::DB (dbase, mysql, odbc, sqlite, pgsql, etc.). En fonction de ce que vous déclarez dans le DSN (voir plus loin), PEAR::DB connaît le serveur utilisé et utilise les fonctions correspondantes.

Téléchargement et installation

PEAR::DB se trouve dans la section « Database » de la liste des librairies. Un clic sur le lien « 1.7.6 » nous invite à télécharger l'archive nommée DB-1.7.6.tgz.
L'onglet Download nous informe également que la librairie DB dépend de PEAR Installer. Nous allons donc récupérer aussi l'archive correspondante (section « Pear » ; archive nommée PEAR-1.4.2.tgz). Ces deux archives sont téléchargées dans un répertoire temporaire.
Nous allons à présent les décompresser. Nous obtenons alors deux répertoires, nommés DB-1.7.6 et PEAR-1.4.2, contenant chacun quelques fichiers et répertoires. Les seuls éléments qui nous importent sont : le répertoire DB/ et le fichier DB.php, ainsi que le répertoire PEAR/ et le fichier PEAR.php (les autres éléments sont des fichiers de documentations ou d'exemples).
Nous déplaçons ces 4 éléments dans notre répertoire de librairies (/var/www/includes/). Puis, nous allons modifier la directive include_path.
En effet, si vous avez la curiosité de vérifier le contenu du fichier phpinfo.php (fichier à créer si ce n'est déjà fait, puis ouvrez la page http://www.mon_site.com/phpinfo.php) :

```
<?php
phpinfo()
?>
```

Celui-ci comporte un tableau contenant, entre autres, la liste des directives PHP (Fig. 2). À la ligne include_path, vous pouvez déjà lire certains chemins (par exemple : .:/usr/share/php, etc.). Il faudra donc, dans votre script PHP, ajouter le chemin vers les librairies que l'on vient d'installer.



highlight.comment	#FF8000	#FF8000
highlight.default	#0000BB	#0000BB
highlight.html	#000000	#000000
highlight.keyword	#007700	#007700
highlight.string	#DD0000	#DD0000
html_errors	On	On
ignore_repeated_errors	Off	Off
ignore_repeated_source	Off	Off
ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	.:/usr/share/php:/usr/share/pear	.:/usr/share/php:/usr/share/pear
log_errors	Off	Off
log_errors_max_len	1024	1024
magic_quotes_gpc	On	On
magic_quotes_runtime	Off	Off
magic_quotes_sybase	Off	Off
max_execution_time	30	30
max_input_time	60	60
memory_limit	8M	8M
open_basedir	no value	no value
output_buffering	no value	no value
output_handler	no value	no value

Fig. 2

Création d'une base de donnée

Pour illustrer l'utilisation de la librairie DB, nous allons créer rapidement une base de données fictive, à l'aide de PhpMyAdmin. Je crée une base appelée « ~~base_test~~ », contenant une table appelée « ~~mytable~~ ». Celle-ci contient 4 champs : ~~id~~, ~~nom~~, ~~prenom~~, ~~age~~. Cette table comporte 4 enregistrements.

Script PHP permettant d'interroger la base de données

À présent, nous allons créer un script, utilisant la librairie DB, qui va nous permettre d'interroger la base de données que nous venons de créer. Ce script est le suivant :

```
<?php
ini_set("include_path", '/var/www/includes/:' . ini_get("include_path"));

require_once "DB.php";

$db = DB::connect('mysql://toto:titi@localhost/base_test');
if (DB::isError($db)) {
    die($db->getMessage());
}

$res = $db->query('SELECT * FROM mytable');
$numrows = $res->numRows();
echo "Nombre de résultats: " . $numrows . "<br>";

while ($row = $res->fetchRow(DB_FETCHMODE_ASSOC)) {
    echo "<pre>";
    print_r($row);
    echo "</pre>";
}
```

```
?>
```

- Tout d'abord, on modifie la directive include_path, à l'aide de la fonction ini_set(), qui permet de changer la valeur d'une option de configuration. La nouvelle valeur de include_path sera prise en compte durant toute l'exécution du script. La directive include_path reprend sa valeur par défaut dès la fin du script. La fonction iniget(), quant à elle, retourne la valeur de l'option de configuration. La concaténation des deux (via le point) permet de prendre en compte les valeurs précédentes ET cette nouvelle valeur.
- Puis, on appelle la librairie DB à l'aide de la commande require_once.
- La fonction DB::connect() permet de se connecter à une base de données. Elle nécessite un DSN (Data Source Name) en tant que premier paramètre, qui peut être soit une chaîne, soit un tableau. Sous forme de chaîne, les paramètres de connexion sont à déclarer de la façon suivante :

  ```
  DB::connect(type_de_base://username:password@protocole+hostname/nom_de_la_base)
  ```

  Cette fonction retourne un nouvel objet DB ou un objet DB_Error en cas d'erreur.

- On teste ensuite si $db est une erreur de type DB, auquel cas un message d'erreur est envoyé et le script est interrompu. La fonction DB::isError() détermine si une variable est un objet DB_Error ou non.
- On applique ensuite la méthode query() sur $db (qui est un objet DB). La méthode query() admet en paramètre une requête SQL. On demande ici à sélectionner tous les enregistrements de la table « mytable ».
- DB_result::numRows() retourne le nombre d'enregistrements qui correspondent à la requête(ou un objet DB_Error en cas d'erreur). On affecte à la variable $numrows la valeur qui est retournée, valeur qui apparaîtra donc à l'écran via la commande echo.
- Nous entamons ensuite une instruction conditionnelle du type « tant que (condition) exécute les instructions ». La fonction DB_result::fetchRow() retourne une ligne de données depuis la liste des résultats (représentée ici par $res), puis déplace le pointeur de résultats à la ligne suivante.

Cette fonction peut admettre comme paramètre le mode de récupération à utiliser et le numéro de la ligne à récupérer. Ce n'est pas indispensable pour notre exemple, mais nous précisons le mode de récupération : DB_FETCHMODE_ASSOC qui permet d'associer les clés du tableau (assimilables à des titres de colonnes) aux noms des champs. Sans cette précision, les clés du tableau sont simplement des numéros (ce qui est moins explicite à l'affichage).
La valeur retournée, assignée ici à la variable $row, peut être un tableau ou un objet contenant la ligne de données, NULL lorsque l'on a atteint la fin de la liste des résultats ou un objet DB_Error en cas d'erreur. La fonction print_r() permet d'afficher à l'écran, la valeur de $row.
La boucle s'arrête lorsque la variable prend NULL pour valeur, c'est-à-dire lorsque tous les résultats de la requête ont été affichés et qu'il n'y a plus de ligne.

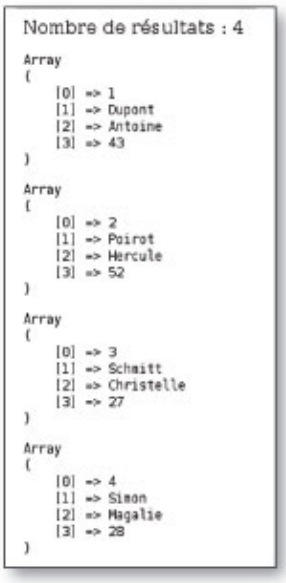


Fig. 3

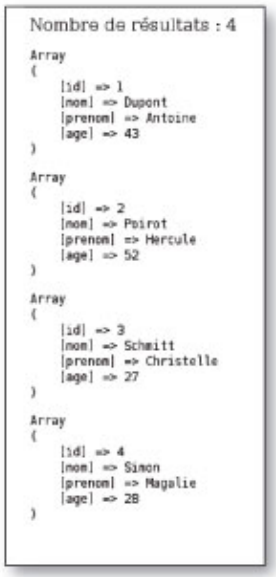


Fig. 4

Conclusion

Comprenez bien que les fonctions ~~DB::connect()~~, ~~DB::isError()~~, ~~numRows()~~ et ~~fetchRow()~~, sont des fonctions propres à la libraire DB, spécialisées dans l'exploitation des bases de données. Elles ne sont utilisables que si vous avez bien téléchargé et installé la librairie et que vous avez correctement renseigné la variable ~~include_path~~.

Il existe beaucoup d'autres packages PEAR, permettant d'utiliser des fonctions prédéfinies, diverses et variées, qui « soulagent » grandement le travail des développeurs web. Ceux-ci n'ont en effet plus à développer chaque petite fonction dont ils ont besoin (pourquoi réinventer la roue ?) et peuvent ainsi se concentrer sur l'essentiel.

Lien :

- Manuel (très détaillé !) d'utilisation de PEAR et des librairies : http://pear.php.net/manual/fr/

~~Retrouvez cet article dans : Linux Pratique 32~~

Posté par Fleur Brosseau (Fleur) | Signature : Fleur Brosseau | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- Anti-forensics sur systèmes de fichiers ext2/ext3
- Proposer plusieurs CSS en fonction du navigateur
- Des plugins pour vos blogs !
- Anonymisation
- Canaux cachés (ou furtifs)
- FON : le grand partage du Wifi !
- MakeHuman : interview de l'équipe de développement
- La stéganographie moderne
- La protection du secret : approche juridique

- Mieux connaître OOo Draw : les objets 3D et leur espace

Actuellement en kiosque :

- # Il y a actuellement

- 751 articles/billets en ligne.

Recherche

- # Catégories

- Administration réseau
  - Administration système
  - Agenda-Interview
  - Audio-vidéo
  - Bureautique
  - Comprendre
  - Distribution
  - Embarqué
  - Environnement de bureau
  - Graphisme
  - Jeux
  - Matériel
  - News
  - Programmation
  - Réfléchir
  - Sécurité
  - Utilitaires

- Web

Archives

- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 108
- GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 38
- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°49
- À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49
- Linux Pratique Essentiel N°3 - Août/Septembre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°3

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse

- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux