Apprendre la programmation

THE HACKADEMY PROG

Juillet - août 2006 / n°7 / 6 euros

100 % Pratique Technique Unique

PHP

Sessions
Upload de Fichiers
Filtrage
Logs
XML

Exemples pratiques
Linux / Windows

Jonglez avec Apache, MySQL et PHP

En vente en kiosque

THE HACKADEMY PROG
st édité par DMP,

26 bis rue Jeanne d'Arc
94160 St-Mandé
Tél.: 01 53 66 95 28

Rédacteur en chef : dvrasp

Conception graphique : Weel

Directeur de la Publication et représentant légal :
O. Spinelli

Principal associé : DMP

IMPRIMÉ EN FRANCE
(PRINTED IN FRANCE)
par Roto Garonne
ZA "Mestre-Marty" 47310
Estillac
Commission paritaire
en cours
ISSN en cours

thehackademy.net

Sommaire

Introduction texte? p.4

Installer PHP, Apache et MySQL sous Windows p.6

Methodes de developpement en PHP p.12

Sessions PHP : créer un espace membres p.16

Opérations sur les fichiers en PHP p.20

Logging et filtrage : blindez votre site ! p.28

Créez un fichier XML en PHP p.34

Les expressions régulières en PHP p.37

Moteur de templates p.40

Simuler register_globals=on ? Coder compatible et sécurisé p.44

PHP dans votre shell ! p.48

Coder des GUI en php p.51

Développement PHP/MySQL accéléré p.52

thehackademy.net

Retour aux sources

Il peut vous paraître étonnant de revenir sur de la programmation Web bas niveau en PHP, juste après avoir consacré un numéro complet aux gestionnaires de contenu, qui justement vous épargnent cette peine. Cette collection est cependant destinées à l'apprentissage de la programmation, sous tous ces aspects. Et en effet, la programmation ne peut être considérée sérieusement qu'à un seul niveau d'abstraction. Cet art réside dans la combinaison d'éléments appartenant à des mondes différents : les opérations arithmétiques, logiques et sur la mémoire permettent de compose des fonctions ; celles-ci peuvent être juxtaposées pour façonner des classes ; ces classes sont aussi souvent intégrées dans des modules, que n'importe quel programme peut utiliser. À leur tour, ces programmes peuvent permettre de réaliser des tâches plus complexes, combinés à d'autres (je pense par exemple à des agrégateurs RSS). Et à l'autre bout, il ne faut pas oublier que PHP est écrit en C, et que son interprétation fait intervenir toutes sortes d'autres objets et d'abstractions avant d'atteindre le processeur.

Dans ce numéro, nous vous présentons le minimum requis pour pouvoir vous débrouiller en PHP. Vous apprendrez à manipuler les fichiers, les sessions HTTP, le filtrage des paramètres, le logging, le XML, les expressions régulières et les templates. Afin d'élargir votre vision de ce langage, nous vous donnons également quelques conseils généraux sur la gestion d'un projet de développement Web. Nous avons aussi voulu vous parlez de certains aspects méconnus de PHP, notamment son existence en tant que langage à part entière, hors d'un environnement Web. Enfin, nous vous livrons une introduction à PEAR, une base très utile de bibliothèques PHP, issues de la communauté et pourtant très cohérentes entre elles, que nous vous conseillons vivement d'essayer.

N'oubliez pas : pour apprendre un langage tel que celui-ci, il faut tout autant le pratiquer que lire le code écrit par les personnes expérimentées qui l'affectionnent aussi. Pas d'excuse : vous aurez bientôt tout en main pour vous y mettre !

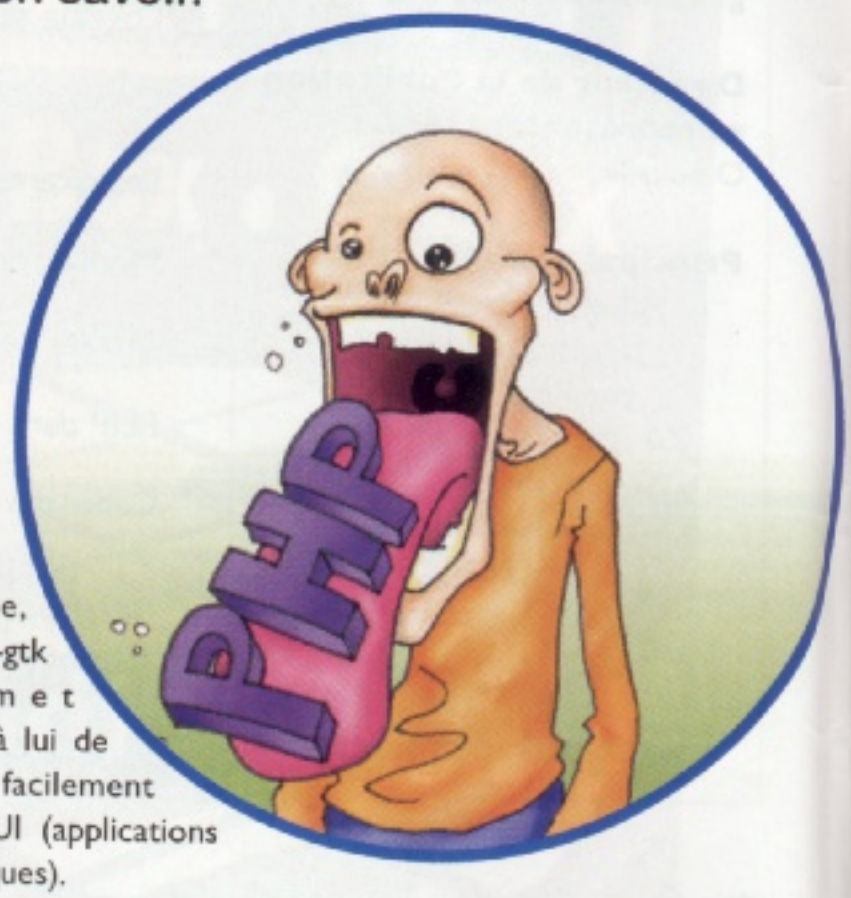
Introducti

Beaucoup d'entre vous ont certainement déjà fait du PHP sans réellement savoir ce que ce langage représentait. Dans cet article, nous allons faire un tour d'horizon sur tout ce que vous devez en savoir.

PHP est un langage de script simple et accessible à tous, relativement jeune puisqu'il fut créé en 1994 par Rasmus Lerdorf. À l'origine, Rasmus Lerdorf a réalisé ce langage afin de générer des statistiques à propos des personnes consultant son CV sur son site web. Le sigle PHP signifie hypertext preprocessor, mais initialement, on pouvait interpréter PHP pour personal home page. Vous l'avez compris PHP, est principalement destiné à la programmation de sites internet dynamiques.
La force de ce langage se trouve dans sa disponibilité pour un large éventail de plates formes telles que Windows, Linux ou autres *BSD, mais aussi pour sa gestion d'une quantité importante de base de données avec lesquelles il peut travailler. Mysql est sans doute la base de donnée la plus utilisée avec PHP, et c'est celle que l'on retrouve dans la plupart des ouvrages traitant de ce langage.

Aujourd'hui, PHP en est à sa version 5, implémentée de nouvelles fonctionnalités lui permettant de couvrir tous les domaines en rapport avec le Web. La gestion de la mémoire est améliorée grâce au nouveau moteur Zend2 dont les propriétés seront décrites dans les prochaines pages de ce manuel.

Dans les dernières versions de PHP, il est possible d'étendre les possibilités du langage en dehors des applications web. PHP-CLI pour Command Line Interface permet de gérer la ligne de commande pour des applications en mode console, et php-gtk permet quant à lui de créer facilement des GUI (applications graphiques).



Le principal intérêt de PHP vient du fait que son code est exécuté directement par le serveur, tout comme les CGI. De cette façon, un script PHP peut devenir aussi puissant qu'un CGI réalisé en Perl ou en C.

Chaque requête HTTP va être interprétée afin de répondre à un besoin défini par le site, puis PHP produira un code HTML différent selon les paramètres de cette requête. Il est important de noter que tout comme le JavaScript, PHP s'intègre directement au HTML. PHP apporte donc un gain de temps par rapport aux CGI pour lesquels il est toujours nécessaire d'écrire plusieurs lignes de code avant de pouvoir y intégrer du HTML.
Après avoir été exécuté, le code PHP est retiré de la page pour laisser place aux fichiers HTML produits par les scripts, contrairement au JavaScript dont l'exécution est côté client (votre navigateur), ce qui justifie que son code soit accessible dans la source de votre site internet.
La syntaxe du PHP est très simple à mémoriser et est semblable à celle du Perl ou du C. De plus, il est possible d'utiliser les expressions régulières grâce aux fonctions ereg*(), ce qui plaira certainement aux adeptes du Perl .

Hello Word !

La syntaxe du PHP est très proche de celle du C ou du Perl. Une fois votre serveur web configuré, un fichier .php contenant ce code suivant vous montrera que tout est correctement installé.

```
<?php
  // Hello World en PHP
  echo 'Hello World!';
?>
```

ion texte?

Pour illustrer la syntaxe et avoir un avant-goût du PHP, voici un petit exemple de code extrêmement simple destiné à vous montrer à quoi ressemble ce fameux langage :

Pour vous convaincre d'utiliser PHP, récapitulons certains de ses points fort :

- simplicité d'utilisation et d'apprentissage,
- un langage OpenSource,
- utilisable avec un grand nombre de base de données,
- une installation simple,
- un grand nombres de fonctions et de librairies,
- des performances élevées,
- et bien d'autres ...

Il existe des applications assez conséquentes qui ont été réalisées en PHP. On retrouve par exemple OScommerce qui est une application permettant de créer des boutiques en ligne, il y a également PHPMyAdmin et PHPNuke ainsi que beaucoup, beaucoup d'autres applications , comme les nombreux gestionnaires de contenu (WordPress, Dotclear, SPIP, Joomla, etc.) que nous vous présentions dans notre précédent numéro.
Aujourd'hui la quasi-totalité des hébergeurs mettent à disposition PHP dans leurs offres et nous allons voir dans ce manuel qu'il est très simple d'installer et configurer soi-même PHP avec Apache et Mysql, que ce soit sous Linux ou Windows.

En conclusion on peut dire que PHP est un langage de référence dans l'écriture de pages webs dynamiques, et le fait qu'il existe environ 17 millions de domaines utilisant cette technologie confirme son importante popularité.

Delete

les sites à retenir

http://fr.php.net/
Site officiel du langage, c'est aussi l'endroit où est centralisée toute la documentation de référence. De nombreux sous-domaines proposent aussi des informations plus spécialisées ou destinées à la communauté À voir notamment : news.php.net, talks.php.net, pear.php.net ou bugs.php.net.

http://www.afup.org/
L'Association Française des Utilisateurs de PHP a pour but de promouvoir l'utilisation de ce langage auprès des professionnels et de participer à son développement. Elle organise un Forum annuel à Paris. Ce site est surtout utile pour ces news, et pour ses annuaires.

http://www.nexen.net/
Nexen est l'un des principaux portails francophone pour les utilisateurs de PHP : des news, des ressources, de la documentation, des articles, des tutoriels, etc.

http://php.developpez.com/
La section PHP de developpez.com est sans doute l'une des plus fournies. Vous y trouverez des tutoriels, des outils et de nombreux informations utiles.

http://www.phpdebutant.org/
Pour les débutants, ce site propose une petit cours d'introduction en 24 points, ainsi que de nombreux autres articles de niveau très accessible.

Installer PHP, Ap sous Windows

Dans cet article, nous allons voir comment installer et configurer PHP5 avec Apache2 et MySQL4.1 sur Windows. Le but de cet article est d'expliquer les différentes options et de comprendre le PHP dans les meilleures conditions.

Introduction

Le couple PHP/MySQL s'est très développé sur Internet au cours de ces dernières années et beaucoup de personnes commencent leur apprentissage du langage PHP sous Windows. La plupart des débutants utilisent des programmes d'installation automatique tels que Easyphp et certains d'entre eux ne connaissent pas la valeur des fichiers de configuration. Afin d'éviter ce genre de désagréments, nous allons expliquer les étapes d'installation d'un serveur Apache/PHP/MySQL ainsi que les options de configuration les plus importantes. Il est toujours plus intéressant de connaître les grandes lignes du php.ini pour pouvoir tirer partie de tous les avantages du PHP.

1 - MySQL

MySQL est un système de gestion de bases de données relationnelles fiable et facile à utiliser, disponible depuis 1996. Il a gagné en popularité grâce à PHP avec lequel il est utilisé pour la conception de sites Internet dynamiques. La majorité des applications PHP manipulent des informations (listes d'utilisateurs, ensemble de produits, etc.). Avec MySQL, nous pourrons gérer ces informations plus efficacement.

Vous pouvez obtenir MySQL depuis son site officiel www.mysql.com, choisissez la dernière version pour Windows. Après avoir téléchargé et décompressé l'archive, vous obtiendrez un fichier Setup.exe. Exécutez-le.

L'installateur vous demande en premier lieu le type d'installation que vous désirez, nous choisirons " Custom " pour personnaliser l'installation. Nous aurons besoin d'au moins 21 mo d'espace disque et il faudra créer un répertoire "". La seule option à modifier est l'emplacement d'installation, choisissons " c:\serveur\mysql ". Pour le reste, il suffit d'appuyer sur " Next " jusqu'à ce que vous arriviez à la dernière étape. Si tout s'est bien passé, vous devez obtenir l'écran suivant : L'installateur nous demande ensuite si nous avons un compte sur le site Mysql. Cette option n'étant pas obligatoire, vous pouvez vous en passer en mettant " Skip Sign-Up ".

Mysql est maintenant installé correctement et il ne nous reste plus qu'à le configurer. Pour cela, un assistant de configuration est lancé automatiquement après l'installation.

Les options proposées par défaut sont correctes pour une utilisation personnelle, on peut alors se contenter de cliquer sur " Next " jusqu'à l'écran de configuration des utilisateurs. Cette section est très importante car si vous n'ajustez pas correctement les droits des utilisateurs, vous vous exposez à des problèmes de sécurité.

Il faut impérativement indiquer un mot de passe pour l'utilisateur root et cocher la case " Root may only connect from localhost ". La dernière option ne doit pas être validée, sinon n'importe quelle personne pourra se connecter au serveur et découvrir les informations contenues dans vos bases de données.

La dernière étape de cet assistant consiste à valider les options choisies, après cela nous pourrons nous connecter

EasyPHP

EasyPHP vous permet d'installer un système Apach/MySQL/PHP sans effort, idéal pour le débutant qui veut s'essayer au PHP sans avoir à rentrer dans les détails de ces serveurs complexes.

Une fois installé à partir de http://easyphp.org, un raccourci permettant de lancer Apache et MySQL apparait sur le bureau. Lorsque le système est lancé, une icône appraît dans le systray. De là vous pouvez lancer modifier la configuration de votre serveur.

ache et MySQL

à nos bases de données et modifier manuellement la configuration en éditant le fichier " C:\serveur\mysql\my.ini " qui vient d'être créé.
Votre serveur de base de donnée MySQL est maintenant installé ; un nouveau service Windows se lancera au démarrage et se mettra en écoute sur le port 3306 (le port de MySQL). Si vous êtes la seule personne à accéder aux bases de données, je vous conseille de configurer votre firewall pour qu'il refuse les connexions externes vers le port 3306. On peut également empêcher les connexions externes depuis les tables " host " et " user " de la base MySQL, cependant en utilisant le firewall on évite que des exploits soient lancés contre notre base de données.
Testons maintenant notre configuration. Pour cela il faut ouvrir une invite de commandes MS-DOS et taper les commandes suivantes :

```
C:\>cd serveur\mysql\bin
C:\serveur\mysql\bin>mysql.exe -u root -p
Enter password: **********
Welcome to the MySQL monitor.  Commands end
with ; or \g.
Your MySQL connection id is 4 to server ver-
sion: 4.1.7-nt

Type 'help;' or '\h' for help. Type '\c' to
clear the buffer.

mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)

mysql> quit
Bye
```

Si vous tentez de lancer la commande " mysql.exe " depuis un répertoire différent de " C:\serveur\mysql\bin> ", vous aurez une erreur.

```
C:\>mysql.exe
'mysql.exe' n'est pas reconnu en tant que commande
interne ou externe, un programme exécutable ou un fichier
de commandes.
```

Pour éviter cela et pouvoir utiliser les exécutables de mysql depuis n'importe quel répertoire, vous devez modifier la variable d'environnement PATH. Vous pouvez le faire depuis la ligne de commande :

```
C:\serveur\mysql\bin>set
PATH=%PATH%;c:\serveur\mysql\bin
C:\serveur\mysql\bin>cd \
C:\>mysql -V
mysql  Ver 14.7 Distrib 4.1.7, for
Win95/Win98 (i32)
```

Lors de l'installation, deux bases de données sont créées : "mysql" et "test". La première est très importante. C'est elle qui va nous permettre d'ajouter des utilisateurs et de gérer les droits sur les autres bases de données, ne l'effacez surtout pas.
Actuellement nous sommes obligés d'utiliser le compte root pour nous connecter aux bases de données, ce qui est dangereux et peut compromettre la sécurité du système. Il faut donc ajouter un nouvel utilisateur dont on se servira pour se connecter avec les scripts PHP. Vous pouvez créer cet utilisateur en insérant une nouvelle valeur dans la table " user " fournie avec la base de données " mysql ".
Il reste une option que l'on peut modifier, il s'agit de la directive " datadir " du fichier de configuration " c:\serveur\mysql\my.ini ", qui contient l'emplacement des bases de données. Il est plus sûr d'utiliser un répertoire situé sur une autre partition.

2 - Apache

Apache est un serveur web Open Source très puissant. Depuis sa création, le nombre de serveurs l'utilisant ne cesse d'augmenter et il est devenu le serveur web le plus populaire du monde. Bien qu'Apache soit déjà, par défaut, un serveur très complet, il est possible d'étendre ses fonctionnalités en ajoutant des modules tels mod_ssl ou mod_rewrite. Ce n'est pas avec ces quelques lignes que vous pourrez vous rendre compte de toute sa puissance étant donné que des livres entiers ont été écrits dans ce but. L'idée que vous devez retenir est que, grâce à lui, vous pourrez publier vos données sur Internet ;) Pour obtenir plus d'informations à propos d'Apache, je vous invite à visiter son site officiel http://www.apache.org.
Nous installerons la dernière version disponible que l'on peut obtenir directement à partir du lien :
http://mir2.ovh.net/ftp.apache.org/dist/httpd/binaries/win32/apache_2.0.58-win32-x86-no_ssl.msi (l'extension .msi contient une version déjà compilée d'Apache, ce qui facilite son installation).

Après avoir lancé notre installateur et accepté la licence (que vous aurez pris le temps de lire...), nous arrivons à la configuration de notre domaine. Le premier champ, " Network Domain ", correspond à votre DNS, le second, " Server Name ", au nom de votre site. Le dernier champ permet de définir le mail de l'administrateur du site.
Lors de l'étape suivante, il faut choisir son type d'installation. Comme pour MySQL, choisissez " Custom ". Ensuite il faut changer le répertoire d'installation et mettre "" à la place. Rien de particulier pour la suite, contentez-vous d'appuyez sur " Next " et de valider l'installation. Voilà, le serveur est en écoute sur le port 80, ce que nous pouvons vérifier en nous connectant sur http://localhost/ où nous verrons la page suivante s'afficher :

Le code de la page qui s'affiche se trouve dans " c:\serveur\Apache2\htdocs ". Vous pouvez effacer les fichiers contenus dans ce répertoire et créer un nouveau fichier " index.html ". Si vous actualisez la page, vous verrez apparaître le contenu de votre fichier nouvellement créé.
Vous avez sans doute remarqué l'apparition d'une nouvelle icône dans la barre de tâche, elle correspond à l' " Apache Service Monitor " qui permet de gérer directement Apache en offrant la possibilité de lancer, redémarrer ou éteindre le serveur.
Pour avoir des informations sur votre version d'Apache ainsi que sur la façon dont il a été compilé, rendez-vous dans le répertoire " C:\serveur\Apache2\bin " et lancez la commande suivante :

```
C:\serveur\Apache2\bin>Apache.exe -V
Server version: Apache/2.0.52
Server built:   May 14 2006 18:28:28
Server's Module Magic Number: 20020903:9
Architecture:   32-bit
Server compiled with....
 -D APACHE_MPM_DIR="server/mpm/winnt"
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D HTTPD_ROOT="/apache"
 -D SUEXEC_BIN="/apache/bin/suexec"
 -D DEFAULT_SCOREBOARD=
                "logs/apache_runtime_status"
 -D DEFAULT_ERRORLOG="logs/error.log"
 -D AP_TYPES_CONFIG_FILE="conf/mime.types"
 -D SERVER_CONFIG_FILE="conf/httpd.conf"
```

L'avantage d'Apache est qu'il utilise un unique fichier de configuration, qui se trouve dans " c:\serveur\Apache2\conf " et se nomme httpd.conf. Ce fichier est composé de trois sections distinctes. Nous allons donc voir quelles sont les principales directives caractérisant chacune de ces sections.

Section 1 : Cette section permet de configurer l'environnement global d'Apache. On y trouve les directives indiquant l'emplacement des fichiers de configuration et celles contrôlant le fonctionnement général des processus.

```
ServerRoot      "C:/serveur/Apache2"
```

-> Cette directive indique le dossier d'installation, c'est le plus haut niveau de l'arborescence d'Apache, on y trouve les fichiers de confs, les logs, la documentation etc. Les autres directives font référence à ServerRoot pour les chemins relatifs.

```
Listen          80
```

-> Permet de définir le port sur lequel Apache reçoit et traite les connexions. Vous pouvez également ajouter une adresse IP en plus du numéro de port afin qu'Apache utilise une interface réseau spécifique.

```
LoadModule nom_module modules/nom_module.so
```

-> À partir de cette directive, vous pouvez ajouter ou supprimer des modules afin d'étendre les capacités du serveur. Pour connaître la liste des modules disponibles, il suffit de lister le répertoire "modules/" .
Vous n'aurez certainement pas besoin de modifier les autres directives de cette section, ce n'est donc pas la peine de les décrire.

Section 2 : Cette section contient les directives définissant les paramètres du serveur par défaut. Cela correspond au site qui s'affiche si l'on n'utilise pas d'hôtes virtuels.

```
ServerAdmin   admin@nom_du_site.com
```

-> La directive ServerAdmin définit l'adresse de la personne gérant le domaine. Cette adresse est utile pour prévenir l'administrateur s'il y a des problèmes d'accès à certaines pages.

```
ServerName    nom_du_site:80
```

-> Définit l'adresse du site principal, si vous n'avez pas de nom de domaine, vous pouvez alors indiquer le nom de votre machine ou l'IP correspondante.

```
DocumentRoot " C:/serveur/Apache2/htdocs"
```

-> Cette directive donne le chemin d'accès aux fichiers du site principal. Comme pour la directive " datadir " de MySQL, je vous conseille d'utiliser un répertoire situé sur une autre partition afin d'éviter d'éventuels problèmes de sécurité.

```
DirectoryIndex index.html index.htm
```

-> Cette option permet d'indiquer la priorité des pages qui seront reconnues automatiquement. Par exemple, si vous utilisez un fichier index.html dans le répertoire " htdocs/ ", vous n'avez pas besoin d'indiquer son nom pour y accéder. Ainsi, http://localhost/ lancera automatiquement http://localhost/index.html.

```
ServerSignature        Off
```
-> Par défaut, cette directive est sur " On " et elle permet d'afficher la version du serveur lorsque l'on tombe sur une page d'erreur. Il vaut mieux la désactiver et donner le moins d'informations possibles sur votre serveur.
Il existe beaucoup d'autres directives intéressantes dans cette section, mais je ne peux pas toutes les décrire ici, vous apprendrez vite à les adapter à votre environnement.

Section 3 : C'est ici que vous pourrez configurer vos hôtes virtuels. Etant donné que vous n'aurez certainement pas besoin de gérer plusieurs domaines, je ne vais pas m'attarder sur les différents éléments de configuration de cette section.

Nous avons à présent un serveur web ainsi qu'un serveur de base de données, ces deux éléments fonctionnant indépendamment. PHP va établir un lien entre chacun de ces serveurs car il utilisera MySQL et sera lui-même utilisé par Apache.

3 - PHP

Pour obtenir une description du langage PHP, je vous laisse tourner quelques pages de ce numéro. Il y a plusieurs façons d'installer PHP, on peut par exemple utiliser l'installateur Windows disponible dans la section download de php.net. Cette méthode est la plus simple, mais elle risque de vous poser problème si vous souhaitez ajouter des extensions à PHP. L'autre technique consiste à installer PHP manuellement, ce qui est plus compliqué et demande plus de temps mais, à l'arrivée, on obtient de meilleurs résultats. De plus, en installant manuellement, on comprend mieux les différentes étapes et il sera alors plus simple de modifier la configuration.
Pour commencer l'installation, il faut récupérer l'archive contenant les binaires Windows disponible à partir de http://www.php.net/downloads.php. Une fois votre archive récupérée, il faut extraire son contenu vers " c:\php5\ " .
Nous allons maintenant installer PHP en tant que module Apache. Pour cela, il faut à nouveau éditer le fichier " httpd.conf ". La première chose à modifier se situe dans la section 1 de ce fichier, où il faut ajouter " LoadModule php5_module "c:/php5/php5apache2.dll " à la suite des autres directives du même type. De même, il est nécessaire d'ajouter les directives " AddType application/x-httpd-php .php " et " PHPIniDir "C:/php5 " dans la section 2. Pour finir, il faut ajouter une valeur à la directive DirectoryIndex afin que les fichiers index.php soient automatiquement reconnus. Nous aurons donc

```
" DirectoryIndex
index.html index.htm
index.php ".
```

Retournons maintenant dans notre répertoire d'installation de PHP pour renommer le fichier php.ini-recommended en php.ini. Il est préférable d'utiliser php.ini-recommended plutôt que php.ini-dist car sa configuration est optimisée et plus orientée sécurité. Bien, PHP est enfin installé. Pour s'en assurer, nous pouvons créer un fichier " test.php " dans lequel nous placerons :

```
<?php
echo "Php fonctionne :D !<br><br>";
phpinfo();
?>
```

Il reste à redémarrer Apache puis à tester notre script sur http://localhost/test.php.

Notre configuration actuelle ne nous permet pas d'utiliser MySQL. Pour remédier à ce problème nous devons ajouter l'extension MySQL, ce que nous verrons en même temps que la description du " php.ini ". Nous allons maintenant voir comment est composé le fichier de configuration de PHP car c'est à partir de celui-ci que nous pourrons ajouter de nouvelles extensions. Lancez donc votre éditeur de texte favori puis ouvrez le fichier php.ini qui doit être placé dans " C:\php5\ ". Tout comme le fichier de configuration d'Apache, php.ini est composé de plusieurs sections traitant de la configuration du langage, des ressources utilisées, de la façon dont sont traitées les variables ou de l'utilisation et de la configuration des extensions dynamiques. Passons en revue quelques-unes des options les plus importantes :

```
safe_mode = Off
```
-> Le safe_mode permet d'ajouter des options de sécurité en désactivant certaines fonctions. Si vous débutez en PHP, je vous conseille de ne pas l'utiliser tout de suite car la plupart des documentations se basent sur des configurations n'utilisant pas cette option. Pour avoir plus d'informations sur le safe_mode, vous pouvez aller visiter http://www.php.net/manual/en/features.safe-mode.php .

```
max_execution_time = 30
```
-> Détermine le temps d'exécution maximum en seconde d'un script. Vous aurez peut-être besoin de changer cette valeur en programmant diverses applications.

```
register_globals = Off
```
-> Cette option est très importante car, suivant sa valeur, certaines applications ne fonctionneront pas. Il est préférable pour la sécurité de son serveur de maintenir cette option sur Off et de prendre la bonne habitude d'utiliser le préfixe $_ afin de traiter les valeurs passées en argument.

```
magic_quotes_gpc = On
```
-> Permet d'échapper les caractères spéciaux par un slash. En activant cette option, vous éviterez des problèmes de sécurité de type SQL injection.

```
extension_dir = "C:/php5/ext/"
```
-> Cette directive sert à définir l'emplacement des exten-

sions de PHP, par défaut sa valeur est de "./" et il faut donc la modifier pour pouvoir activer le support des bases de données MySQL.

```
extension=php_mysql.dll
```

-> En décommentant cette option, on active l'extension MySQL. Vous devez auparavant avoir modifié la valeur d' " extension_dir ". Si vous souhaitez activer d'autres extensions telles que bz2 ou gd2, il suffit de décommenter chacune des directives leur correspondant.
Le fichier de configuration php.ini contient également une section de configuration pour certaines extensions, celle-ci se trouve à la fin du fichier. Vous pouvez essayer d'adapter ces valeurs afin d'obtenir de meilleures performances.
Une chose importante à ne pas oublier est de copier le fichier libmysql.dll fournit dans " c:\php5\ " vers " c:\windows\system32\ " ou " c:\WINNT\system32\ ", selon votre système d'exploitation.

Après tout cela, il reste à redémarrer Apache pour que les modifications du php.ini prennent effet. En actualisant la page de notre test.php, on voit dans le phpinfo que MySQL est maintenant disponible.
Vous disposez à présent d'un serveur web capable d'utiliser la technologie PHP. Vous pouvez perfectionner vos connaissances du PHP en ajoutant de nouvelles extensions à votre version de PHP. En apprenant à utiliser ces extensions, vous serez capable de créer et d'héberger des sites internet dynamiques très puissants.

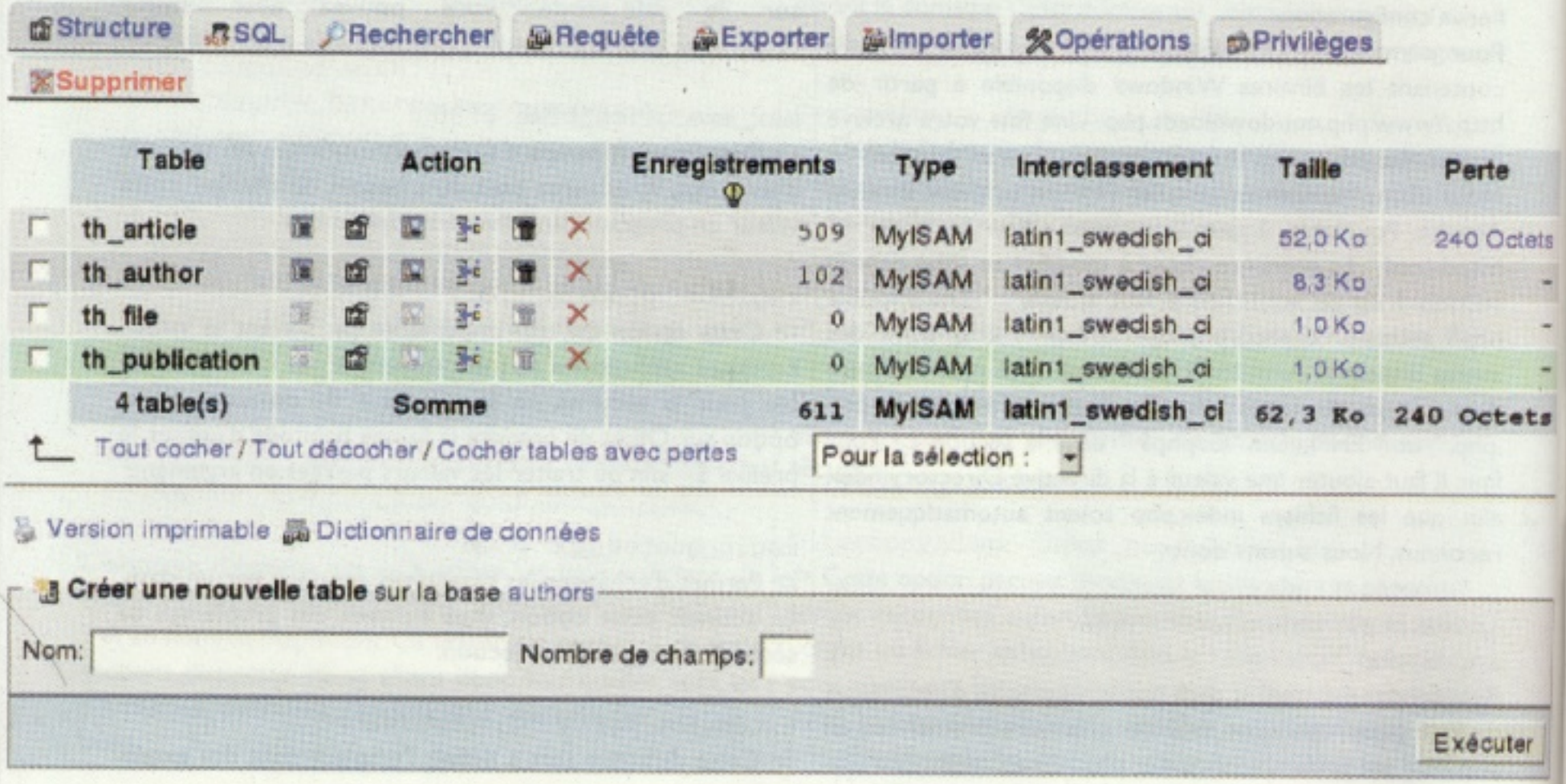
Références

" Apache en action " de Ken Coar & Rich Bowen
" PHP & MySQL " de Luke Welling & Laura Thomson
Site officiel d'Apache http://www.apache.org
Site officiel de PHP http://www.php.net
Site officiel de MySQL http://www.mysql.com

Delete

PhpMyAdmin

Pour intervenir sur vos bases de données de manière plus conviviale, nous vous conseillons d'installer phpMyAdmin, qui permet d'administrer les bases, les tables, les utilisateurs, etc. à partir de votre navigateur.
Il est installé par défaut avec EasyPHP. Pour l'installer séparément, il suffit de décompresser l'archive disponible sur http://www.phpmyadmin.net à la racine de votre serveur et d'utiliser scripts/setup.php pour configurer l'installation (copier le fichier dans le répertoire principal de votre phpmyadmin et brancher son navigateur sur l'adresse correspondante, par exemple http://localhost/phpMyAdmin/setup.php).



Installer un système LAMP

Linux/Apache/MySQL/PHP

On désigne par LAMP un système Linux, doté d'un serveur Web Apache, d'une base de donnée MySQL et, pour faire le lien, du langage de génération dynamique d'html que vous connaissez bien : PHP. Ces trois éléments sont souvent réunis. Une grande partie des sites que vous utilisez tous les jours reposent sur un tel système.

Car il s'agit de logiciels libres, sous licence GPL. Leur succès phénoménal, dû à leur gratuité mais aussi à leur qualité et fiabilité, a entraîné l'apparition d'un masse impressionnante de documentation et d'aides en ligne que les développeurs et administrateurs système consultent et enrichissent tous les jours. Ainsi, lorsque quelque chose ne marche pas, coller quelques messages d'erreur dans l'invite de Google vous aidera le plus souvent à corriger le problème en quelques minutes. Au pire, vous pourrez toujours poser une question sur un forum ou une liste de diffusion. C'est ce qui démarque le plus les technologies libres des propriéataires : même si le support centralisé de Microsoft, par exemple, est immense, il ne pourra jamais rivalisé avec les efforts conjoints d'une communauté mondiale, soudée et passionée.
C'est ce qui rend ce système particulièrement attractif pour les débutants comme pour les développeurs avancés.

Pour disposer d'un système LAMP chez soi, ou sur un serveur dédié, la méthode générale est la suivante :

1. Installer Linux (voir http://www.tlm-project.org/ pour télécharger les principales distributions avec Bit Torrent)
2. Installer Apache/MySQL/PHP. On peut le faire depuis les distributions source (http://www.ze-linux.org/howto.html?id_docs=7). Cependant, cela est fastidieux et a des chances de vous faire introduire des erreurs de configuration. Il est plus simple d'utiliser les systèmes de paquetages des différentes distributions.

Les remarques de l'article précédent sur la configuration d'Apache sont également valables pour Linux. Le fichier de configuration principal, httpd.conf, se trouve généralement dans /etc, dans /etc/apache ou dans /etc/httpd selon la distribution ; de même pour PHP, vous trouverez le php.ini dans un sous-répertoire de /etc (faire find /etc -name php.ini). Si par contre vous avez installé Apache depuis les sources, vous trouverez ces fichiers dans /usr/local/apache/conf pour Apache ou /usr/local/lib/ pour php.ini.

Quelques liens utiles :

- **Installer et obtenir Debian :**
  http://www.debian.org/releases/stable/i386/index.html.fr
- **Installer Apache/MySQL/PHP sur Debian :**
  http://www.ac-creteil.fr/reseaux/systemes/linux/installation-lamp-debian.html
- **Aide pour installer Debian :**
  http://giminik.developpez.com/articles/debian-gnu-linux/os-installation/
- **Installer et télécharger Fedora :**
  http://fedora.redhat.com/download/
- **Installer A/M/P sur Fedora (en) :**
  http://www.flmnh.ufl.edu/linux/install_apache.htm
- **Installer et obtenir Mandriva :**
  http://www.uselinuxathome.com/FRinstall.htm
- **Installer A/M/P sur Mandriva :**
  http://lea-linux.org/cached/index/Reseau-web-Apache_PHP_MySQL.html
- **Recevoir gratuitement les CD de Ubuntu**
  http://shipit.ubuntu.com/

Methodes de deve

Beaucoup d'entre vous ont certainement déjà fait du PHP sans réellement savoir ce que ce langage représentait. Dans cet article, nous allons faire un tour d'horizon sur tout ce que vous devez en savoir.

Etapes du développement d'un projet

Un projet comprend plusieurs étapes de développement, qui, dans la pratique sont souvent confondues et mélangées par les développeurs débutants ou même initiés.

a) Cahier des charges

C'est la première chose à faire : rassembler toutes les exigences que doit remplir le script (ou le site, ou ce que vous voulez faire), afin de se fixer des objectifs clairs et précis. Il sera ainsi plus facile d'avoir une idée du rendu final et de savoir à tout moment si ce que l'on fait correspond vraiment à ce que l'on veut faire, ce qui n'est pas toujours le cas. Ensuite notez toutes les idées qui vous passent par la tête sur le produit (cette phase ne s'arrête pas tant que le développement n'est pas terminé).

b) Plan du projet (structure) : conception générale

Il faut alors organiser ses idées selon un plan bien précis. Ce plan, souvent sous forme de schéma annoté, doit présenter les différentes parties de votre script et ce qu'elles font. C'est donc dans cette partie que vous dressez la structure (ou architecture) de votre script : les dossiers et les pages doivent porter des noms indiquant ce qu'ils contiennent, ainsi que des liaisons vers d'autres pages (inclusions, redirections, liens). Vous devez obtenir une arborescence commençant par la page principale de votre script (souvent index.php).
C'est aussi ici que vous définissez quels tests il vous faudra effectuer sur chaque partie.

c) Apprentissage

Si vous ne maîtrisez pas une partie du script, c'est maintenant qu'il vous faut apprendre et faire des tests jusqu'à ce que vous la maîtrisiez suffisamment pour l'exploiter sans risquer l'erreur ou le trou de sécurité. Exemple : vous faites un portail et vous ne savez pas comment envoyer un mail, vous devez donc apprendre à vous servir de la fonction mail correctement et faire des essais jusqu'à ce que vous pensiez savoir suffisamment la manier.

d) Recherche : déjà fait ?

On trouve beaucoup de scripts déjà faits sur le Web : vous pouvez donc éviter de perdre du temps à en refaire un déjà prêt en l'incluant directement dans votre projet. Il vous faut bien sûr faire attention à la licence, ou à défaut avoir l'autorisation de l'auteur et ne pas oublier de citer celui-ci dans les fichiers d'aide et d'information. Cette partie se fait après l'apprentissage, car il vous faut être en mesure de vérifier les scripts que vous intégrez. N'utilisez jamais des scripts dont vous ne comprenez pas le fonctionnement, sinon la totalité du code.

e) Conception détaillée

Là, on détaille, on peaufine, on choisit les fonctions à utiliser ou à définir, on résume rapidement le code à mettre dans chaque page et les fichiers à mettre dans chaque dossier, etc. À la sortie de cette étape, vous ne devez plus avoir qu'à transformer ce que vous avez sur le papier (enfin dans vos notes) en code pour que ça fonctionne (en théorie), tests et débuggages exclus. C'est ici également que se planifie la deuxième partie de la batterie de tests à laquelle votre script sera soumis, chaque élément (fonction, module, page) devant être testé séparément puis dans la structure.

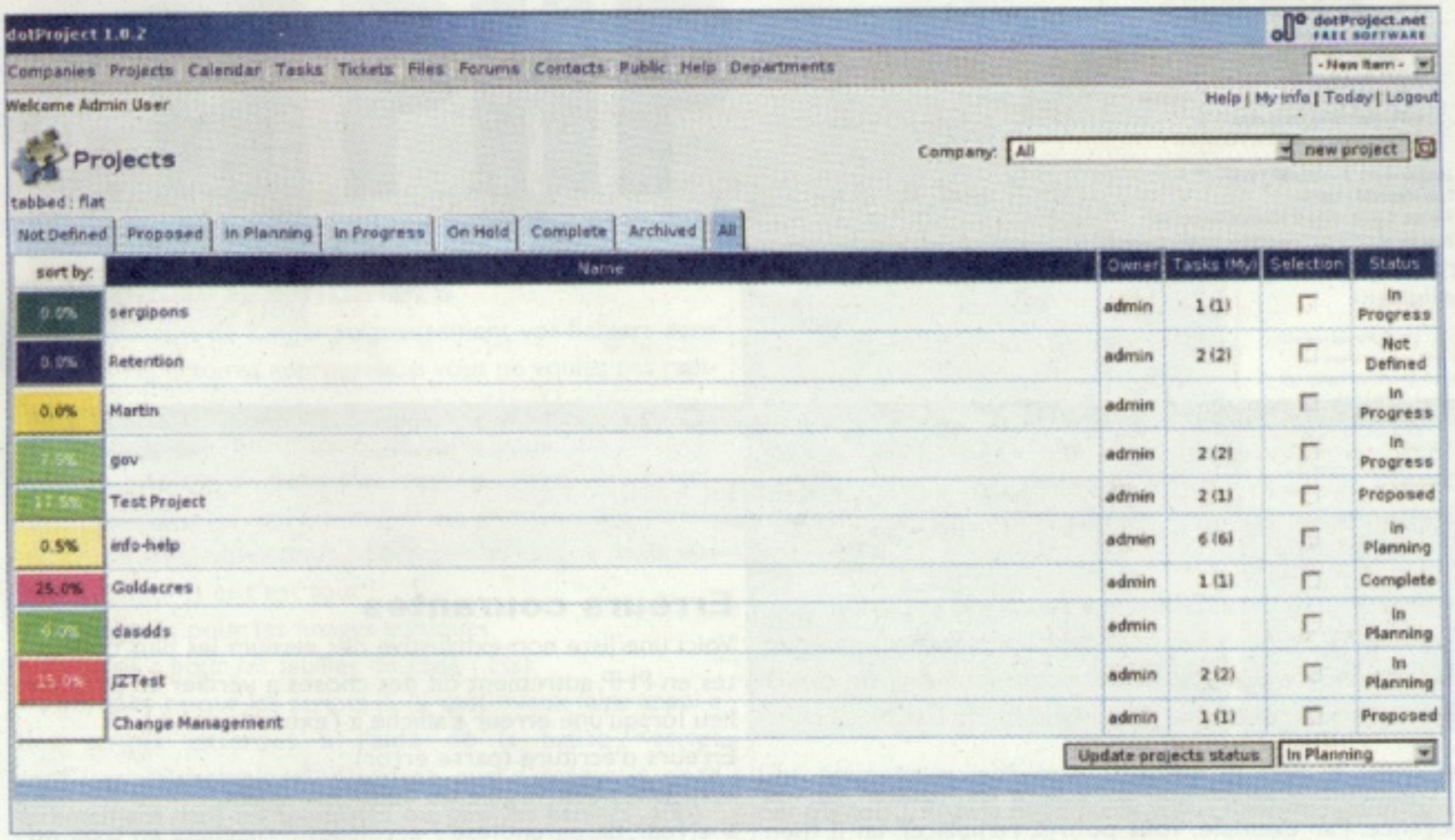
f) Code, code, code et implémentation

C'est parti ! Vous pouvez maintenant commencer à coder votre script, en n'oubliant pas de débugger au fur et à mesure (10 lignes sont plus faciles à débugger que 200), et à vous référer à l'aide en ligne dès que vous avez une incertitude. Je vous conseille de commencer par la structure de base, puis d'implémenter un par un les autres éléments et les compléments (design, finition), mais vous pouvez également choisir de commencer par le plus compliqué avant la structure de base.

g) Architecture quasi-finale & Alpha-tests 1

Quand suffisamment de parties sont prêtes (même si non-achevées), il faut les tester afin de vérifier que vous ne faites pas d'erreurs dans la structure de votre script : il vaut mieux le corriger maintenant que lorsque le code sera quasiment complet. Les tests à effectuer portent sur les différentes parties, mais aussi sur leurs interactions : vous pouvez par exemple vérifier les droits d'accès, le bon fonctionnement de l'inclusion et de la redirection de pages, celui de la création/suppression de fichiers temporaires, etc. Ce sont les alpha-tests de base, que vous aviez prévus en b et e, destinés à éprouver chaque élément seul et dans la structure ensemble, dans le maximum de cas possibles (y compris une tentative de piratage).

loppement en PHP



h) Alpha-tests 2 et bêta-tests

Une fois que les parties de base de votre script sont prêtes, vous pouvez terminer les alpha-tests : il s'agit ici de tester l'ensemble du script par tous les moyens que vous pourrez mettre en oeuvre (exemple : un audit de sécurité est toujours le bienvenu). N'hésitez pas à prendre l'utilisateur pour un crétin fini ou un petit curieux lorsque vous anticipez ses actions. Reprenez ensuite depuis l'étape f jusqu'à ce que le produit fonctionne correctement.

Vous passez alors au bêta-test : le but étant de faire tester, si possible par des personnes extérieures et dans différentes conditions, votre création afin d'en éliminer les derniers bugs et défauts. Les bêta-testeurs sont donc des utilisateurs curieux qui découvrent votre produit, et qui auront une approche forcément différente de la vôtre. Attention : même dans le cas de projets Open-Source, il ne faut pas confondre bêta-testeurs et développeurs-contributeurs : ceux-ci n'interviennent que quand la première version (au moins) du script est finie.

i) Dernier débuggage & Optimisation du code

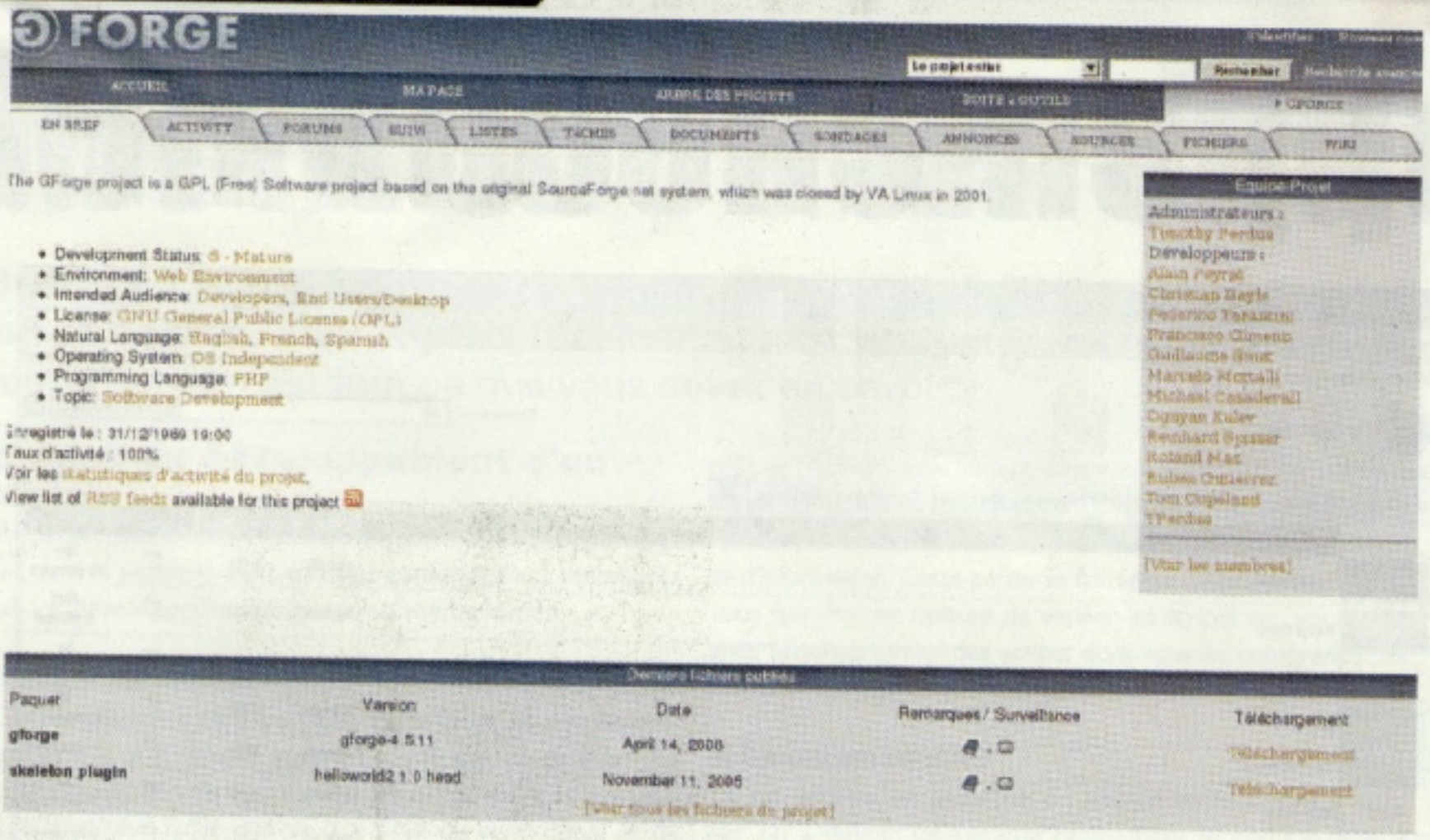
Une fois les impressions recueillies, il ne reste plus qu'à débugger les éventuelles erreurs, et à optimiser le code (c'est-à-dire essayer, sans en changer le fonctionnement, de le rendre plus petit ou plus rapide à exécuter). Bien sûr, de nouveaux tests viennent confirmer la réussite de l'opération.

j) Première version

La première version est prête : vous pouvez donc la mettre à disposition des utilisateurs, accompagnée de fichiers d'aide et d'information. N'oubliez pas de leur laisser un moyen de vous joindre pour toute question ou remarque.

Conseils sur le développement en PHP

- Vérifier la configuration du serveur : il convient de vérifier que le serveur ne bloque pas les fonctions utilisées dans le script (en exécutant la fonction phpinfo()), qu'il utilise une version suffisamment récente de PHP (avec phpversion()) et les bonnes extensions (aussi dans phpinfo()). Vous pouvez faire un script vérifiant tout cela, ou indiquer la configuration requise dans les fichiers d'aide.
- Gestion de erreurs personnalisée : il vaut mieux ne pas laisser s'afficher les message d'erreur (grâce à l'opérateur @), mais utiliser des fonctions de test pour vérifier la bonne marche du script (ainsi que or die($message_erreur);) (cf. Article " Blindez votre site ! ")
- Utiliser des noms de variables/fonctions/pages explicites : n'hésitez pas à utiliser des synonymes ou à mélanger l'anglais et le français. Si vous voulez être vraiment rigoureux, vous pouvez faire comme ceci : $ + type de variable (1 lettre) + nom explicite. De même pour les noms de fonction,



où le type est celui de la valeur retournée par la fonction. Ex : bIsValid($sEmail) pour une fonction vérifiant la validité d'une adresse e-mail, avec s pour string (chaîne de caractère) et b pour booléen (TRUE/FALSE).

- Optimisation taille/rapidité : à faire en fonction de vos exigence. Par exemple, vous pouvez remplacer un if then else par l'opérateur ternaire ?:, ou choisir de lire un fichier avec file plutôt qu'avec fgets.
- Ne pas hésiter à consulter la documentation en ligne de PHP, et à faire des tests en complément, car celle-ci contient quelques erreurs.
- Utiliser un éditeur spécialisé avec coloration syntaxique (PHPed, Jext, etc.) : cela permet d'éviter pas mal d'erreurs.
- Tester, lorsque c'est possible, en local plutôt que sur un serveur (question de sécurité).
- Les tests(en particulier les bêta-tests) doivent, si possible, être effectués sur différentes plate-formes (au moins Windows et un Unix-like).
- Joindre des fichiers d'aide et d'information indiquant le ou les auteurs (avec leurs emails et sites web), une description rapide du script, un historique des versions et une configuration particulière requise si besoin est.

le saviez-vous ?

Dans l'optique de la gestion et le suivi de vos projets personnels ou de groupes, PHP a vu fleurir des gestionnaires de projets ("projects managers") réellement dignes de ce nom. On peut ainsi citer DotProject (http://http://www.dotproject.net) ou encore Gforge (http://www.gforge.org). N'hésitez pas à les tester !

Erreurs courantes

Voici une liste non-exhaustive des erreurs les plus courantes en PHP, autrement dit des choses à vérifier en premier lieu lorsqu'une erreur s'affiche à l'exécution.

Erreurs d'écriture (parse error) :

- virgule, point-virgule ou dollar manquant,
- erreur de parenthèse / accolades / crochets en trop ou non-fermé(e)s,
- guillemets doubles et simples en trop ou manquants, caractères non-échappés,
- fautes de frappe,
- confusion (ex : print et echo, \n et \r\n),

Autres :

- erreurs mathématiques (racine carrée d'un nombre négatif, division par 0, etc.),
- portée des variables,
- fichiers inexistants ou inaccessibles,
- header tardif,
- boucle infinie.

Le manque de rigueur de PHP est souvent la cause de confusion, aussi faut-il être particulièrement vigilant à la syntaxe et l'orthographe de chaque fonction.
Exemple courant : is_set() ou isdir() (au lieu de isset() et is_dir()).

Conclusion

Cette méthode n'est évidement pas obligatoire, et est surtout destinée à de gros projets, les petits nécessitant rarement suffisamment de temps pour différencier les étapes. Cela dit, elle se rapproche de celle utilisée par les ingénieurs (elle a d'ailleurs été lue, corrigée et approuvée par un ingénieur que je remercie au passage).
Sur ce, je vous souhaite à tous un bon développement, et n'oubliez pas : il y a deux manières d'écrire des programmes sans bugs, seule la troisième fonctionne.

Coder proprement en PHP

Organiser les fichiers

Il est important de ranger soigneusement vos fichiers dans des sous-répertoires appropriés, si vous ne voulez pas rapidement être perdu dans votre propre désordre. Vous pouvez par exemple vous inspirer de cette structure :

- **/ : la racine**, où vous placerez les contrôleurs principaux (les pages qui correspondent aux requêtes des utilisateurs, comme index.php), d'éventuelles pages statiques (contact.html ?) et c'est tout !
- **/images :** pour les images statiques
- **/styles :** pour les feuilles de style (.css)
- **/scripts :** pour les scripts JS (il vaut mieux faire appel à des scripts externes à l'aide de la balise <script src="/scripts/exemple.js"></script> que d'insérer le code directement dans les templates ou, pire, les fichiers .php)
- **/lib** ou **/include :** pour ranger les scripts PHP qui ne sont pas appelés directement par l'utilisateur, mais inclus dans d'autres fichiers .php. C'est aussi un bon endroit pour les fichiers de configuration. Vous pouvez même configurer Apache pour que ces fichiers ne soit pas accessible via HTTP.
- **/admin** , **/forum** etc. : créez un sous répertoire pour chaque partie individuelle du site

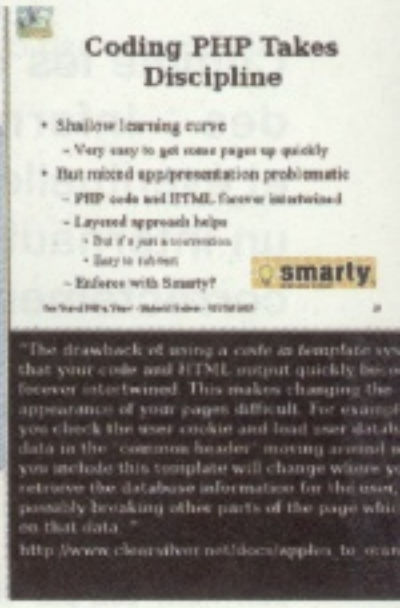
Séparation code/contenu

Même si PHP vous permet de mêler du contenu et du code, ce n'est pas la meilleure approche. Pour un projet un tant soi peu sérieux, il faut utiliser un moteur de template (voir à titre d'exemple l'article sur Modelixe). De cette façon, vos fichiers .php contiendront presque exclusivement du code. On peut comparer ces deux versions équivalentes :

```
<?php
include('ModeliXe.php');
$page = new ModeliXe('exemple.mxt');
$page -> SetModeliXe();
$page -> MxText(
   'Page.Titre','Avec ModeliXe');
$page -> MxText(
   'Page.Texte',
   'Bienvenue '.$_SERVER['REMOTE_ADDR']);
$page -> MxWrite();
?>
```

ou :

```
<html>
```



Une présentation intitulée « One Year of PHP at Yahoo! » par Michael J. Radwin http://public.yahoo.com/~radwin/talks/

```
<body>
<h1>Exemple sans ModeliXe</h1>
<p>Bienvenue <?php echo
$_SERVER['REMOTE_ADDR']; ?></p>
</body>
</html>
```

On voit que le premier exemple, utilisant un template, est beaucoup plus clair. Il sera plus facile à lire pour une autre personne (ou vous, dans quelques mois), parce que il est organisé sémantiquement et relativement explicite - même pas besoin de commentaires !

Il est vrai que le deuxième exemple est peut-être plus facile à lire pour un designer habitué à lire l'html. Cependant il risquerait de barbouiller le code par mégarde s'il ne fait pas attention. En revanche, si on lui passe un template (exemple.mxt ici), qui ne contient pas un gramme de code, il pourra travailler sereinement.

Prototypage

Enfin, comme beaucoup d'entre vous sont probablement plus pressés de coder que de suivre d'obscures méthodologies de génie logiciel, n'hésitez pas à commencer par réaliser un prototype, codé à la va-vite, pour vous faire une idée des difficultés que vous aurez à résoudre. Une fois que vous êtes sûr de vous, reprenez le projet à zéro, sérieusement, tout en récupérant les morceaux de code qui vous paraissent être bon, grâce à du copier/coller.

Sessions PHP un espace me

Comme les cookies, les sessions (depuis PHP 4.0) permettent de garder des informations en mémoire entre plusieurs pages PHP, et éventuellement entre plusieurs accès. Il est ainsi possible de reconnaître un internaute d'une page à une autre, ou de simplement vérifier s'il s'est correctement identifié, autorisant ainsi la création d'un espace membre.

Sessions PHP : créer un espace membres

Chaque session créée génère un identifiant unique, qui passe de page en page soit via l'URL, soit via un cookie.
Son nom est, par défaut, PHPSESSID (modifiable dans le php.ini ou directement dans le code). Les sessions peuvent être utilisées de deux façons différentes :

- soit de façon booléenne (la session a-t-elle été créée ou non ?)
- soit pour enregistrer des données (quelles options a choisi le membre ? Quel est son identifiant ? etc.)

Nous allons étudier ici les deux possibilités.

Pour nos exemples, nous allons considérer une table SQL "membres" contenant les champs suivants :

- id, l'identifiant numérique unique du membre (1, 2... 443, etc.).
- login, le nom d'utilisateur du membre, unique aussi pour chaque membre, composé de caractères et de chiffres.
- pass, le mot de passe de l'utilisateur, composé de caractères et de chiffres.

L'espace membre se composera de deux pages :

- page1.php, contenant le formulaire de Log In et le code d'authentification.
- page2.php, qui est l'espace membre à proprement parler ; réservé aux membres.

Pour ne pas devoir répéter la connexion à la base de données sur les deux pages, on crée un fichier header.php dans lequel on mettra le code suivant, qui sera inclu dans les pages qui en ont besoin :

```
<?
// Informations nécessaires
// à la connexion :
$dbhost     = "localhost";
$dblogin    = "root";
$dbpassword = "";
$dbname     = "sessions";

// Connexion au serveur :
$db = mysql_connect($dbhost, $dblogin,
                    $dbpassword);
// Connexion à la DB :
mysql_select_db($dbname,$db);
?>
```

Commençons donc par l'utilisation booléenne des sessions. La première chose à faire est d'initialiser une session via la fonction session_start();.
Note : PHP peut le faire automatiquement si session.auto_start est activé dans le fichier de configuration php.ini.

Il nous faut ensuite, dans cette page page1.php, un formulaire POST dans lequel l'utilisateur devra entrer son nom d'utilisateur et son mot de passe. Comme la vérification de ces derniers se fera dans la même page, l' "action" sera page1.php (on pourrait également la laisser vide, ce qui prendrait page1.php par défaut comme "action") :

```
<form method="POST"
 action="page1.php">
Mot de passe :
  <input type="password"
   name="mpass"><br>
```

: créer mbres

```
Nom d'utilisateur :
<input type="text" name="mlogin"><br>
<input type="submit"
 name="send" value="Log In">
</form>
```

Mais ce formulaire ne devra pas s'afficher à chaque fois : s'il a été envoyé et que le mot de passe et le nom d'utilisateur sont corrects, le membre devra être authentifié et pourra rentrer dans son espace membre.
Le code de la page page1.php est donc, à notre niveau :

```
<?
// Connexion au serveur sql, ... :
include("header.php");
// On initialise une session :
session_start();

if (!isset($_POST["send"]))
{
 // Si le formulaire n'est pas
 // envoyé, on l'affiche :
?>
<form method="POST" action="page1.php">
Nom d'utilisateur :
<input type="text" name="mlogin"><br>
Mot de passe :
<input type="password"
 name="mpass"><br>
<input type="submit"
 name="send" value="Log In">
</form>
<?
}else{ // Sinon :
[...]
}
?>
```

Si le formulaire est déjà envoyé, il y a deux possibilités : les nom et mot de passe utilisateur sont corrects, et on redirige vers page2.php ou bien ils ne le sont pas, alors on affiche un message d'erreur et on réaffiche le formulaire.

Cette partie, représentée dans le code ci-dessus par [...], devrait donc ressembler à ce qui suit :

```
$membre =
   addslashes($_POST["mlogin"]);
$passw = $_POST["mpass"];

$sql = "SELECT pass ".
       "FROM membres WHERE".
       "login='$membre'";
$req = mysql_query($sql)
   or die('Erreur SQL');
$res = mysql_fetch_array($req);

if ($passw != NULL
    && $res['pass'] == $passw)
{
    session_register("membre");
    echo "<a href=\"page2.php\">".
         "Entrez dans l'espace".
         "membres.</a>";
} else {
    echo "<a href=\"page1.php\">".
         "Mauvais login ou mot".
         " de passe.</a>";
    }
```

Petite analyse :
On stocke d'abord les résultats du formulaire dans deux variables : le login dans $membre et le mot de passe dans $passw.

Ensuite on extrait de la table "membres" le mot de passe qui correspond au login entré par l'utilisateur, qui sera stocké dans $res['pass']. On compare ce dernier au mot de passe entré par l'utilisateur ; s'il est incorrect on affiche le message "Mauvais login ou mot de passe" avec un lien vers le formulaire, et s'il est correct on affiche le message "Entrez dans l'espace membre" avec un lien vers page2.php, et on crée bien sûr une session.
Pour créer cette session, on utilise la fonction session_register(), qui a comme argument le nom de la session

que l'on veut créer. Ici notre session portera donc le nom "membre".
Note : À propos des redirections, quelques problèmes de compatibilité ont été remarqués entre les sessions et la redirection via header(), il se peut donc que l'utilisation du javascript (par exemple) soit nécessaire.

Reste maintenant à créer le code de page2.php, qui vérifiera si oui ou non le membre est bien un membre (donc si une session a bien été créée). Rien de bien compliqué grâce à la fonction session_is_registered(), qui reçoit en argument le nom d'une session, et qui renvoie TRUE si elle est bien enregistrée, et FALSE si elle ne l'est pas.

Il sera peut-être aussi nécessaire, pour faire les choses proprement, de créer une page logout.php afin de déconnecter la personne en fin d'utilisation du script (entre autres pour la sécurité du membre).

Pour cela, il faut utiliser, dans notre cas, deux fonctions :

- session_unregister(), qui va " désenregistrer " la session créée avec session_register() et
- session_destroy(), qui va détruire la session lancée via session_start(). logout.php contiendra donc le code :

```
<?
session_start();
session_unregister("membre");
session_destroy();
?>
```

D'autres fonctions peuvent être utiles pour les sessions, comme :

- session_name(), qui lit et/ou modifie le nom d'une session,
- session_unset(), qui détruit toutes les variables d'une session,
- et d'autres encore, disponibles en français sur le manuel officiel de http://www.php.net.

Au lieu d'utiliser session_register(), l'équipe PHP conseille, de façon à être indépendant de l'option register_globals, d'utiliser le tableau superglobal $_SESSION (anciennnement $HTTP_SESSION_VARS). En effet, session_register() risque de ne pas fonctionner correctement dans les environnements où register_globals est désactivé.

En dehors de ce dysfonctionnement, $_SESSION est fort pratique pour identifer plus rapidement un membre ou enregistrer des préférences.

Revoyons donc notre espace membres, composé de page1.php, page2.php et logout.php, mais cette fois-ci en utilisant $_SESSION.

Voici ce que va devenir page1.php (header.php ne change pas)

```
<?
include("header.php");
session_start();

if (!isset($_POST["send"])){

?>

<form method="POST" action="page1.php">

Nom d'utilisateur :
<input type="text" name="mlogin"><br>

Mot de passe :
<input type="password" name="mpass"><br>

Boisson préférée :
<input type="text" name="mboisson"><br>

<input type="submit" name="send"
value="Log In">
</form>

<?
} else {

    $membre = addslashes($_POST["mlogin"]);
    $passw = $_POST["mpass"];

    $sql = "SELECT pass FROM membres
            WHERE login='$membre'";
    $req = mysql_query($sql)
              or die('Erreur SQL');
    $res = mysql_fetch_array($req);

    if ($passw != NULL
        && $res['pass'] == $passw){
      $_SESSION["membre"] = $membre;
      $_SESSION["boisson"] =
        htmlspecialchars($_POST["mboisson"],
        ENT_QUOTES);
      echo "<a href=\"page2.php\">'.
       Entrez dans l'espace membre.</a>";

    } else {
      echo "<a href=\"page1.php\">'
       Mauvais login ou mot de passe.</a>";
    }
}
?>
```

On voit deux différences majeures par rapport au premier espace membres. D'abord j'ai rajouté un champ "Boisson préférée" au formulaire :), et aussi (et surtout) la ligne :

```
session_register("membre");
```

a été remplacée par les lignes :

```
$_SESSION["membre"] = $membre;
$_SESSION["boisson"] =
 htmlspecialchars($_POST["mboisson"],
                  ENT_QUOTES);
```

Contrairement au code précédent, on peut pour un membre stocker plusieurs valeurs dans un tableau qui seront réutilisables sur chaque page de l'espace membres.

page2.php pourrait maintenant devenir :

```
<?
include("header.php");

session_start();

if (isset($_SESSION["membre"])){
  $sql = "SELECT pass FROM membres
          WHERE login='".
             $_SESSION["membre"]."'";
  $req = mysql_query($sql)
          or die('Erreur SQL');
  $res = mysql_fetch_array($req);

  echo "Bienvenue ".$_SESSION["membre"].
       ", vous prendriez bien un verre de
       ".$_SESSION["boisson"]." ? <br>";

  echo "Votre mot de passe est : <i>".
       $res['pass']."</i>.";

} else {
  echo "<a href=\"page1.php\">"."
       Désolé, vous devez être logué pour
       accéder à l'espace membre !</a>";
}
?>
```

On remarque clairement les avantages de cette technique :

- On reconnaît le membre qui s'est logué, ce qui n'était pas possible dans l'espace membre précédent, on peut alors, par exemple, extraire d'autres informations de la base de données à son sujet.
- On peut directement utiliser les valeurs enregistrées dans $_SESSION, comme pour afficher sa boisson préférée.

Enfin, le fichier logout.php ne doit plus utiliser session_unregister(), mais bien unset(), comme une variable classique. Ce qui se transformera ici en :

```
<?
session_start();
unset($_SESSION["membre"]);
unset($_SESSION["boisson"]);
session_destroy();
?>
```

Note : Avant PHP 4.3 et avec register_globals activé, c'est toujours session_unregister() qui doit être utilisé.
Comme je l'ai indiqué en début de texte, l'identifiant de session est envoyé via URL ou via cookies.

Cet identifiant, contenu dans une constante "SID", est de la forme "nom_session=valeur_session", "nom_session" étant la valeur donnée à session.name dans le php.ini (PHPSESSID par défaut). Il est envoyé en priorité par cookie, mais tout le monde n'utilise pas les cookies.
Dans ce cas, il faut nous mêmes ajouter à la fin de chaque lien le SID en question. Dans notre espace membres, il n'y a qu'une ligne à changer dans page1.php, c'est :

```
echo "<a href=\"page2.php\">".
     "Entrez dans l'espace".
     " membre.</a>";
```

en :

```
echo "<a href=\"page2.php?".
     htmlspecialchars(
       SID,ENT_QUOTES).
     "\">Entrez dans l'espace ".
     "membre.</a>";
```

Si vous voulez malgré cela forcer l'utilisation des cookies, il faut activer l'option "session.use_only_cookies".

Vous voilà maintenant capables d'utiliser les sessions afin de créer un espace membres, en fonction de votre configuration et de votre version PHP. Tous les codes utilisés ont été simplifiés au maximum, à vous d'en faire quelque chose d'attrayant pour l'utilisateur !
De nombreux détails sont toujours disponibles (en français) sur le site officiel, comme à l'adresse : http://be2.php.net/session. Vous y trouverez toutes les fonctions relatives aux sessions, les options de configurations en rapport expliquées, l'influence de certaines autres options... Tout ce qu'il faut pour les maîtriser sur le bout des doigts.

Bon coding !

Germain Randaxhe
aka frog-m@n

Opérations sur le

Comme il n'y a pas que les bases de données dans la vie, PHP possède une flopée de fonctions (plus de 70) permettant la création et la manipulation de flat files, c'est-à-dire de fichiers " simples ". Nous allons nous attarder sur les opérations de base à travers un exercice pratique : un script de gestion de fichiers.

Ouverture et fermeture

Bien que toutes les opérations sur les fichiers ne nécessitent pas l'ouverture préalable dudit fichier, les plus courantes et standards que sont fputs() et fgets() (un peu de patience) se placent entre fopen() et fclose(). Ainsi, le schéma type d'une utilisation de fichier est :

```
<?
$handle = fopen($nomDuFichier, $mode);
// On manipule le fichier... Et on le ferme.
fclose($handle);
?>
```

Si l'ouverture du fichier est réussie, fopen() renvoie un pointeur qui devra être utilisé dans les opérations de lecture/écriture sur ce fichier. Celui-ci est donc stocké dans la variable $handle. Le terme anglais handle désigne la variable utilisée pour atteindre le fichier ouvert.
Ce dernier est évidemment le fichier dénommé par la variable $nomDuFichier (qui est donc une chaîne de caractères). La variable $mode, également chaîne de caractères, indique le mode dans lequel le fichier doit être ouvert, c'est-à-dire l'utilisation que l'on va en faire : ce mode peut donc être lecture (dans ce cas $mode vaut " r " comme read), écriture (" w " comme write, le contenu du fichier est alors effacé) ou ajout (" a " comme append). On peut l'ouvrir en lecture et écriture ; soit en effaçant le contenu du fichier (" w+ "), soit sans l'effacer (" r+ " ou " a+ ").

Lecture d'un fichier

Il existe plusieurs façons de lire le contenu d'un fichier, donc, pour ce faire, plusieurs fonctions.
Fgets()est sans doute la plus courante.
Sa forme générale est :

```
$chaine = fgets($handle, $longueur);
```

où $handle est le handle obtenu à l'ouverture du fichier par fopen(), et $longueur le nombre de caractères + 1 (!) à lire. L'exécution de cette fonction se termine lorsque le nombre de caractères spécifié est lu, ou s'il rencontre un caractère de fin de ligne (newline) ou une fin de fichier (EOF, end of file).
Il faut alors rappeler la fonction pour continuer la lecture (sur une nouvelle ligne si c'est le newline qui a arrêté la lecture précédente), sauf évidemment si l'on se trouve à la fin du fichier. Cette fonction est donc souvent utilisée dans une boucle, par exemple dans un while (!feof($handle)) comme nous le verrons plus bas.
fgetc($handle) est l'équivalent de fgets($handle, 2), c'est-à-dire qu'il ne lit qu'un seul caractère.
fgetss() : cette fonction se comporte de la même façon que fgets() mais supprime en plus les balises HTML se trouvant dans les enregistrements lus. Un troisième argument, facultatif, permet de préciser les balises que l'on souhaite conserver. Exemple :

```
$chaine = fgetss($handle, 1024, "<b><i><u>");
```

Ces trois fonctions nécessitent une ouverture en lecture (" r ", " r+ ", " w+ " ou " a+ ") du fichier par fopen() (ou équivalent).

Deux autres fonctions utiles de lecture, file() et readfile(), ont la particularité de ne pas nécessiter l'ouverture préalable du fichier par fopen(). Elles ne prennent donc comme argument que le nom du fichier, et non un handle. file() renvoie un tableau dont chaque ligne contient une ligne du fichier, et readfile() lit le contenu du fichier avant de l'afficher dans le navigateur. readfile() retourne TRUEen cas de succès, FALSE sinon. Exemple :

```
<?
$tableau = file("test.txt");
echo "Le fichier test.txt contient
".count($tableau)." lignes.";
echo "Voici son contenu:<br>";
readfile("test.txt");
?>
```

s fichiers en PHP

Écriture dans un fichier

Il n'existe que deux fonctions, très semblables, pour écrire dans un fichier préalablement ouvert en écriture : fputs(), la plus courante, et fwrite(), que nous verrons dans la lecture/écriture en binaire. Leur forme générale est :

```
$nb = fputs($handle, $chaine [, $longueur] );
```

Le troisième argument, facultatif, permet de limiter la longueur de la chaîne écrite ($chaine) dans le fichier dont le handle est passé en premier argument ($handle). La fonction retourne le nombre de caractères écrits, FALSE en cas d'échec.

Création de fichier

Pour créer un fichier, il suffit de l'ouvrir en écriture (ou ajout) avec " w ", " w+ ", " a " ou " a+ ". En effet, si l'on tente d'ouvrir en écriture un fichier qui n'existe pas, PHP tente de le créer. Pour ouvrir un fichier en écriture sans le créer s'il n'existe pas, on peut utiliser " r+ " (lecture-écriture).

Premier listing : edit.php

À ce stade, nous pouvons commencer notre gestionnaire de fichier en PHP par une page regroupant les fonctions de lecture et écriture.

Exemple d'un fichier edit.php :

```
<?
if(!isset($fichier))
  header("Location: main.php");
?>

<html>
<head>
<title>Editer un fichier</title>
</head>
<body>

<?
if (isset($contenu)) {
 $f = fopen($fichier, "w");

 if (fputs($f, $contenu))
   echo "Le fichier $fichier a été enre-
         gistré avec succès.";
  fclose($f);
}
?>
<form action="#" method="post">
<b>Editer le fichier
<? echo $fichier; ?></b><br>

<input type="hidden" name="fichier"
       value="<? echo $fichier; ?>">
<textarea name="contenu" rows="20"
          cols="70">

<? readfile($fichier); ?>

</textarea>
<br>
<input type="submit" value="Enregistrer">
</form>

<form action="main.php">
<input type="submit" value="Retour">
</form>
</body>
</html>
```

Cette page permet l'édition (donc la création) du fichier dont le nom est passé à la page dans la variable $fichier. On utilise readfile() pour mettre le contenu du fichier dans la textearea et un fputs() pour enregistrer le contenu dans le fichier.

On s'aperçoit que pour créer le fichier, il faut passer à la page le nom du fichier qui n'existe pas encore (dans $fichier) et un contenu vide mais existant ($contenu="").

main.php sera la page principale de notre gestionnaire de fichier.

Précisions sur la lecture/écriture

Lecture/écriture en binaire

Pour pouvoir lire et écrire en binaire dans un fichier, il faut l'ouvrir en ajoutant " b " au mode d'ouverture, et utiliser fread() et fwrite() au lieu de fgets() et fputs(). Ceci n'est évidemment valable que dans les systèmes d'exploitation qui distinguent les fichiers de type texte des fichiers binaires (c'est le cas de Windows). Sinon les fonctions s'utilisent indifféremment.

Fonction	Vérifiant que
file_exists()	le fichier existe
is_dir()	il s'agit bien d'un répertoire
is_executable()	le fichier est exécutable
is_file()	il s'agit bien d'un fichier
is_readable()	accessible en lecture
is_writable()	accessible en écriture
is_writeable()	alias de is_writable
is_uploaded_file()	le fichier est un fichier téléchargé

Le pointeur de lecture/écriture

C'est le pointeur qui indique la position actuelle dans le fichier, c'est-à-dire la position à laquelle on va écrire ou celle à partir de laquelle on va lire. Lorsque l'on ouvre un fichier en lecture (ou en lecture-écriture avec " r+ "), le pointeur est à la fin du fichier, et inversement à la fin du fichier lorsque celui-ci est ouvert en écriture ou ajout (ou avec " w+ " ou " a+ ").

Lorsqu'on lit ou que l'on écrit dans le fichier, le pointeur se déplace du nombre de caractères que l'on écrit ou que l'on lit. Mais il existe des fonctions qui permettent de manipuler ce pointeur, la plus utilisée étant rewind(), qui prend comme paramètre le pointeur sur fichier (handle) et renvoie le pointeur au début du fichier.
Autre fonction utile : fseek(), qui permet de positionner le pointeur par rapport à sa position courante ou par rapport au début (ou à la fin) du fichier.
Sa forme générale est : fseek($handle, $decalage, $origine) où le pointeur sera décalé de $decalage caractères par rapport à $origine dans le fichier de pointeur $handle. $origine peut prendre les valeurs SEEK_SET (début du fichier), SEEK_CUR (position courante, valeur par défaut) ou SEEK_END (fin du fichier).

Enfin, la fonction ftell() qui, tel rewind, prend comme paramètre le pointeur sur fichier (handle), renvoie la position actuelle du pointeur. Donc, si vous avez bien suivi, dans l'exemple suivant $pos vaut 4 :

```
<?
 $handle = fopen("toto.txt", "r");
  rewind($handle);          //position: 0
  fseek($handle, 4, SEEK_CUR);
  $pos = ftell($handle);
  fclose($handle);
?>
```

Tests sur les fichiers

Avant de faire n'importe quoi sur les fichiers, il convient de s'assurer que l'on ne va pas travailler sur un fichier qui n'existe pas, écrire dans un répertoire, ou déplacer un fichier dans un autre. Aussi PHP dispose-t-il d'une batterie de fonctions destinées à faire des tests sur les fichiers et dossiers. En voici une liste quasi-exhaustive :
Chacune de ces fonctions prend un nom de fichier ou de dossier comme argument et renvoie TRUE si le test est vérifié, FALSE dans le cas contraire.
Une autre fonction de test, feof(), permet de savoir si l'on est arrivé au bout d'un fichier (dans ce cas feof($handle) renvoie TRUE) : il est utilisé lors de la lecture. Dans l'exemple suivant, on lit le fichier tant qu'on n'est pas arrivé à la fin, et l'on affiche le tout à la fin :

```
<?
$handle = fopen($nomDuFichier, "r");
while(!feof($handle))
  $chaine .= fgets($handle, 1024);
fclose($handle);
echo $chaine;
?>
```

Ce qui est l'équivalent de :
`<? readfile($fichier); ?>.`

Effacer un fichier

Listing 2 : delete.php

La fonction unlink() supprime le fichier dont le nom lui est passé en argument ($fichier). Ce qui nous permet de faire la seconde page de notre gestionnaire de fichier, delete.php :

```
 <?
if(!(isset($fichier) and
   is_file($fichier)))
  header("Location: main.php");
?>
<html>
<head>
<title>Supprimer un fichier</title>
</head>
<body>
<?

if(unlink($fichier)
   or !file_exists($fichier))
   echo "Fichier effacé avec succès.";

?>
<form action="main.php">
<input type="submit" value="Retour">
</form>
</body>
</html>
```

On vérifie bien sûr au début que le fichier existe avec is_file($fichier).

Renommer un fichier

Listing 3 : rename.php

C'est le rôle de la fonction rename($ancien_nom, $nouveau_nom), qui renomme le fichier $ancien_nom en $nouveau_nom. D'où le listing 3, rename.php :

```
<?
if(!(isset($fichier)
   and is_file($fichier)))
  header("Location: main.php");
 ?>
<html>
<head>
<title>Renommer un fichier</title>
</head>
<body>

<?
if (isset($nom)) {
  if (rename($fichier, $nom))
    echo "Le fichier $fichier à été
          renommé en $nom.";
}
?>
<form action="#" method="post">
Nouveau nom:<br>
<input type="hidden" name="fichier"
value="<? echo $fichier; ?>">
<input name="nom"
       value="<? echo $fichier; ?>"><br>
<input type="submit" value="Renommer">
</form><form action="main.php">
<input type="submit" value="Retour">
</form></body></html>
```

Remarque : Pour déplacer un fichier, on le renomme en précisant le dossier de destination dans le nouveau nom. Exemple :

```
rename("test.txt", "tests/test.txt")
```

Copier un fichier

Listing 4 : copy.php

La fonction copy($fichier, $nouveau_fichier) permet de copier le fichier $fichier en $nouveau_fichier. D'où le listing 4, copy.php :

```
<?
if(!(isset($fichier)
   and is_file($fichier)))
 header("Location: main.php");
?>
<html>
<head>
<title>Copier un fichier</title>
</head>
<body>
 <?
if (isset($newfile)) {
  if (copy($fichier, $newfile))
   echo "Le fichier $fichier à été copié
         en $newfile.";
}
?>

<form action="#" method="post">
Nom du nouveau fichier:<br>
<input type="hidden" name="fichier"
value="<? echo $fichier; ?>">
<input name="newfile" value="<? echo
$fichier; ?>.bak"><br>
<input type="submit" value="Copier">
</form><form action="main.php">
<input type="submit" value="Retour">
</form>
</body>
</html>
```

Lister les fichiers d'un répertoire

Listing 5 : main.php

Tripoter les fichiers implique souvent de savoir quels sont les fichiers présents dans le répertoire qui nous intéresse, c'est pourquoi PHP est doté de quelques fonctions permettant la manipulation de répertoires.

La manipulation de répertoires ressemble beaucoup à la manipulation de fichiers : il faut ouvrir le répertoire avec opendir($dossier), en lire le contenu (fichiers et dossiers) avec readdir($handle), et fermer le répertoire avec closedir($handle). readdir() est souvent utilisée dans une boucle, car à chaque appel, cette fonction renvoie un nom de fichier ou de dossier (tout comme fread() renvoie une ligne).

Dans notre gestionnaire, il faudra faire attention à ne pas confondre fichiers et répertoires : proposer d'éditer, renommer, copier et supprimer les fichiers, et d'explorer les répertoires ; on aura donc recours aux fonctions is_dir() et is_file() (on aurait bien sûr pu utiliser is_dir() et !is_dir() ou is_file() et !is_file()).

Les fichiers et répertoires sont donc listés dans des <select>, et l'on utilise un peu de javascript pour changer le paramètre " action " du formulaire principal (form1) afin de diriger l'utilisateur vers la page correspondant à son action. main.php :

```
<html>
<head>
```

```
<title>Gestion de fichiers</title>
<script language="Javascript">
function charge(page) {
  document.form1.action = page;
  document.form1.submit(); }
</script>
</head>

<body>
<form action="#" method="post">

Dossiers:
<select name="dir">

 <?
// Si $dir est vide ou erroné, on prend le
   dossier courant
if(!isset($dir)
   or !is_dir($dir)
   or $dir=="") $dir = ".";

$d = opendir($dir);
while ($dossier = readdir($d)) {
 if (is_dir($dossier))
   echo "<option value=\"$dir/$dossier\">
        $dossier</option>\n";
}
closedir($d);
?>

</select>
<input type="submit" value="Ouvrir">
</form>
<hr><form name="form1">
<select name="fichier">

<? $d=opendir($dir);
while ($file = readdir($d)) {
 if (is_file($file))
  echo "<option>$file</option>\n";
}
closedir($d);
?>

 </select>
<input type="button" value="Editer"
       onClick='charge("edit.php")'>
<input type="button" value="Renommer"
       onClick='charge("rename.php")'>
<input type="button" value="Copier"
       onClick='charge("copy.php")'>
<input type="button" value="Supprimer"
       onClick='charge("delete.php")'>
</form>
<hr>
<form action="edit.php">
<input type="text" name="fichier" max-
       lenght="50">
<input type="hidden" name="contenu"
       value="">
<input type="submit" value="Créer le
       fichier">
</form>
<hr>
<form action="upload.php">
<input type="submit"
       value="Uploader un fichier">
</form>
</body>
</html>
```

Upload de fichier

Listing 6 : upload.php

Un formulaire permet de télécharger des fichiers du client (visiteur) vers le serveur (dans un fichier temporaire), si celui-ci est configuré pour. Il faut pour cela un formulaire de méthode POST et possédant l'attribut ENCTYPE="multipart/form-data", ainsi qu'un input de type file. Un autre input, facultatif, permet de limiter la taille du fichier à uploader si la limite du serveur ne vous convient pas ainsi. <input type="hidden" name="MAX_FILE_SIZE" value="20000"> limitera la taille des fichiers à 20 000 caractères (20 000 octets, ou 19,5 Ko si vous préférez).

Il faut ensuite récupérer le fichier, toujours uploadé dans le même répertoire, avec un script : c'est là que PHP intervient. C'est la fonction move_uploaded_file() qui s'en charge : elle prend pour argument le nom temporaire du fichier uploadé et le nom sous lequel il doit être enregistré (je parle de nom complet : emplacement+nom).

Le premier s'obtient en fouillant dans les tableaux des fichiers uploadés ($HTTP_POST_FILES), en précisant la variable utilisée (le nom de l'input de type file) et l'attribut " tmp_name " (le nom temporaire que l'on cherche). Par exemple : $HTTP_POST_FILES["fichier"]["tmp_name"]

La fonction renvoie TRUE si l'exécution s'est bien déroulée.

On peut voir dans le dernier listing (upload.php) l'application de ce concept, où l'on vérifie que le fichier existe bien avec file_exists (j'avoue que c'est un excès de zèle).

Remarque : L'upload de fichier ne fonctionne pas toujours, notamment sous Windows, lorsque le nom complet (chemin d'accès + nom) du fichier à uploader contient des espaces. On peut y remédier en déplaçant ses fichiers à la racine du disque dur (C:\ dans la plupart des cas) avant de les uploader.

La fonction move_uploaded_file() agit de la même manière que copy(), à la différence près qu'elle déplace le fichier au lieu de le copier (mais de toutes façons, l'original étant dans

le dossier " spécial upload " du serveur, il sera supprimé après un certain temps).
upload.php :

```
<html>
<head>
<title>Upload</title>
</head>
<body>

<?
if (isset($nom) and isset($fichier)) {
 if(!$nom or $nom=="")
   $nom=tempnam("./");
 $chemin = "./".$nom;

 if (move_uploaded_file(
       $HTTP_POST_FILES["fichier"]
            ["tmp_name"],
       $chemin) {
     and file_exists($nom))
   echo "Le fichier $chemin a été uploadé
         avec succès.";
 } else {
   echo "Echec de l'upload.";
 }
}
?>

<form enctype="multipart/form-data"
      method="post" action="upload.php">
<input type="hidden"
       name="MAX_FILE_SIZE"
       value="100000000">
Upolader un fichier:<br>
<input name="fichier" type="file"><br>
Sous quel nom doit-il être enregistré ?
<br>
<input name="nom" type="text"><br>
<input type="submit" value="Upload">
</form>
<form action="main.php">
<input type="submit" value="Retour">
</form>
</body>
</html>
```

Autres fonctions utiles

Voici, en vrac, une liste de fonctions qui peuvent être utiles pour la manipulation de fichiers et de répertoires. Je n'ai hélas pas la place de tout détailler ici, aussi je vous conseille de jeter un coup d'œil à la documentation en ligne pour de plus amples détails.

Fichiers

- `filesize($fichier)` renvoie la taille (en octets ou nombre de caractères) du fichier dont le nom ($fichier) lui est passé en argument.
- `filetype($fichier)` renvoie son type.
- `fileatime($fichier)` renvoie la date du dernier accès au fichier, filemtime($fichier) sa date de modification.
- `stat($fichier)` renvoie des statistique sur le fichier, fstat($handle) aussi mais prend un handle en argument.
- `Chmod` permet de changer les droits d'accès au fichier (sous UNIX).

Un peu de sécurité...

Il n'est pas difficile de renforcer un tant soit peu la sécurité de votre serveur Web sous UNIX en jouant convenablement sur les droits d'accès des fichiers UNIX.
Par exemple votre serveur web devrait avoir un compte utilisateur qui lui est dedié (utilisateur www, groupe www). Les pages du site devraient, dans la mesure du possible, n'appartenir qu'au root et au groupe 'www' : les pages ne sont plus alors disponibles qu'en lecture seule.
Cela vous evitera le "deface" : les pages ne peuvent êtres réécrites par le serveur web, ce dernier ne pourra que les lire, pas les posséder ou les réécrire. De même, aucun utilisateur sur la machine (à part root et 'www' cela s'entend) ne pourra lire le contenu de vos pages.
Notez également qu'il n'est pas judicieux de laisser des répertoires privés ou sensibles dans votre rÈpertoire de diffusion (htdocs), tels les rÈpertoires de logs, de scripts CGI...
Si vous voulez éviter la compromission de votre système, pensez également à faire un chroot. Un chroot est une prison virtuelle pour toutes formes d'applications, mais essentiellement pour des serveurs Web.
Construire un chroot n'est pas difficile, c'est plus fastidieux. Vous pouvez trouver une documentation officielle sur http://www.debian.org/doc/manuals/reference/ch-tips.en.html#s-chroot

Répertoires

- `mkdir($nom_rep, $droits)` crée le répertoire de nom $nom_rep avec les droits d'accès $droits (nombre octal, ex: 0733).
- `rmdir($nom_rep)` détruit le répertoire dénommé par $nom_rep.
- avec `chdir($nom_rep)`, le répertoire $nom_rep devient le répertoire courant.
- `getcpwd()` (sans argument) renvoie le chemin absolu du répertoire courant (par exemple C:\Program Files\EasyPHP\www\).
- `rewinddir($handle_rep)` est l'équivalent de rewind pour les répertoires.
- `diskfreespace()` permet de connaître l'espace disponible dans le répertoire courant (sur le disque dur sous Windows).

Noms de fichiers et de répertoires

- `dirname($nom)` renvoie le nom du répertoire qui contient le fichier ou dossier de nom $nom.
- `basename($chemin)` prend en paramètre le chemin complet d'un fichier et en extrait le nom du fichier.
- `realpath($chemin)` résout tous les liens symboliques, et remplace toutes les références /./ , /../ et / de $chemin puis retourne le chemin canonique absolu ainsi trouvé.

Exemples :
Avec le fichier C:\toto.txt
(on se trouve dans le répertoire c:\test) :
`dirname("../toto.txt")` renvoie C:\
`basename("C:/toto.txt")` renvoie toto.txt
`realpath("../toto.txt")` renvoie :
Erreur! Référence de lien hypertexte non valide.

Remarque : Dans les chemins de dossiers sous Windows, on peut utiliser indifféremment slash : " / ", ou antislash " \ ". Mais le slash est fortement conseillé car multi-systèmes et ne nécessite pas d'échappement (contrairement à l'antislash qui doit être doublé) : " C:\\test\\test.txt " est l'équivalent de " C:/test/test.txt ".

Gestion des erreurs

Comme expliqué plus haut (on voit ceux qui suivent :p), il convient, lorsqu'on manipule des fichiers et/ou des répertoires, de s'assurer de leur existence et de s'informer sur les droits que l'on a dessus (sur des serveurs correctement configurés, ce dernier point est rarement nécessaire), afin que le script fonctionne.
De même, s'informer de la bonne exécution d'une fonction à l'aide d'un test permet d'éviter l'affichage inopiné d'erreurs, toujours à éviter. Il existe des astuces, qui doivent devenir des réflexes pour contrôler les éventuelles erreurs : l'opérateur @, placé devant une fonction, empêche l'affichage des erreurs liées à cette fonction. De même, on utilisera la fonction " die " pour stopper l'exécution d'un script en cas d'erreur et afficher un message personnalisé. Voici un exemple de script correct :

```
<?
if file_exists("toto.txt") {
 $handle = @fopen("toto.txt", "r")
    or die(
   "Erreur à l'ouverture du fichier :(");
 while(!feof($handle))
    $chaine .= @fgets($handle, 1024);
 fclose($handle);
}
else
 echo "le fichier toto.txt n'existe pas !";
?>
```

Remarques sur le script : Il est évident que ce script, destiné à servir d'exemple, est loin d'être terminé. Voici donc quelques idées de choses à améliorer, dont certaines sont aussi des conseils pour vos scripts propres :

- toujours travailler dans le répertoire courant (avec chdir pour changer), et non relatif au répertoire où se trouve le script (utiliser realpath),
- récolter et afficher les infos sur les fichiers,
- gestion des droits d'accès (en fonction du système d'exploitation),
- gestion des erreurs,
- sécuriser toutes les pages (par exemple avec un " htaccess "),
- soigner le design (affichage en arborescence),
- utiliser du javascript, des tableaux et des variables de session pour manipuler plusieurs fichiers,
- tout en une page (avec un " switch "),
- plus d'actions (piocher des idées dans la liste de fonction de la documentation PHP).

Conclusion

On ne peut faire un tutorial sur la manipulation de fichiers sans avertissement sur la sécurité. Evidemment, lorsqu'on manipule des fichiers et des répertoires, il faut être particulièrement vigilant quant à la sécurité des scripts car, mal protégés, ils peuvent permettre à des tiers malintentionnés d'effacer (ou de défacer) le contenu de votre serveur, voire d'en prendre le contrôle si la machine elle-même est mal configurée (scripts ayant accès aux fichiers sensibles). Prudence donc...

Il ne me reste qu'à remercier dvrasp et Clad pour l'opportunité de l'article et vous dire @+

eks

Logging et filt votre site !

Certaines failles et erreurs PHP peuvent se révéler difficiles à démasquer lorsque ce sont des visiteurs qui les trouvent. Un script adapté permettrait d'enregistrer les erreurs survenant lorsqu'une personne visite votre site, et il deviendrait possible de bannir les adresses IP des personnes les ayant provoqué.

Ce script nécessite d'être inclus (include('log.php');) au début des fichiers PHP dans lesquels vous désirez détecter les erreurs.
Ces erreurs sont classées, par répertoires de date (jour-mois-année), puis par un fichier par heure contenant les erreurs (type d'erreur, IP, localisation de l'erreur et variables d'environnement (POST, GET...)).

Méthodologie

Tout d'abord, il nous faut rediriger les erreurs PHP. Pour cela nous utilisons set_error_handler (nom de la fonction où seront renvoyées les erreurs), soit :

```
set_error_handler ("recupere_e");
```

Ceci nous renverra à la fonction recupere_e(), dont les arguments sont : type d'erreur, message d'erreur, fichier où l'erreur se trouve, ligne où se trouve l'erreur et tableau des variables l'entourant (nous ne l'utiliserons pas).
Les deux premiers arguments son obligatoires, les autres facultatifs. Nous définissons le type de l'erreur avec la fonction recupere_e() et nous renvoyons la chaîne contenant la description de l'erreur à la fonction traitement_e().
Celle-ci va créer le répertoire de classement et le fichier de l'heure – s'ils ne sont pas présents - et ajouter dans le fichier les informations sur l'erreur et le visiteur. Ensuite elle fera appel à la fonction ban_ip qui rajoutera un avertissement à l'IP, voire la bannira si un certain nombre d'avertissements est dépassé.

Une IP contenant un ou plusieurs avertissements sera un fichier nommé *.ip.tmp avec comme contenu le nombre d'avertissements. Une IP bannie sera un fichier *.ip :
* faisant référence à l'IP, dans ce fichier sera contenu le nombre d'avertissements que l'IP possède.
Il suffira de supprimer le fichier pour enlever le ban ou les avertissements. Enfin nous mettons une fonction permettant de savoir si l'IP est bannie (l'existence du fichier *.ip) et nous ferons appel à celle-ci dès le début du script pour pouvoir stopper l'exécution de celui-ci si nécessaire.

Détails

La fonction serialize() permet de sauvegarder un objet sans perdre ni sa structure ni son type. Il est possible, entre autres, de sauvegarder un tableau.

Exemple :

```
$a[0]=10;
$a['php']=1;
$val=serialize($a);
```

$val prend la valeur de la sérialisation du tableau $a. La valeur retournée est une chaîne de caractères et est donc très facile à sauvegarder dans une base de données ou autre. Pour récupérer l'objet à partir de la chaîne, il faut utiliser la fonction unserialize(), par exemple :

```
$b=unserialize($val);
echo $b[0]."\n".$b['php'];
```

$b aura la même structure, les mêmes valeurs que $a.

Au travail !

Le script complet ci-dessous réalise le travail présenté. Si vous le désirez, vous pouvez créer une partie administrative qui permettrait de lister, lire et supprimer les logs comme de lister, ajouter ou supprimer les bans sur les adresses IP.

Pour cela, les fonctions suivantes vous seront utiles :

rage : blindez

`opendir()` permettra de lister les répertoires,
`fopen()` d'ouvrir/créer un fichier,
`unlink()` de supprimer un fichier,
`rmdir()` de supprimer un répertoire (vide).

Vous trouverez le détail de ces fonctions sur **http://www.php.net/manual/fr/**.

```
<?
/* GESTION DES ERREURS */

// Si la constante repertoire_log n'est pas définie (ceci permet de choisir le
// répertoire de destination dans le fichier PHP où l'on inclut celui-ci, utile
// lorsqu'ils ne se trouvent pas tous au même niveau)...
if ( !defined('repertoire_log') )
{
        // ...alors il l'a défini avec comme valeur log/
        // (ce répertoire contiendra le classement des erreurs,
        // à vous de le créer)
        define("repertoire_log","log/");
}

// Constante permettant de choisir si le mode de ban IP est activé, TRUE pour
// le mode activé, FALSE pour le mode désactivé.
define("ban_active",TRUE);

// Constante permettant de choisir à partir de combien d'erreurs
// l'IP se fait bannir.
define("nb_avert",3);
//Constante permettant de définir le message lorsque l'IP est ban.
define("message_ban","Ip banni contactez l'admin");

//Si le mode du ban est activé...
if(ban_active)
{
        // ... alors si is_ban() retourne vrai, afficher le message du ban et
        // stopper l'exécution du script.
        if(is_ban()){echo message_ban;exit;}
}

// Redirection des erreur vers la fonction recupere_e
set_error_handler("recupere_e");

// Que l'on définit ici :
function recupere_e($errno, $errstr, $errfile, $errline)
{
```

```
//$errno contient le type de l'erreur(int)
//$errstr contient le message d'erreur
//$errfile contient le fichier où se trouve l'erreur
//$errline contient la ligne de l'erreur

/*$errno est un nombre, nous nous limiterons à détecter les valeurs
  suivantes :

        1   E_ERROR
        2   E_WARNING
        8   E_NOTICE
        256      E_USER_ERROR
        512      E_USER_WARNING
        1024     E_USER_NOTICE

*/

switch ($errno) {
        //si $errno est égal à E_USER_ERROR soit 256...
        case E_USER_ERROR :
                $type="Fatal:\n";
                break;
        //si $errno est égal à E_USER_WARNING soit 512...
        case E_USER_WARNING :
                $type="Erreur:\n";
                break;

        //si $errno est égal à E_USER_NOTICE soit 1024...
        case E_USER_NOTICE :
                $type="WARNING:\n";
                break;

        //si $errno est égal à E_ERROR soit 1...
        case E_ERROR :
                $type="Fatal:\n";
                break;

        //si $errno est égal à E_WARNING soit 2...
        case  E_WARNING :
                $type="Erreur:\n";
                break;

        //si $errno est égal à E_NOTICE soit 8...
        case  E_NOTICE :
                $type="WARNING:\n";
                break;


        //Dans tous les autres cas
        default:
                $type="Inconnu:\n";
                break;

}
```

```
        //On renvoie la capture à la fonction de traitement.
        traitement_e($type."[$errno] $errstr\n".
                        "Ligne: ".$errline.
                        " Fichier: ".$errfile."\n");
}

function traitement_e($erreur)
{
        //$erreur contient le message d'erreur.
        //$info = date:jour-mois-anne heure:minute:seconde + IP du client...
        $info=date("d/m/Y  H:i:s",time())." : ".$_SERVER['REMOTE_ADDR'].
                //+ Serialisation du tableau $_GET, soit toute les variables
                // passer par URL...
                "\n\t GET:".serialize($_GET).

                //+ Sérialisation du tableau $_POST, soit toutes les
                // variables passées par méthode POST...
                "\n\t POST:".serialize($_POST).

                //+ si la variable $_COOKIE est définie, alors renvoie
                //Sérialisation du tableau $_COOKIE, soit tout le contenu
                // du cookie, sinon renvoie " Undefined ".
                "\n\t COOKIE:".
                  (isset($_COOKIE) ?  serialize($_COOKIE) : "Undefined" ).

                //+ Si la variable $_SESSION est définie, alors renvoie
                //Sérialisation du tableau $_SESSION, soit toutes
                // les variables contenues dans la session, sinon renvoie
                // " Undefined ".
                "\n\t SESSION:".
                  (isset($_SESSION) ? serialize($_SESSION) : "Undefined" ).

                //+ Serialisation du tableau $_SERVER, soit toutes
                // les variables Serveur...
                "\n\t SERVER:".serialize($_SERVER)."\n\n";
//$rep = répertoire où sont contenus les logs concaténés au
// jour-mois-année actuels.
$rep=repertoire_log.date("d-m-Y",time());
//Si le répertoire $rep n'existe pas...
if(!is_dir($rep))
{
        //alors crée le répertoire (crée le répertoire
        //jour-mois-annee actuel). S'il échoue, renvoie un message
        //et stoppe la fonction.
        if(!@mkdir($rep))
        {
                echo "Verifier l'existence du repertoire "
                        .repertoire_log;
                return FALSE;
        }
}
```

```
        // $fp= ouvre le fichier se trouvant dans $rep du nom de l'heure
        // actuelle concatenée à "h" en mode ajout. Si le fichier n'existe pas,
        // il se crée.
        $fp = fopen($rep."/".date("H",time())."h","a");

        // Rajoute dans le fichier, à la fin, $erreur et $info, soit les
        // informations de l'erreur et les informations du visiteur.
        fwrite($fp,$erreur.$info);
        //Ferme le fichier.
        fclose($fp);

        //Si le mode ban est activé...
        if(ban_active)
        {
                // alors ajouter un avertissement, retourne TRUE si l'IP,
                // suite à l'avertissement, est ban ; FALSE autrement.
                if(ban_ip())
                {
                        //Affiche le message de ban.
                        echo message_ban;
                        //Stoppe l'exécution du script.
                        exit;
                }
        }
}

function ban_ip()
{
        //$fichier = répertoire où sont contenus les logs plus l'IP
        $fichier=repertoire_log.$_SERVER['REMOTE_ADDR'].".ip";

//Si $fichier concatené à " .tmp " est un fichier et qu il existe...
        if(is_file($fichier.".tmp"))
        {
                // ...ouvrir ce fichier en lecture.
                $fp=fopen($fichier.".tmp",'r');

                //$avert = a la valeur contenue dans le fichier
                //(un nombre correspondant au nombre d'avertissements).
                $avert=fread($fp,filesize($fichier.".tmp"));

                //Ferme le fichier.
                fclose($fp);

                //Ajoute un avertissement.
                $avert++;

                //Si le nombre d'avertissements est supérieur au nombre
                //d'avertissements tolérés...
                if($avert>nb_avert)
                {
                        //alors renommer le fichier $fichier.tmp en $fichier
                        //(transforme le fichier d'avertissement en fichier
                        // de ban).
```

```
                rename($fichier.".tmp",$fichier);
                //La fonction se termine et renvoie TRUE
                // (soit que l'IP est ban).
                return TRUE;
        }
//Sinon (si $fichier concatené à " .tmp " n'existe pas...)
} else {
        //Si on bannit sans avertissement...
        if(nb_avert==0)
        {
                //ouvre le $fichier (le crée).
                $fp = fopen($fichier,"w");
                //Écrit 0 dedans
                //(juste pour la référence de dire 0 avertissement).
                fwrite($fp,"0");
                //Ferme le fichier.
                fclose($fp);
                //La fonction se termine et renvoie TRUE
                //(l'IP est bannie).
                return TRUE;
        }
        //$avert=1 correspond à son premier avertissement.
        $avert=1;
}
//Ouvre le fichier $fichier concatené à " .tmp " en mode écriture
// (crée le fichier s'il n'existe pas, ou écrase le contenu s'il existe).
$fp = fopen($fichier.".tmp","w");
//Écrit dans le fichier le nombre d'avertissements.
fwrite($fp,$avert);
//Ferme le fichier.
fclose($fp);
//La fonction se termine et renvoie FALSE
// (l'IP n'est pas bannie par cet avertissement).
return FALSE;
}

function is_ban()
{
        //Si le fichier contenu dans le répertoire répertoire_log du nom de
        // l'ip concatenée à " .ip " existe...
        if(is_file(repertoire_log.$_SERVER['REMOTE_ADDR'].".ip"))
        {
                //alors la fonction renvoie TRUE (l'IP est bannie).
                return TRUE;
        }else{
                //Sinon la fonction se termine et renvoie FALSE (l'IP n'est pas bannie).
                return FALSE;
        }
}
?>
```

La prévention reste LE facteur crucial sur lequel pivote toute politique de sécurité. Il est facile de prévenir et d'analyser les attaques potentiellement porteuses à l'aide du script que nous venons d'étudier. Gageons que vous en ferez un excellent usage !

Nyx

Créez un fichi

Le XML (eXtensible Markup Language) est, au même titre que l'HTML (qui est d'ailleurs son prédécesseur), un langage de balisage, c'est-à-dire un langage qui encadre l'information dans des balises. Mais contrairement au HTML, qui est un langage d'affichage, le XML est un langage de structuration.

La différence ne s'arrête pas là. Alors que dans le HTML, les balises sont prédéfinies et donc figées, le XML est extensible, et permet de créer ses propres balises en fonctions des données traitées.
Ainsi, bien que le XML et le HTML fonctionnent tout deux avec des balises, sont indépendants de la plate forme et sont en mode texte, ce sont deux langages bien distincts, autant dans la forme que dans le fond.
Le but d'un fichier XML est d'être réutilisé par un autre langage. Il faut donc structurer les informations nécessaires à un futur traitement de façon clair.
Le corps du fichier XML simplifié d'une petite bibliothèque pourrait ressembler à :

```
<biblio>
<rubrique name="policier">
  <livre>
   <auteur>Edgar Poe</auteur>
   <titre>Double assassinat dans la rue
        Morgue </titre>
  </livre>
  <livre>
   <auteur>Egar Poe</auteur>
   <titre>le Mystère de Marie
        Roget</titre>
  </livre>
</rubrique>
<rubrique name="voyage">
  <livre>
   <auteur>Routard</auteur>
   <titre>Tunisie</titre>
  </livre>
</rubrique>
</biblio>`
```

Cette bibliothèque est vraiment fort petite :)
Malgré la liberté de création que permet le XML, certaines règles syntaxiques (que l'on retrouvera dans les langages dérivés comme le XHTML, le WML ou le MathML), comme on peut le constater dans l'exemple ci-dessus doivent être respectées :

- Un nom de balise ne peut pas contenir de caractères spéciaux (-,;,,<,>,...) ; rien que des chiffres et des lettres (les accents sont permis mais déconseillés), et éventuellement des underscores. Pas d'espaces non plus.

```
<nom-prenom></nom-prenom>
ou <nom prenom></nom prenom>
```

sera donc à remplacer par exemple par :

```
<nom_prenom></nom_prenom>
```

- Un nom de balise ne peut ni commencer par un chiffre, ni par les lettres "xml".

```
<1prix></1prix>
et
<xmlprix></xmlprix>
seront donc interdits.
```

- Un nom de balise est sensible à la casse !

<Rubrique></Rubrique> et <rubrique></rubrique> sont des balises différentes.

- Une balise qui est ouverte doit être refermée.

Si un élément esseulé est nécessaire, il aura la syntaxe :
<elementtoutseul\>

- Les balises doivent être correctement imbriquées.

<parent><enfant></parent></enfant>est incorrect.

<parent><enfant></enfant></parent>est correct.

- Chaque fichier XML doit commencer par une balise "racine", qui encadrera la totalité du fichier.

<biblio> dans le cas de l'exemple au dessus.

- Les valeurs des attributs doivent être mises entre guillemets.

```
<rubrique langue="fr"></rubrique>est correct.
<rubrique langue=fr></rubrique>est incorrect.
```

ier XML en PHP

Passons maintenant à la création d'un fichier XML, qui permettra à d'autres sites web d'utiliser les news éditées par le nôtre. La question qu'il nous faut nous poser est : de quelles informations les utilisateurs de notre fichier XML auront-ils besoins pour l'utiliser de façon optimale ? Il y a d'abord les informations concernant notre site lui même :

- Le titre
- L'url
- Eventuellement la langue et/ou un logo

Puis les informations concernant chaque news :

- Le titre
- L'url

Imaginons que ces dernières informations (concernant les news), se trouvent dans notre base de données dans la table "nouvelles" contenant les champs :

- ID, la clef primaire
- titre, le titre de la news
- contenu, le contenu

et que chacune de ces news est disponible via une URL du type http://www.notrewebsite.xx/news.php?id_news=[ID]

Pour que la création du code PHP mettant à disponibilité le fichier XML soit bien claire, nous allons d'abord nous figurer à quoi devra ressembler le fichier XML (que nous nommerons backend.php), selon ce que l'on vient de définir :

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<!DOCTYPE rss PUBLIC "-//Netscape
Communications//DTD RSS 0.91//EN"
"http://my.netscape.com/publish/formats/
rss-0.91.dtd">

<rss version="0.91">
<channel>
  <website>
   <title>Microsoft et la sécurité</title>
   <link>http://www.notrewebsite.xx</link>
   <logo>http://www.notrewebsite.xx/logo.gif
   </logo>
  </website>
  <news>
   <item>
    <title>Anti-Virus pour le dernier ver
          Windows XP</title>
    <link>
    http://notrewebsite.xx/news.php?id_news=26
    </link>
   </item>
   <item>
    <title>Patch de sécurité pour Windows
          XP</title>
    <link>
    http://notrewebsite.xx/news.php?id_news=24
    </link>
   </item>
   <item>
    <title>HOAX : Windows serait sécu-
          risé</title>
    <link>
    http://notrewebsite.xx/news.php?id_news=22
    </link>
   </item>
  </news>
</channel>
</rss>
```

On a ici fixé une limite de cinq nouvelles disponibles, de façon à ne pas alourdir l'exemple. Ici la balise racine pourrait être "rss" comme "channel", mais comme <rss> est là par convention pour définir la version utilisée, on choisira plutôt <channel>. Voici le code PHP qui a généré ce fichier XML :

```
<?
header("Content-Type: text/xml");

echo "<?xml version=\"1.0\"
          encoding=\"ISO-8859-1\"?>\n\n";
echo "<!DOCTYPE rss PUBLIC \"-//Netscape
     Communications//DTD RSS
     0.91//EN\"\n";
echo " \"http://my.netscape.com/publish/".
     "formats/rss-0.91.dtd\">\n\n";
echo "<rss version=\"0.91\">\n\n";

echo "<channel>\n";
echo "  <website>\n";
echo "   <title>Microsoft et la sécu-
          rité</title>\n";
```


```
echo "<link>http://notrewebsite.xx</link>\n";
echo " <logo>http://notrewebsite.xx/logo.gif</logo>\n";
echo "   </website>\n";
$result = mysql_query(
 "SELECT id, titre FROM nouvelles
   LIMIT 5");
echo "   <news>\n";

while (list($id, $titre) =
 sql_fetch_row($result, $dbi)) {
     echo "    <item>\n";
     echo "     <title>$titre</title>\n";
     echo " <link>http://notrewebsite.xx/".
           "news.php?id_news=$id</link>\n";
     echo "    </item>\n";
}
echo "   </news>\n";
echo "</channel>\n";
echo "</rss>";
?>
```

On définit d'abord le type du fichier grâce à l'header "Content-Type" qui vaut "text/xml".

Puis on affiche les lignes cocnernant le fichier XML lui même, suivi des informations que l'on a choisi de rendre disponibles au sujet de notre website.
On extrait ensuite les cinq dernières news de la table "nouvelles" via MySQL, puis on les affiche selon la structure choisie, pour enfin fermer les balises ouvertes au début du fichier.

Comme vous avez pu le remarquer, la création d'un fichier XML est extrêmement simple. Et malgré cela, son dynamisme et son efficacité ne sont pas blousés. Il ne reste plus qu'à un autre webmaster de créer un petit script qui lira, extraira (via des expressions régulières, cf. page 14) et affichera les news de votre site.

Quelques URLs qui pourront vous êtres utiles :
http://www.chez.com/xml/ (apprendre le XML)
http://xml.developpez.com (club d'entraide des codeurs francophones)
http://www.xmlfr.org (l'espace XML francophone)
http://www.phpsecure.info/v2/.php?zone=pBackend (exemple d'extraction de données d'un fichier XML)

frog-m@n

le saviez-vous ?

Le W3C – World Wide Web Consortium, fondé en 1994, est un groupement international d'organisations diverses (entreprises, centres de recherches...). Il a pour charge de définir les standards technologiques en utilisation sur la toile.
En 1997 il publie les recommandations pour HTML 3.2, qui devient HTML 4.01 en DÉcembre 1999. Au dÉbut de l'annee 2000 c'est l'annonce des recommandations pour XHTML. Le besoin de disposer de dynamisme se fait déjà ressentir, HTML vieillit.
C'est dans cette même période, en Février 98, que se diffusent les premieres recommandations du XML, alors version 1.0.
Pour consolider l'édifice en construction, le W3C publie le DOM, le Document Object Model. De DOM level 1 en 1998 a DOM level 3 en 2004, cette API n'a cessé de fournir plus d'efficacité pour faciliter le traitement et la manipulation des langages web. Le DOM ne précise pas le langage, mais les méthodes d'accès aux représentations du langage pour permettre plus facilement la construction de pages et la récuperation de données HTML, XML.

Les expressions régulières en PHP

Les expressions régulières - ou rationnelles en bon Français - constituent un outil puissant pour le développeur qui doit reconnaître, rechercher ou remplacer des motifs particuliers dans des chaînes de caractères. Elles sont particulièrement adaptées à la validation et au filtrage des données utilisateur, et donc indispensables à un programmeur PHP sérieux.

Une expression régulière (ou Regex, pour l'anglais Regular Expression) est une expression qui va correspondre à une certaine forme de chaînes de caractères, par exemple une adresse mail. En interne, il s'agit d'une machine d'état, mais en pratique, il suffit de combiner des groupements logiques de caractères (chiffres, mots, etc.).

Premiers symboles, premières expressions

Le tout premier symbole que l'on étudie quand on aborde les regex est le point (.) Il désigne non pas le signe de ponctuation, mais l'expression tout caractère. Pour cibler un peu plus, nous pouvons décider de chercher une lettre majuscule, une lettre minuscule, ou un chiffre. Respectivement, les symboles désignant ces occurrences sont [A-Z],[a-z] et [0-9]. Encore trop vague ? De la même manière, un nombre hexadécimal est composé de chiffres de 0 à 9 ou de lettres A à F (en minuscule ou en majuscule). On les reconnaîtra donc avec l'expression : [A-Fa-f0-9].
Les caractères ^ et $ non échappés (c'est à dire non précédés d'un \, nous y reviendrons plus tard) désignent respectivement le début et la fin d'une ligne. ^[A-Z][a-z]$ désigne alors une ligne qui ne contient qu'un mot de deux lettres commençant par une majuscule (comme on peut en trouver dans les dictionnaires de cruciverbistes).

Plus de précision

Vous savez maintenant rechercher un caractère qui ferait partie d'un groupe de caractères bien défini. Il s'agit là d'une première expression régulière, sachant que les regex qui vont suivre ne sont en fait que l'assemblage de plusieurs. Pour nous aider, il existe des outils, qu'on appelle les jokers : ***, ? et +**. Ils vont vous permettre de définir combien de fois l'objet représenté par l'expression doit se répéter. Accoler + à une regex permet de rechercher la présence d'une ou plusieurs fois cette expression régulière.
Par défaut, ces jokers essayent de couvrir l'espace le plus grand possible (greedy - avide). Mais ce mode peut-être inversé en rajoutant le caractère ? au joker. Si la chaîne était _kljhsdkds123123sdqds_, alors la recherche de [0-9]+ renverrait 123123 car c'est la plus grande occurrence qui la satisfasse.

? : il s'agit presque de l'inverse de +. Si ce dernier cherche _au moins une fois_ une correspondance à l'expression régulière qu'il suit, ? en cherche au plus une occurrence (on utilise aussi l'expression : zéro ou une fois). [1-9] ? correspond donc à au plus un chiffre, et permet de trouver une occurrence telle que "7" (si 24 se trouvait dans la chaîne, on aurait deux occurrences : 2 et 4).

*** :** permet de tester la présence ou non de l'expression régulière qu'il suffixe. [HACK]* correspond à la recherche _de la lettre H, ou de A, ou C, ou de K, zéro ou plusieurs fois, et permettra de trouver une occurrence telle que "CHACHACHAK", ou simplement "C".

Espace

Ensuite, restent ces caractères que nous ne savons pas encore précisément cibler : les espaces. D'abord, vous pouvez inclure simplement l'espace dans la regex : ([a-z] [a-z])* correspond à la recherche _d'une minuscule, suivie d'un espace et d'une autre minuscule, le tout répété ou non. Sachez qu'il existe des raccourcis pour représenter certains groupes et leurs opposés. \d représente un chiffre tandis que \D représente tout sauf un chiffre. De même, on trouve \w et \W pour représenter tout caractère alphanumérique ou underscore et son contraire. Enfin, \s représente un espace (que ce soit un simple espace ou une tabulation), et possède lui aussi son contraire.

Code

Voilà pour les rudiments. Passons au PHP. Nous allons pour cela utiliser une expression régulière assez simple, qui va

correspondre à une adresse mail. On peut décrire ce genre d'adresse par la phrase :" une adresse mail est une suite de caractères alphanumériques (ou underscores) suivi d'un arobase, lui-même suivi d'une chaîne de caractères alphanumériques ou d'underscores, puis d'un point est d'une extension de 2 à 4 lettres ". Voici l'expression régulière correspondante : (\w)+@(\w)+.([A-Za-z]{2,4}). Remarquez la présence d'un antislash devant le point. Cette fois, on désigne bien le signe de ponctuation.

Les regex ont trois utilités majeures : tester la présence de chaînes de caractères dans un texte, récupérer une chaîne d'une forme particulière ou les remplacer par d'autres. Pour le faire, on utilise en PHP des fonctions dont le préfixe est preg_ (Perl compatible Regular exceptions).

preg_match

preg_match() prend deux paramètres : l'expression régulière et la chaîne dans laquelle nous cherchons une occurrence. Un exemple : testons la présence d'adresse mail dans une chaîne.

```
$chaine = "Contact :
Koreth@TheHackademy.net"
$regex = "/(\w)*@(\w)*.([A-Za-z]{2,4})/"
if ( preg_match($regex,$chaine,$store) ) {
  echo "email: ".$store[0][1];
} else die("Pas d'email !");
```

Notez la présence de deux / encadrant la regex : c'est comme ceci que s'écrivent les regex en perl et donc le module _preg_ utilise aussi cette syntaxe. Voir aussi http://fr.php.net/preg_match_all qui permet de remplir un tableau ($store) avec de multiples occurrences du modèle. Ici, on pourrait se passer du paramètre $store si l'on se contente de tester la présence d'une adresse.

preg_replace

Le remplacement est lui aussi assez simple. Nous allons faire appel à une fonction à laquelle nous donnerons une expression régulière et la chaîne par laquelle remplacer toute occurrence.

```
$chaine = "Contact :
Koreth@TheHackademy.net ou
CodeJ@TheHackademy.net";
$regex = "/(\w*)@(\w*).([A-Za-z]{2,4})/";
$count = 0;
print preg_replace($regex,"$1-nos-
pam@$2.$3",$chaine,-1,$count)
```

print "$count éléments ont été remplacés".

Le but est d'ajouter au nom d'utilisateur de toutes les adresses mail de $chaine le suffixe "-nospam". On voir que l'on fait références aux trois groupes désignés entre parenthèses dans la regexp (l'utilisateur, le domaine, et le tld). Il son numérotés dans l'ordre, il n'y a donc pas d'ambiguïté.

Attention cependant : cette expression régulière est très imprécise dans la modélisation d'une adresse email, et ne tient pas compte de certaines exceptions.

Exemples Adresse IP

On peut utiliser simplement :

```
\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b
```

Mais le problème est que 999.0.0.0 serait alors correct.

Si on veut être scrupuleux, ça se complique :

```
\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)\b
```

Certaines fois, il est peut-être plus simple d'utiliser une opération comme split() sur la chaîne, et de vérifier les nombres un par un.

Tags HTML

On peut utiliser, pour une balise particulière :

```
<TAG[^>]*>(.*?)</TAG>
```

Ou sinon, le cas général :

```
<([A-Z][A-Z0-9]*)[^>]*>(.*?)</\1>
```

Cependant, il s'agit d'une solution de fortune et peu performante. Si vous devez parser de l'html, utilisez un parser html (c'est encore plus confortable d'utiliser une implémentation du DOM, cf : www.w3.org/DOM et www.php.net/dom). Cependant, dans des cas simples où la fiabilité n'est pas un élément critique, cela peut faire l'affaire. On peut par exemple extraire des liens avec quelque chose comme : `/<a href=\"([^\"]*)\">(.*)<\/a>/iU` .

Supprimer des doublons

On peut utiliser preg_replace avec l'expression suivante, par exemple : ^(.*)(\r?\n\1)+$. En remplaçant par \1, on supprime ainsi les doublons ("stupidstupid" devient "stupid").

Client-side

Il est aussi possible d'utiliser des expressions régulières dans de nombreux autres langages, et notamment en Javascript. Ce peut être une bonne manière d'implémenter une validation préliminaire, avant l'envoi d'un formulaire. Voici un exemple simple :

```
<script type="text/javascript">
function validate(form) {
 var pattern = /^\d+$/;
 var str = form.age.value;
 if(str.search(pattern) == -1)  {
   alert("Veuillez entrer un âge en chif-
fres");
   return 0;
 } else {
   return 1;
   //form.submit();
}
</script
```

Koreth

Créer des images dynamiques

PHP permet de créer des images générées à la volée, à l'aide d'une bibliothèque prévue à cet effet : GD. Elle permet de créer et de manipuler des fichiers graphiques, comme des graphes ou des logos contenant du texte dynamique.

La librairie de fonctions GD permet de créer assez facilement des fichiers au format gif, png ou jpeg, en fonction par exemple de données stockées dans la base de données. Il faut tout de même savoir que ce genre de procédé met à rude épreuve le processeur, il faut donc utiliser ces fonctions à bon escient (par exemple pour des diagrammes statistiques à barre, des graphiques sectoriels, ...), et prévoir de cacher le résultat si la charge est trop grande.
Les fonctions de la librairies GD permettent de retourner au navigateur une image plutôt qu'une page HTML. Pour que le navigateur sache qu'il s'agit d'un fichier de type GIF, par exemple, la premiere chose à faire (avant d'envoyer n'importe quelle autre information au navigateur) est d'envoyer un en-tête HTTP indiquant le type MIME du fichier, c'est-à-dire : headers("Content-Type: image/gif");
GD permet de travailler sur une image abstraite, crée grâce à la fonction imagecreate(). Vous pouvez ensuite lui donner le format désiré, avec la fonction imagegif() ou imagepng() par exemple.
La librairie GD fournit un panel de fonctions permettant de créer des formes primaires telles que des rectangles, des ellipses (donc des cercles), des arcs, des lignes, etc.
Généralement ces fonctions admettent en premier paramètre l'identifiant de l'image dans laquelle la forme doit être créée, puis les coordonnées permettant de générer la forme, et enfin la couleur de l'élément. Elles retournent 1 si l'élément a pu être dessiné, 0 dans le cas contraire

Voici un exemple simple qui crée une image contenant un polygone.

```
<?php
 // Création de l'objet image
 $img = imagecreate(160,100);
 // Définition des couleurs utilisées
 $fond = imagecolorallocate($img,0xEF,0xF2,0xFB);
 $noir = imagecolorallocate($img,0,0,0);

 // Fond de l'image
 imagefill($img,0,0,$fond);
 // Dessin de la forme en fonction des
    coordonnées des points qui la composent
 imagefilledpolygon($image,
     array(80,15,45,85,125,85),3,$noir);

 // Envoi du header (pour informer le
    navigateur du type de document)
 header("Content-Type: image/gif");
 // Envoie des données au format GIF
 imagegif($image);
?>
```

Cet article s'inspire en grande partie du document intitulé « PHP - Génération d'images » issu de l'encyclopédie informatique Comment Ça Marche (www.commentcamarche.net), mis à disposition sous les termes de la licence Creative Commons, et que nous vous conseillons de consulter.
Vous pouvez copier, modifier des copies de cet article, dans les conditions fixées par la licence, tant que cette note apparaît clairement.
Voir http://www.commentcamarche.net/php/phpimg.php3 ainsi que les autres documents relatifs à PHP.

INSTALLER GD

Pour pouvoir utiliser ces fonctions il faut que PHP soit installé avec l'extension GD, c'est-à-dire passer le paramètre --with-gd à configure lors de la compilation (une version de GD est distribuée avec PHP). Sur la majeure partie des distribution, vous pouvez vous contenter d'installer le package php-gd ou équivalent, qui se chargera de tout configurer pour vous.
Sur Windows, vous devez vous assurer que le fichier php_gd2.dll est bien présent dans le répertoire de PHP et que la ligne correspondante dans php.ini est bien activée (et non commentée ; faire une recherche sur 'php_gd2.dll' avec votre éditeur). Avec EasyPHP, il suffit d'activer GD à partir du menu.
Pour plus d'informations : http://fr.php.net/gd

Moteur de

Quand on se lance dans la réalisation d'un site, on doit en permanence jongler entre le PHP et le HTML, ce qui rend le code source tout simplement illisible. Il existe une solution permettant de travailler sur le code PHP tout en gardant un code HTML clair : utiliser un moteur de templates.

Introduction

Ceux d'entres-vous qui ont déjà goûté aux joies de la création de sites webs dynamiques, notamment grâce à PHP, ont sûrement été au moins une fois perdu dans leurs pages, car il s'y mélangeait du code HTML avec du code PHP. La solution serait d'avoir d'un côté le code HTML que l'on peut modifier tranquillement sans risquer de prendre un bug de pleine face, et de l'autre le code PHP, clair, net et commenté. C'est chose possible grâce à ce qu'on appelle des moteurs de templates. Cet article sera basé sur le moteur ModeliXe, dont vous pouvez consulter le site [1].

1 – Le moteur de templates

Le moteur de templates est en fait un module PHP à inclure à chaque page l'employant. Ce module fait l'interface entre une template (autrement dit une page XHTML, et non plus HTML) et le code PHP, ce qui permet de traiter séparément les deux langages. Cette template contient des balises spécialement formatées, qui seront par la suite utilisées par le moteur pour générer dynamiquement le contenu, et permettre ainsi au navigateur de l'afficher.

Car c'est ça le secret du moteur de templates, c'est une surcouche du PHP, écrit en PHP, ce qui le rend formidable J'ai déjà parlé de l'avantage indéniable qu'offre ces moteurs de templates, c'est à dire celui de séparer le code HTML du code PHP, mais en approfondissant l'étude de ces moteurs, on voit qu'ils offrent de nombreuses autres facilités ! Par exemple, ModeliXe propose une méthode simple pour créer facilement des tableaux de toutes tailles, et évite ainsi de répéter tout un bloc de balises <tr> et <td>. Et ça fonctionne aussi avec n'importe quel autre bloc de code HTML. L'utilisation de ModeliXe se fait en grande partie en PHP objet, c'est d'ailleurs ce qui fait sa force.'

a) Installation et configuration de ModeliXe

Comme nous allons travailler avec ce moteur de templates tout au long de cet article, je vais vous guider dans son installation et sa configuration. Tout d'abord, récupérez l'archive sur le site officiel [2], puis désarchivez le dossier ModeliXe dans le dossier de votre site web (local ou sur internet via un client FTP). Créez ensuite un répertoire "cache" et un autre nommé "template". Toute la difficulté réside dans le fichier MxConf.php.

b) Test de fonctionnement

Pour vérifier le bon fonctionnement du moteur de templates, nous allons créer un fichier template que vous mettrez dans le répertoire /template/ destiné justement... aux templates. Pour notre test, nous allons faire un simple HelloWorld et l'afficher grâce à ModeliXe. Cet exemple n'est pas très utile, mais permettra de tester la bonne configuration du moteur.

```
[fichier : helloworld.mxt]
<html>
<head>
 <title>Premier test avec ModeliXe</title>
</head>
<body>
<center>hello world !</center>
</body>
</html>
[/fichier]
```

Puis nous allons créer une page index, en PHP, utilisant notre moteur et affichant le template.

```
[fichier : index.php]
<?php
include('ModeliXe.php');
// on crée une page selon le template
$page = new ModeliXe('helloworld.mxt');
// on l'analyse
$page -> SetModeliXe();
// et on l'affiche
$page -> MxWrite();
?>
[/fichier]
```

templates

Ok, c'est un peu plus long qu'un simple Hello World en PHP. Par contre c'est beaucoup plus clair, et l'emploi du PHP objet simplifie grandement le code.

2 – Les templates de ModeliXe

Je vous vois venir avec vos gros sabots, et c'est pourquoi je vais vous rassurer : oui, ModeliXe est simple d'emploi. Ce moteur de templates est facile à intégrer à un site, et rend bien des services, foi de webmaster. Le site sur lequel je travaille, oldhopes.tuxfamily.org pour ne pas le nommer, utilise ces templates. Vous pourrez constater qu'en aucun cas le moteur de templates ralenti le site, et qu'il permet de tout faire en transparence.

Mais avant d'en arriver à ce stade, il serait peut-être intéressant de voir la conception d'une template, et comment manipuler des templates en PHP, grâce au moteur. Attelons-nous à cette tâche (qui a dit "ingrate" ?).

Une template n'est, comme vous avez pu le voir, rien d'autre qu'un document XML, avec une extension "mxt" par exemple (mais ce n'est pas obligatoire, vous pouvez mettre n'importe quelle extension). La template servant à la génération dynamique de la page, on y définit donc des éléments XML hiérarchisés, permettant d'une part d'être adressés par un nom, et d'autre part d'être paramétrés via le moteur. Le marquage typique pour ModeliXe est le suivant :

```
<xml:text|img|... [propriétés]/>
```

La balise indiquant à ModeliXe un élément à évaluer est toujours de cette forme. Nous allons voir par la suite les principaux éléments disponibles et comment s'en servir dans des exemples concrets.

a) Le texte simple

Pour pouvoir intégrer dans la page un texte simple, cette balise XML suffit :

```
<mx:text id="mon_texte"
         class="texte gras"/>
```

Vous pourrez remarquer que les propriétés applicables au texte final en HTML sont présentes dans la balise XML, ce qui simplifie l'écriture. Chaque objet du template doit se voir assigner un ID permettant de le désigner.

b) L'image

Pour intégrer une image dans la page, cette autre balise suffit :

```
<mx:img id="mon_image"/>
```

Encore une fois, l'objet possède un ID et l'attribut "img" spécifie qu'il s'agit d'une image. Rien de bien compliqué jusqu'ici.

c) Le bloc

Là, on entre dans la partie intéressante. ModeliXe permet de définir des blocs, blocs qui peuvent être imbriqués afin de pouvoir gérer une structure dans la page. Un bloc est défini par un ID et doit obligatoirement posséder une balise de fin, à son ID, contrairement aux exemples précédents qui respectaient la norme XHTML 1.0. Cet exemple sera assez explicite :

```
<mx:bloc id="Page">
      <mx:text id="Titre" class="cen-
ter"/><br/>
      <mx:img id="Image" border="0"/>
</mx id="Page">
```

Le document étant hiérarchisé, l'adressage de l'objet "Titre" se fera via le bloc page, on notera donc la référence à l'objet "Titre" : Page.Titre. De même pour l'objet "Image" (Page.Image vous l'aurez deviné je pense). Bon, nous avons vu les principaux objets. Notez bien qu'il y en a d'autres, mais que je ne traiterai pas en totalité toutes les possibilités de ModeliXe.

3 – Intégration des templates

Il est désormais temps d'intégrer nos templates dans le code PHP. Pour ce faire, la classe ModeliXe propose un ensemble de fonctions simples d'utilisation, permettant d'agir sur les éléments XML définis dans nos templates. Réalisons une template qui nous servira d'exemple, que nous nommons exemple.mxt. Cet exemple servira à afficher des informations sur un auteur de notre cher mag ;)

```
[fichier : exemple.mxt]
<html>
<head>
<title>Exemple ModeliXe</title>
</head>

<body>

<mx:bloc id="Page">
 <xml:text id="Titre" class="center"/>
 <br/><br/>

 <mx:bloc id="Infos">
  <table border="0">
   <tr><td><b>Pseudo</b></td>
       <td><b>E-mail</b></td>
       <td><b>Niveau</b></td></tr>
   <tr><td><mx:text id="Pseudo"
            class="texte"/></td>
       <td><mx:text id="Email"
            class="texte"/></td>
       <td><mx:text id="Nivo"
            class="texte"/></td></tr>
  </table>
 </mx:bloc id="Infos">

  <br/><br/>
 <mx:text id="Ip" class="center"/>
</mx:bloc id="Page">

</body>
</html>
[/fichier : exemple.mxt]
```

Cette template affichera des informations sur un auteur. Créons maintenant la page PHP qui utilisera cette template.

```
[fichier : auteurs.php]
<?php

include('ModeliXe.php');

$page = new ModeliXe('exemple.mxt');
$page -> SetModeliXe();

//on met le titre
$page -> MxText('Page.Titre',
                'Exemple avec ModeliXe');
//et le nom de l'auteur
$page -> MxText('Page.Infos.Pseudo',
                'virtualabs');
$page -> MxText('Page.Infos.Email',
                'virtualabs@gmail.com');
$page -> MxText('Page.Infos.Nivo',
                'Z&eacute;ro');

//et on affiche la page
$page -> MxWrite();

?>
[/fichier : auteurs.php]
```

Comme vous pouvez le constater, pour paramétrer des objets de type texte, on emploie la méthode MxText(). Cette méthode prend pour premier argument une référence à l'objet, comme détaillé précédemment (selon la hiérarchie donc), puis le texte à y mettre. Je tiens à préciser que dans mon exemple, les valeurs à afficher sont fixées, mais elles peuvent très bien être issues d'une base de données. Et là miracle, pas de code HTML dans la page PHP. C'est clair et concis.
Pour les images, c'est un peu plus délicat. Il faut employer la méthode MxImage(@Reference, @fichier_img, @alternative où @Reference représente la référence de l'objet, @fichier_img le chemin du fichier à afficher et @alternative le texte alternatif (pour les vieux navigateurs n'affichant pas les images, comme Lynx).

Je vous parlais avant cela du grand avantage des blocs, c'est le moment de s'y essayer. Le principe est basé sur le fait que ModeliXe permet de répéter un bloc défini, c'est à dire d'en créer une copie conforme, à la suite d'un bloc déjà traité. Cela permet de faire facilement des centaines de rangées à un tableau sans avoir à prévoir autant de rangées dans la template (et cela évite donc des heures d'un boulot harassant). Pour cela, on utilise la méthode MxBloc qui permet de boucler un bloc. Tout d'abord, modifions notre template de manière à définir un bloc pour une rangée :

```
[fichier : exemple2.mxt]
<html>
<head>
<title>Exemple n°2 ModeliXe</title>
</head>

<body>
<mx:bloc id="Page">
  <xml:text id="Titre" class="center"/>
  <br/>
  <br/>

  <table border="0">
      <tr><td><b>Pseudo</b></td>
          <td><b>E-mail</b></td>
          <td><b>Niveau</b></td>
          </tr>
```

```
    <mx:bloc id="Auteur">
        <tr><td><mx:text id="Pseudo"
                class="texte"/></td>
            <td><mx:text id="Email"
                class="texte"/></td>
            <td><mx:text id="Nivo"
                class="texte"/></td>
            </tr>
    </mx:bloc id="Auteur">

    </table>
    <br/>
    <br/>
    <mx:text id="Ip" class="center"/>
</mx:bloc id="Page">
</body>
</html>
[/fichier : exemple2.mxt]
```

On considère ensuite que l'on a un tableau associatif d'auteurs à afficher. On concevra donc notre page PHP de cette manière :

```
[fichier : auteurs.php]
<?php

include("ModeliXe.php");

$page = new ModeliXe("exemple2.mxt");
$page -> SetModeliXe();

//on met le titre
$page -> MxText("Page.Titre",
                "Exemple avec ModeliXe");

$auteurs[0]['pseudo']='virtualabs';
$auteurs[0]['email']=
    'virtualabs@gmail.com';
$auteurs[0]['niveau']='zero';

$auteurs[1]['pseudo']='Xelory';
$auteurs[1]['email']='xelory@hotmail.com';
$auteurs[1]['niveau']='zero aussi';

// et le nom des auteurs
for($a=0;$a<2;$a++){
    $page -> MxText("Page.Auteur.Pseudo",
                    $auteurs[$a]['pseudo']);
    $page -> MxText("Page.Auteur.Email",
                    $auteurs[$a]['email']);
    $page -> MxText("Page.Auteur.Nivo",
                    $auteurs[$a]['niveau']);
    $page -> MxBloc("Page.Auteur","loop");
}

//et on affiche la page
$page -> MxWrite();

?>
[/fichier : auteurs.php]
```

Cette fois-ci, un tableau de deux lignes apparaît, et même plus si l'on utilise une base de données ! Sans pour autant avoir rajouté énormément de code HTML, ni PHP. Le code PHP reste très lisible, tout comme le HTML, et la maintenance des deux en est simplifié. Quand je vous disais que c'était simple...

Conclusion

Les moteurs de templates sont très utiles, même s'ils demandent plus de rigueur, car ils proposent des fonctions adaptées au besoin des webmasters. Ils permettent aussi une maintenance plus aisée du site, ce qui permet de retrouver et de corriger rapidement les bugs, ou encore de changer facilement sans tout chambouler l'organisation de l'interface du site, indépendamment du code gérant son dynamisme. Je pense que c'est une solution aussi valable pour les entreprises, car ces moteurs sont vraiment éprouvés, comme ModeliXe, bien qu'ils ne soient pas très connus. Vous trouverez bien sûr la documentation complète et détaillée sur le site officiel [1], cet article n'étant qu'une petite démonstration de la puissance de ces moteurs.

[1] http://modelixe.phpedit.com
[2] http://modelixe.phpedit.com/download/

Virtualabs

Simuler registe
Coder compatib

Depuis PHP 4.0.2, les variables post, get, etc. ne sont plus enregistrées comme des variables globales accessibles directement. Dans ces conditions, de nombreux scripts cessent de fonctionner, sur certains hébergeurs gratuits par exemple. Il y existe des moyens de rendre ces scripts compatibles, certains plus sûrs ques d'autres...

Les règles de priorités qui régissent l'attribution des variables globales dans un script sont souvent mal connues et sont donc sources ne nombreuses failles de sécurités. Après avoir éclairci ces points, importants en programmation PHP, nous verrons comment coder de manière sûre un substitut à register_globals=on, et les pièges à éviter.

Introduction aux super-globales

Il y a plusieurs sortes de variables en php : les variables définies dans les scripts, celles provenant de la requête http (GET, lorsqu'elles sont dans l'url, POST pour les formulaires, plus les variables contenues dans les cookies) et celles qui sont générées par le serveur. Il peut arriver que des variables de sortes différentes aient le même nom. On peut cependant les différencier en spécifiant explicitement leur provenance. Les tableaux $_GET (ou $HTTP_GET_VARS), $_POST (ou $HTTP_POST_VARS), $_COOKIE (ou $HTTP_COOKIE_VARS) et $_REQUEST (qui est la synthèse des trois précédents) contiennent les variables envoyées par http. Le tableau $_GLOBALS contient les variables définies dans le script ou... autre chose que nous verrons par la suite. Il y a aussi $_ENV (variables d'environnement de l'OS) et $_SERVER (adresse du client, referer, etc.), mais ils ne seront pas utilisés dans ce texte.

Lors qu'il n'y a pas d'ambiguïté, et lorsque register_globals est enclenché dans la configuration de PHP, on peut accéder à une variable donnée dans l'url : (http://site.com/index.php?myVar=0) aussi bien avec $myVar qu'avec $_GET['myVar']. Cette équivalence provoque souvent des failles de sécurité dans les scripts qui n'initialisent pas correctement certains variables (on pense que c'est une variable interne au programme, alors que l'utilisateur peut en réalité modifier sa valeur).

Par contre, lorsque des variables de sources différentes (par exemple, GET et POST) ont le même nom, la valeur de la variable globale est attribuée selon une priorité donnée. On va voir dans cet article que cela est également une source de failles de sécurité.

Les priorités

Imaginons un fichier : http://www.target.url/priorites.php contenant le code suivant :

```
<?
echo "GET : ".$HTTP_GET_VARS["MyVar"];
echo "\n<br>POST : ".
     $HTTP_POST_VARS["MyVar"];
echo "\n<br>COOKIE : ".
     $HTTP_COOKIE_VARS["MyVar"];
echo "\n<br>GLOBAL : ".$MyVar;
?>
```

On peut tester les priorités en envoyant une requête contenant la variable MyVar sous plusieurs formes (get, post et cookie). Pour cela, on utilise un script python (voir encadré).

On obtiendra alors comme résultat côté client :

```
GET : GetValue
<br>POST : PostValue
<br>COOKIE : CookieValue
<br>GLOBAL : CookieValue
```

On peut donc en conclure qu'une variable COOKIE est en quelque sorte plus " puissante " que les variables GET et POST, car c'est à elle qu'une variable globale prendra sa valeur en priorité.

Avant de parler des conséquences, voyons les priorités

r_globals=on ? le et sécurisé

PHPPrioCheck.py

Ce programme permet d'envoyer une requête http initialisant la variable MyVar de différentes manières :

```
import httplib

http=httplib.HTTP("www.target.url")

http.putrequest("POST","/priorites.php?MyVar=GetValue")
http.putheader("Content-Type","application/x-www-form-urlencoded")
http.putheader("Cookie","MyVar=CookieValue")
http.putheader("User-Agent","PHPPrioCheck.py")
http.putheader("Host","www.target.url")
http.putheader("Content-Length",str(len("MyVar=PostValue")))
http.endheaders()

http.send("MyVar=PostValue")

code,msg,headers = http.getreply()

print code,"\n",msg,"\n",headers

file=http.getfile()

print "Result : \n"+file.read()
```

entre GET et POST, en supprimant simplement la ligne : http.putheader("Cookie","MyVar=CookieValue").
Ce qui donne le résultat :

```
GET : GetValue
<br>POST : PostValue
<br>COOKIE :
<br>GLOBAL : PostValue
```

Les variables du tableau $_REQUEST réagissent exactement de la même façon que les variables du tableau $_GLOBALS.
On peut maintenant définir l'ordre des priorités :

1. COOKIE
2. POST (comprenant le tableau $_FILES, les fichiers envoyés par formulaire)
3. GET

Cet ordre est en fait défini lui aussi dans le php.ini par l'option " variables_order " qui est par défaut " EGPCS ", c'est-à-dire Environnement, GET, POST, COOKIE, Serveur. L'ordre de priorité pour les variables de type GPC est donc bien par défaut celui que nous avions déduit.
Les conséquences de ces priorités peuvent être dangereuses si, dans un code PHP, on fait des vérifications sur une variable dont on spécifie le type, puis que par la suite on l'utilise sans spécifier le type.

Par exemple avec ce code (include.php) :

```
<?
if( eregi("\.\.",$_GET["file"])
    || eregi("\\",$_GET["file"])
    || eregi("/",$_GET["file"])
    || ereg("\\0",$_GET["file"]) ){
    die("Illegal access.");
} else {
    if ( file_exists("/files/".$file)
)
        include("/files/".$file);
}
?>
```

Ici, il suffit d'envoyer un cookie nommé " file " et contenant la valeur " ../../../../file/to/show " sur l'url : http://www.target.url/include.php?file=blabla pour inclure le fichier " ../../../../file/to/show ".
En effet, les tests vérifiant que la variable GET file ne contient ni " .. ", ni " \ ", ni " / ", ni " \0 " sont effectués sur la valeur " blabla ". Par contre, lors de l'inclusion, le type n'est pas donné, et vu les priorités, si l'on a donné le nom " file " à un cookie, c'est sa valeur qui sera prise en compte, alors qu'aucune vérification n'a été faite à son sujet. Pour un exemple réel, voir TrueGalerie qui permettait de copier et d'uploader des fichiers à cause des priorités (http://www.phpsecure.info/v2/tutos/frog/TrueGalerie.txt).

(http://www.phpsecure.info/v2/tutos/frog/TrueGalerie.txt).

Il est évident que ce genre de problème ne se posera pas avec un COOKIE, vu qu'il est en haut de la liste des priorités. Mais cet ordre peut être modifié.
Plusieurs solutions sont envisageables pour ne plus avoir de problèmes :

1. Utiliser uniquement des variables de type $_REQUEST, reprenant les 3 types ; GET POST et COOKIE.
2. Utiliser partout, sans se tromper, les bons types de variables (beaucoup plus dangereux ;)).
3. Ne JAMAIS définir le type de la variable.
4. Mettre register_globals à off et lire la suite de l'article :)

Simulation de register_globals

Il est possible en PHP, avec un petit code, de simuler l'option register_globals à on quand elle est à off. J'ai trouvé de nombreux moyens pour aller dans ce sens, mais aucun de valable pour simuler au contraire un register_globals à off alors qu'il est à on.
Ce genre de code est souvent utilisé dans des applications distribuées, permettant d'être compatible avec les deux configurations, ou par des webmasters dont l'hébergeur ne permet pas de changer la configuration du php.ini.
Un exemple est le code suivant :

```
<?
foreach ($_REQUEST as $key=>$value) {
  ${$key} = $value;
}
echo $boum;
?>
```

Grâce à lui, la variable $boum pourra être définie par GET, par POST ou par COOKIE, que l'option register_globals soit sur on ou sur off. Dans cet exemple, ${$key} aurait pu être remplacé par $_GLOBALS[$key].
Voyons maintenant l'utilisation possible des fonctions extract() et import_request_variables() :

```
<?
extract($_REQUEST);
echo $boum;
?>

<?
import_request_variables('GPC');
echo $boum;
?>
```

Etc...

Dans tous ces exemples, $boum aurait pu être remplacé par $_GLOBALS["boum"].
Tout ces codes sont très pratiques, mais ils peuvent poser de sérieux problèmes. On peut en effet redéfinir n'importe quelle variable globale déjà définie dans le script, quel que soit l'état de register_globals.
<
Imaginons un code tout simple :

```
<?
$adminpass="blob";
foreach ($_REQUEST as $key=>$value) {
  ${$key} = $value;
}

if (!isset($pass) {
  echo "<form method=\"POST\">
        Entrez le mot de passe :<br>";
  echo "<input name=\"pass\"><br>
        <input type="submit"></form>";
} else {
    if ( $pass == $adminpass ) {
      echo "Bienvenue dans la partie
            administration.";
    }
}
?>
```

Vu que le code dangereux se trouve après la définition de la variable $adminpass, il est possible de la redéfinir. On sera donc considéré comme admin avec une simple url : http://www.target.url/admin.php?adminpass=hop&pass=hop.
Ce genre de code ne pose donc pas de problème s'il se trouve avant toute définition de variable globale.
Voici trois applications, qui, avant d'être corrigées, utilisaient cette méthode, soit trois exemples à ne pas suivre...
a) Nuked-KlaN b1.5
Des variables pourraient être redéfinissables en global via GET, POST et COOKIE.
nuked.php :

```
function nk_globals($table) {
  if (is_array($GLOBALS[$table])) {
    reset($GLOBALS[$table]);
    while (list($key, $val)
           = each($GLOBALS[$table])) {
      $GLOBALS[$key] = $val;
    }
```

```
  }
}
globals.php :

nk_globals('HTTP_GET_VARS');
nk_globals('HTTP_POST_VARS');
nk_globals('HTTP_COOKIE_VARS');
nk_globals('HTTP_SERVER_VARS');
```

Le problème, dans ces scripts, est qu'une autre faille permet d'inclure globals.php dans n'importe quel module, et donc de pouvoir changer diverses variables de configuration avec des requêtes GET (http://www.phpsecure.info/v2/tutos/frog/Nuked-KlaN.txt).

b) XOOPS 2.0.5

Des variables globales pourraient être redéfinies via POST. edituser.php, imagemanager.php :

```
if (isset($HTTP_POST_VARS)) {
  foreach ($HTTP_POST_VARS as $k => $v) {
    ${$k} = $v;
  }
}
```

On peut ainsi redéfinir certaines variables locales avec une requête POST, parce que cette portion de code est précédée par d'autres initialisations.
(http://www.phpsecure.info/v2/tutos/frog/XOOPS2.0.5.txt)

c) Mambo Server 4.0.14

Des variables globales pourraient être redéfinies via GET, POST et COOKIE si register_globals est à off.
regglobals.php :

```
<?php
if (!ini_get('register_globals')) {
  session_start();

  // [...]
  while(list($key,$value)= each($_FILES))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_ENV))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_GET))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_POST))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_COOKIE))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_SERVER))
    $GLOBALS[$key]=$value;
  while(list($key,$value)=each($_SESSION))
    $GLOBALS[$key]=$value;

  foreach($_FILES as $key => $value) {
    $GLOBALS[$key]=
      $_FILES[$key]['tmp_name'];
    foreach($value as $ext => $value2) {
      $key2 = $key."_".$ext;
      $GLOBALS[$key2]=$value2;
    }
  }
}
?>
```

Dans plusieurs composants, on a :

```
include ("configuration.php");
[...]
include ("regglobals.php");
```

On peut donc modifier la configuration du programme (http://www.phpsecure.info/v2/tutos/frog/MamboServer.txt).

Solution

Comme solution, j'ai composé deux codes pouvant permettre de simuler register_globals à on quand il est à off, pouvant être placés n'importe où dans le script sans danger :

```
<?php
if (!ini_get("register_globals")){
  foreach ($_REQUEST as $k=>$v){
    if (!isset($GLOBALS[$k])){
      ${$k}=$v;
}}}
?>
```

Ce code ne s'exécute que si register_globals est à off. Ensuite il va vérifier que les variables de type $_REQUEST ne soient pas déjà définies dans le tableau des $_GLOBALS. Pour les variables dont ça n'est pas le cas, il va les y placer. L'avantage de cette possibilité est que l'on peut placer un filtre sur toutes les variables définies par l'utilisateur, ce qui n'est pas possible dans ma deuxième proposition :

```
<?php
extract($_REQUEST,EXTR_SKIP);
?>
```

La fonction extract() va définir tous les éléments du tableau donné en premier argument comme étant des variables globales, le tableau étant ici $_REQUEST. Le deuxième argument définit le type de l'extraction, ici EXTR_SKIP, c'est-à-dire qu'il n'écrase pas les variables globales déjà définies. Ce dernier argument est par défaut à EXTR_OVERWRITE, c'est-à-dire qu'il écrase les variables globales.

Conclusion

Voilà, tout ça prouve qu'il faut faire très attention avec les super-globales, quel que soit l'état de register_globals.

PHP dans v

Beaucoup de personnes connaissent PHP pour la programmation internet, mais il existe un autre aspect moins connu de ce langage. C'est la possibilité de créer des programmes indépendants pouvant être lancés depuis la ligne de commande.

Depuis la version 4.3.0 de PHP, il est possible d'utiliser une nouvelle interface de programmation d'applications serveur permettant d'exporter les fonctionnalités de PHP en dehors du Web. Ce SAPI s'appelle CLI - pour command line interface. Vous l'aurez compris, cela permet de lancer des applications PHP autonomes à partir de la ligne de commande.

À qui profite PHP-CLI ?

Toutes les personnes programmant en PHP apprécieront le fait de pouvoir travailler sur de nouveaux supports. Les webmasters pourront écrire des scripts d'installation pour leurs applications web et tous ceux qui hésitaient à apprendre PHP car son domaine d'application était restreint à Internet peuvent désormais se lancer dans ce langage.

Pourquoi utiliser PHP-CLI ?

PHP-CLI procure de nombreux avantages pour toutes les personnes désireuses de créer des scripts d'administration ou bien des applications graphiques grâce à php-gtk. Avec PHP-CLI, vous n'êtes plus obligé de vous investir dans l'apprentissage d'un nouveau langage car, dans la majorité des cas, PHP répondra à votre besoin.

Installation

Si vous êtes linuxien et que vous avez installé PHP depuis les sources, vous n'avez pas de soucis à vous faire à propos de PHP-CLI car celui-ci est automatiquement installé. En revanche si vous installez PHP avec le gestionnaire de paquetages de votre distribution, vous devrez certainement ajouter un nouveau paquetage correspondant à PHP-CLI.

Une petite recherche depuis votre gestionnaire de paquetage et le tour est joué (sous debian sarge par exemple, on utilisera apt-get install php4-cli). Une fois que votre installation sera terminée, vous disposerez d'un nouveau binaire appelé PHP. Pour connaître son emplacement, il suffit de lancer la commande suivante :

```
dd@laptop:~/programmation/php$ whereis php
php: /usr/bin/php /usr/share/php
/usr/share/man/man1/php.1.gz
```

Les utilisateurs de Windows trouveront un exécutable appelé php.exe dans lmeur répertoire d'installation de PHP (c:\php5\php-win.exe si vous avez suivi le tutorial d'installation de ce manuel). Vous devrez certainement modifier la variable d'environnement PATH pour pouvoir accéder à ce programme depuis n'importe quel emplacement de votre système.

Pour la suite de cet exposé, j'utiliserai PHP en version 4.3.9 sous Linux, mais vous pourrez adapter chacune des commandes à une version différente de PHP, que ce soit avec Linux ou Windows. Vous pouvez obtenir la version exacte de votre PHP en exécutant la commande suivante :

```
dd@laptop:~$ php -v
PHP 4.3.9-1 (cli) (built: Oct  5 2004
                    08:45:32)
Copyright (c) 1997-2004 The PHP Group
Zend Engine v1.3.0, Copyright (c) 1998-
2004 Zend Technologies
```

Propriétés

Lorsque vous utilisez PHP depuis la ligne de commande, certaines directives du php.ini sont automatiquement désactivées ou modifiées car elles ne présentent aucun intérêt dans un environnement utilisant un terminal (invite de commande MS-DOS ou terminaux Unix). Ces fameuses directives sont les suivantes :

`html_errors = Off`

-> En effet, il serait totalement absurde d'afficher des erreurs au format HTML dans votre console...

`register_argc_argv = On`

-> Cette directive permet de déclarer les variables $argc et $argv que l'on pourra utiliser dans nos scripts pour récupérer les arguments de la ligne de commande.

`implicit_flush = On`

-> Permet d'afficher directement les valeurs des fonctions print et echo.

`max_execution_time = 0`

-> Avec cette directive, le temps d'exécution de nos scripts devient illimité.

otre shell !

Il y a également trois constantes définies pour que l'on puisse utiliser l'ensemble des fonctions d'entrée-sortie. Ce sont respectivement STDIN pour gérer le flux d'entrée standard ouvert en lecture seule (permet de recevoir des données), STDOUT qui représente le flux de sortie standard ouvert en écriture seule (permet d'afficher des données à l'écran) et STDERR qui permet d'afficher les problèmes rencontrés dans un programme (permet également d'afficher des données à l'écran). Ces constantes s'utilisent de la même façon qu'un manipulateur de fichier, c'est-à -dire que l'on peut écrire une instruction telle que
" fwrite(STDOUT, "Affichage sur votre écran !!!\n"); ".

Nos applications vont passer du navigateur web à un interpréteur de commande, ce bouleversement d'environnement implique certains changements dans votre façon d'utiliser PHP. Par exemple, il n'est plus question de formater ces données en HTML et d'utiliser des balises telles que
 pour changer de ligne, on utilisera donc le caractère de nouvelle ligne classique " \n " .

Pour pouvoir gérer le terminal, c'est-à -dire en utilisant des couleurs dans vos applications ou bien pour écrire à des endroits particuliers, il est possible d'utiliser la bibliothèque ncurses spécialement conçue à cet effet. L'autre solution consiste à utiliser php-gtk pour créer des GUI (applications graphiques) et ne plus avoir à utiliser de terminal. Étant donné que la majorité des utilisateurs préfèrent les applications graphiques, cette fonctionnalité de PHP peut devenir très intéressante.

Notre utilitaire PHP propose plusieurs options d'utilisation. Vous pouvez afficher l'ensemble de ces options avec la commande " php -h ". Certaines d'entre elles servent uniquement à donner des informations sur la version de PHP ou les extensions disponibles comme les marqueurs " -i " et " -m ". Les options les plus intéressantes sont " -f " et " -r ", qui nous permettent respectivement d'exécuter un script PHP et d'écrire directement ces instructions sur la ligne de commande. Dans tous les cas, je vous suggère de lire la page de man de PHP afin de pouvoir exploiter les options de façon optimale.

La pratique

Commençons tout de suite par tester un script simple affichant un message sur le terminal :

```
<?
/* hello.php */
echo "Hello world !\n";
?>
```

Pour exécuter ce script, il suffit de se placer dans le même répertoire que celui-ci et de faire :

```
$ php -f hello.php
Hello world !
```

Pour lancer ce script, on aurait pu se passer de l'option " -f ", de même, l'extension .php n'est pas obligatoire. Si vous utilisez Linux, il est possible d'ajouter la ligne #!/usr/bin/php au début de vos scripts. Cette ligne s'appelle le " shebang ", si vous l'insérez et donnez les droits d'exécution sur ce fichier, alors vous pourrez lancer le script en faisant ./hello.php.
Voyons maintenant comment utiliser les différents paramètres que l'on passe au script. On utilisera la variable $argc ainsi que le tableau $argv qui sont définis automatiquement. Il est inutile de décrire leur fonctionnement, un exemple est bien plus parlant :

```
#!/usr/bin/php
<?
/* arguments.php */
echo "Le nom du script est $argv[0]\n";
for($i=1; $argv[$i] != ""; $i++)
 fwrite(STDOUT,
   "L'argument $i vaut $argv[$i]\n");
?>
```

Testons tout de suite son fonctionnement :

```
$ chmod +x arguments.php
$ ./arguments.php toto tutu tata titi

Le nom du script est ./arguments.php
L'argument 1 vaut toto
L'argument 2 vaut tutu
L'argument 3 vaut tata
L'argument 4 vaut titi
```

Dans notre dernier script, nous allons voir comment dialoguer avec l'utilisateur en lui demandant d'entrer des valeurs que nous analyserons. C'est le même principe que pour les formulaires que l'on retrouve dans les sites internet dynamiques.
Voyons cela avec le fichier add_user.php :

```
#!/usr/bin/php
<? /* add_user */ ?>
Ce script permet d'ajouter des utilisateurs
au fichier users.list ...
<?
$fp = fopen("users.list","w+");
fwrite(STDOUT, "Nom: ");
$nom = trim(fgets(STDIN));
fwrite(STDOUT, "Prenom: ");
$prenom = trim(fgets(STDIN));
$result = "$nom $prenom\n";

if(fwrite($fp,$result))
  fwrite(STDOUT,
   "\n---\n$nom $prenom a bien été
    ajouté!\n");
else
  fwrite(STDERR,
   "Une erreur s'est produite !!!\n");

?>
```

Une fois qu'on l'exécute, on obtient :

```
$ php add_user
Ce script permet d'ajouter des utilisateurs
au fichier users.list...
Nom: Mitnick
Prenom: Kevin
---
Mitnick Kevin a bien été ajouté!
```

Quand on regarde le fichier users.list, on voit bien qu'un utilisateur a été ajouté :

```
$ cat users.list
Mitnick Kevin
```

Ces quelques exemples montrent qu'il est très simple d'utiliser PHP-CLI et que, même si les performances sont moins bonnes que pour d'autres langages de scripts tels que Perl ou Python, on peut se servir de ce langage pour créer des scripts d'administration très complets.
Vous pouvez également utiliser l'option " -r " pour lancer du PHP sur la ligne de commande en se passant des balises <? ?>, cela donne :

```
dd@laptop:~/programmation/php/cli$php  -r
'for($i=1;$i<=3;$i++)print "cmd $i\n";'
cmd 1
cmd 2
cmd 3
```

Les exemples utilisés ici sont extrêmement simples mais vous pouvez adapter chacun d'eux en ajoutant des fonctions ou en utilisant des extensions qui permettront certainement de satisfaire chacune de vos requêtes.
L'utilisation de PHP-CLI pour créer un script permettant de gérer une base de données en mode console ou pour réaliser un exploit est tout à fait envisageable, alors j'espère que tous ceux qui pensaient que PHP est un langage sans intérêt commencent à changer d'avis ;-)

Nous nous sommes contentés de décrire les fonctionnalités de base de PHP-CLI, mais il faut savoir qu'il est possible d'utiliser des fonctions du type readline pour améliorer l'interaction avec l'utilisateur. De plus, avec les fonctions PEAR::Console_Getopt, vous pourrez parser facilement les différents paramètres passés à vos scripts.

Conclusion

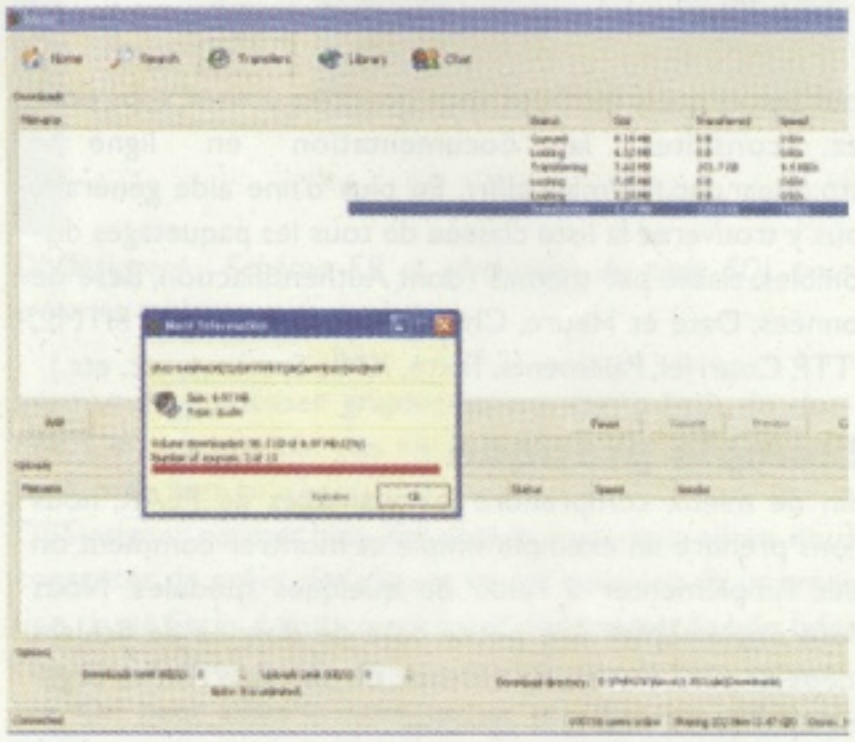
Voilà qui achève cette introduction à PHP-CLI. Nous avons vu à quel point il est simple d'utiliser PHP en environnement shell. Vous êtes maintenant capable d'exporter vos applications PHP en dehors du Web et ce, en toute simplicité alors... À vos claviers ;-)

Delete

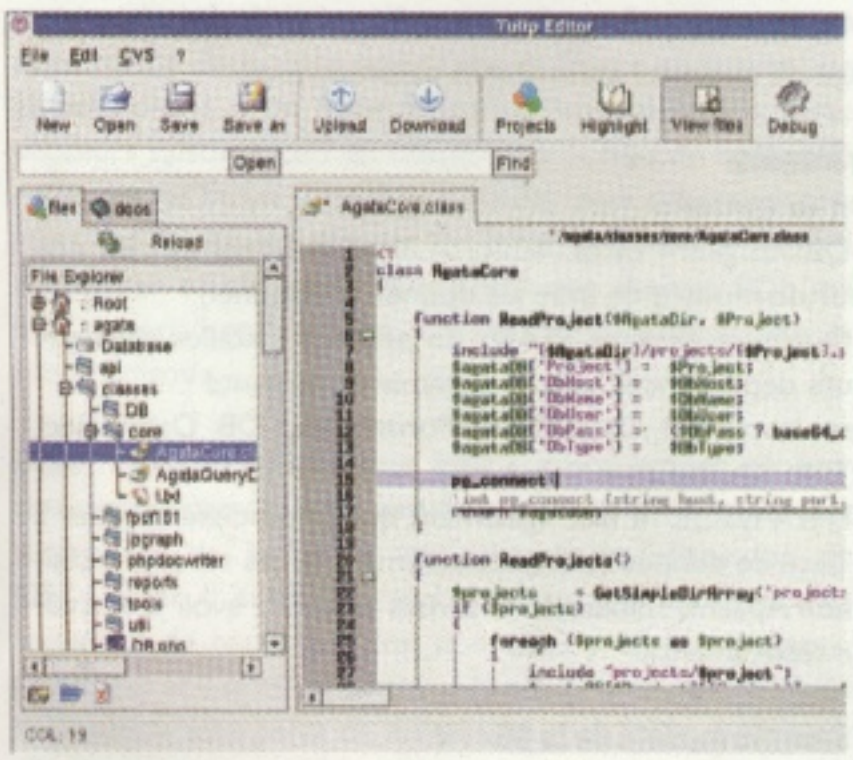
```
demo@evil:/tmp$ chmod a-w users.list
demo@evil:/tmp$ php add_user.php

Warning: fopen(users.list): failed to open stream: Permission denied in /tmp/add
_user.php on line 2
Nom: Une
Prenom: Demo

Warning: fwrite(): supplied argument is not a valid stream resource in /tmp/add_
user.php on line 9
Une erreur s'est produite !!!

demo@evil:/tmp$
```

Coder des GUI en php

PHP-GTK est une extension pour le langage PHP qui fournit une interface orientée objet avec la bibliothèque GTK+. Elle permet donc de réaliser des applications graphiques tournant sur tous les systèmes avec lesquels PHP et GTK+ sont compatibles.
La manière la plus simple d'installer tout ce qui est nécessaire est d'utiliser le framework Gnope (http://www.gnope.org/). Sur Windows, il vous suffit de lancer http://www.gnope.org/downloads/GnopeSetup-1.0.exe. Sur les autres systèmes, vous pouvez utiliser PEAR (voir la documentation du gnope.org et l'article sur PEAR de ce numéro).

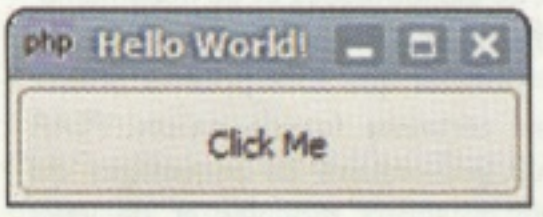


Nova, un client Gnutella écrit en PHP – novap2p.sf.net



Tulip, un éditeur PHP-GTK – agatareport.com/tulip

PHP, qui comme on l'a vu dans l'article précédent peut être considéré comme un langage de programmation à part entière, permet également de réaliser des applications graphiques, totalement indépendantes d'un quelconque serveur Web.



Exemple

Rien ne vaut un bon vieux hello world pour ce faire une idée.

```
<?php

function pressed_handler()
{
    echo "Ne me touchez pas !";
}

$window = new GtkWindow();
$window->set_title('Hello World!');
$window->connect_simple(
            'destroy',
            array('Gtk', 'main_quit'));

$button = new GtkButton('Click Me !');
$button->connect_simple(
            'clicked',
            'pressed_handler');

$window->add($button);

$window->show_all();
Gtk::main();

?>
```

On voit que comme avec n'importe quel autre langage, on commence par créer une fenêtre, à laquelle on ajoute ensuite des éléments. On peut ensuite connecter des fonction à des évennents, comme ici le handler au bouton.

Voir aussi :

- http://gtk.php.net - site officiel
- http://www.php-gtk.org - portail francophone
- http://gtk.org/api – documentation GTK de référence

Développement PH

Le but de ce dernier article est de vous donner l'envie de vous intéresser à quelques outils utiles en PHP, notamment à PEAR, une base impressionnante de code réutilisable. À l'aide d'un exemple pratique, nous allons montrer à quel point certains de ses modules peuvent simplifier la tâche du développeur.

Introduction à PEAR

PEAR (pour PHP Extension and Application Repository) est un dépôt de scripts, de bibliothèques et de documentation pour PHP. À la manière du CPAN de Perl, il s'agit d'en ensemble cohérent de modules que vous pourrez utiliser pour coder rapidement certaines fonctionnalités. PEAR comprend également un gestionnaire de paquetages qui simplifie l'installation de nouveaux modules et de leurs dépendances.

Pour pouvoir utiliser et installer des modules PEAR, il faut commencer par installer le gestionnaire. Pour cela, il suffit de télécharger le script disponible à l'adresse http://go-pear.org/ (faire un simple sauver sous 'go-pear.php') et de l'exécuter avec l'interpréteur en ligne de commande, comme expliqué dans l'article p. 48. Sur beaucoup de distributions Linux, soit PEAR est installé par défaut avec PHP, soit il suffit d'installer le paquetage php-pear ou équivalent. Une fois le gestionnaire installé, vous pouvez normalement utiliser la commande pear (vérifier votre PATH si ce n'est pas le cas). Les commandes principales sont les suivantes.

Installation et téléchargement d'un module :
pear install 'nom du package'

Par exemple :
```
# pear install MP3_ID
downloading MP3_Id-1.2.0.tgz ...
Starting to download MP3_Id-1.2.0.tgz
(8,833 bytes)
.....done: 8,833 bytes
install ok: channel://pear.php.net/MP3_Id-
1.2.0
#
```

Installation d'un module compatible (téléchargé directement sur pear.php.net, par exemple) :
pear install exemple.tgz

Supprimer un module :
pear uninstall 'nom du package'

Recherche d'un module :
```
$ pear search motclé
```

Liste des modules installés :
```
$ pear list
```

Liste des modules disponibles :
```
$ pear remote-list
```

Pour savoir quels modules vous pourriez utiliser, vous pouvez consulter la documentation en ligne : http://pear.php.net/manual/fr/. En plus d'une aide générale, vous y trouverez la liste classée de tous les paquetages disponibles, classé par thèmes (dont Authentifiaction, Base de données, Date et Heure, Chiffrement, Gtk, Images, HTML, HTTP, Courriel, Paiements, Texte, XML, Système, etc. etc.).

Exemple pratique

Afin de mieux comprendre les avantages de PEAR, nous allons prendre un exemple simple et montrer comment on peut l'implémenter à l'aide de quelques modules. Nous allons implémenter une petite base de donnée de fichiers vidéos disponibles sur les réseaux de p2p, que l'on va organiser par série, saison et épisode.

Nous allons montrer comment utiliser quelques classes PEAR permettant d'accéder à une base de donnée MySQL à l'aide d'une interface orientée objet, et profitant au passage de quelques mécanismes automatiques.

Prérequis

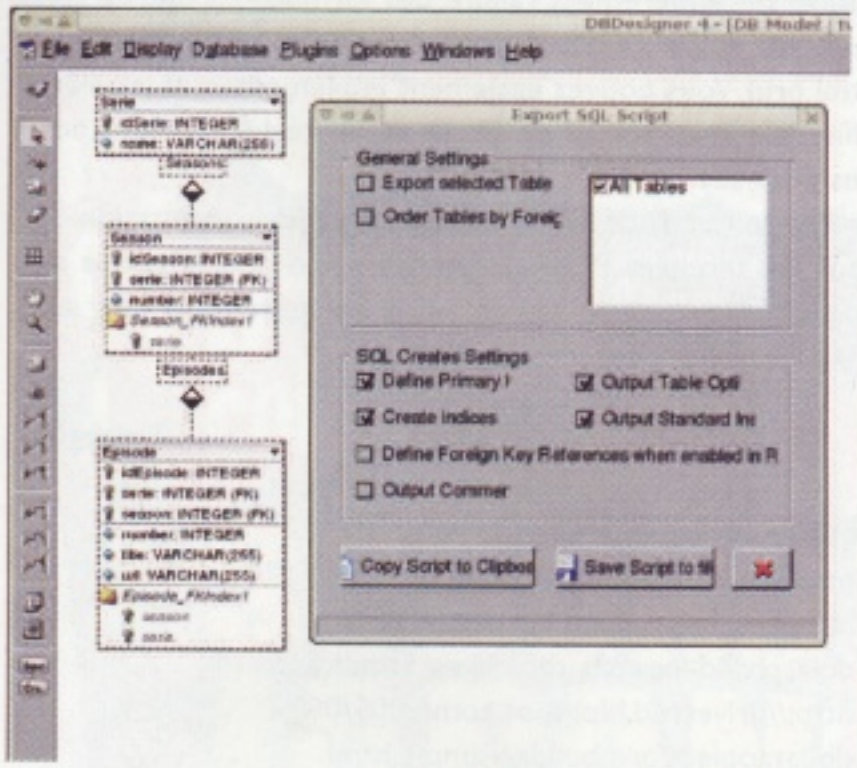
On va utiliser :

- DBDesigner4 (http://fabforce.net/dbdesigner4/) : un éditeur de modèle de base de donnée (optionnel)
- Quelques package PEAR : on peut les installer, ainsi que leurs dépendances, avec la commande suivante :
pear install DB_DataObject_FormBuilder DB_DataObject HTML_QuickForm
- PHP-MySQL : il faut également que php puisse accéder à la base de donnée (c'est normalement le cas sur une installation Apache standard, mais vous pourriez avoir à l'installer séparément)

1. Créer le modèle de la DB

Je vous conseille de profiter de cette occasion pour essayer DBDesigner4, un logiciel libre pour Linux et Windows qui

P/MySQL accéléré



DBDesigner4 : Schéma ER et génération du code SQL pour créer les tables

permet de modéliser graphiquement votre base de donnée. Ce n'est de loin pas nécessaire pour un projet aussi petit, mais bien pratique.
DBDesigner permet bien des choses, mais nous allons nous contenter de créer des classes et des relations. Pour créer une classe Serie, il faut commencer par trouver le bon bouton, puis cliquer sur le plan de travail pour la faire apparaître. On peut alors la nommer, et modifier ses différents attributs. Comme suggéré par le logiciel, on crée un champs idSerie ainsi qu'un champs pour le nom. On crée ensuite de la même manière une classe Season, avec un numéro. On peut alors relier l'une et l'autre en créant une associations 1..n de Serie à Season (une série comporte plusieurs saisons), ceci en cliquant sur le bouton approprié (regarder les bulles d'aide) puis sur les deux classes successivement. En double-cliquant sur la relation crée, on peut la renommer et choisir le nom exact des champs SQL qui vont matérialiser la relation. On fait de même pour relier la classe Episode.

Après avoir vérifié que tout était correct, on peu générer automatiquement la structure de la base de données. Le plus simple est de se servir des plugins de connexion de DBDesigner. Il suffit de créer la base ainsi qu'un couple login/mot de passe que l'on donne au programme (menu Database) et il se charge de tout (y compris de mettre à jour la structure sans perte de données si vous l'avez changée).
On peut aussi simplement l'exporter en langage SQL (menu File).

2. Générer les classes PHP

Une fois la base mise en place, on peut commencer à s'intéresser à ces fameux modules PEAR. Nous allons utiliser DB_DataObject, pour l'interface objet à la base, ainsi que DB_DataObject_FormBuilder pour s'éviter la peine de créer à la main des formulaires.
DB_DataObject utilise deux ou trois fichier de configuration central qui décrivent la base. Dans le répertoire de votre projet, il faut commencer par créer un fichier contenant les identifiants et des chemins. Par exemple :

```
; /tmp/peardemo/db_dataobject.ini
[DB_DataObject]
database = mysql://prog7:xx@localhost/prog7
schema_location = /tmp/peardemo/www/config
class_location = /tmp/peardemo/www/lib
extends = DB_DataObject
extends_location = DB/DataObject.php
```

À partir de ces information, le module est capable d'identifier les différents objets de votre base.
Cherchez le script createTables.php que PEAR a installé précédemment et lancez-le avec le chemin du fichier que vous venez de créer.

```
$ php /usr/share/php/DB/DataObject/createTables.php \
    /tmp/project/db_dataobject.ini
```

Cette commande aura pour effet de créer dans le répertoire lib un fichier php pour chaque table (Serie.php, Season.php, Episode.php), définissant une classe.
Vous constaterez également la création d'un fichier prog7.ini (du nom de la base de donnée), qui contient une définition codée du type de chaque colonne. Vous pouvez le modifier afin de préciser certaines nuances (dates, etc.). Voir la doc de référence.

On va dans la foulée créer manuellement un fichier prog7.links.ini, dans le même répertoire, où l'on précise les relations entre les tables (ce sera utile pour pouvoir afficher automatiquement cette relation dans le formulaire) :

```
; /tmp/peardemo/config/prog7.links.ini
[Episode]
season = Season:idSeason
serie = Serie:idSerie
[Season]
serie = Serie:idSerie
```

Vous constatez que jusqu'à présent, nous n'avons pas écrit une ligne de PHP. Pourtant tout est en place.

3. Interactions utilisateur

On peut maintenant s'atteler au code minimal responsable de traduire les actions de l'utilisateur en opérations sur la base de données. Voici le code principal d'un index.php :

```
<? #/tmp/peardemo/project/index.php
require_once "lib/Serie.php";
require_once "lib/Season.php";
require_once "lib/Episode.php";
// Pour éviter un warning dans les logs
define( "DB_DATAOBJECT_NO_OVERLOAD", TRUE );
// Pour utiliser le fichier de config vu précédement
$config = parse_ini_file(
   '/tmp/peardemo/db_dataobject.ini',TRUE);
foreach($config as $class=>$values) {
 $opts = &PEAR::getStaticProperty($class, 'options');
 $opts = $values;
}
// En-tête html
echo '<h1>Vos épisodes</h1><table>';
echo "<thead><tr><th>Série<th>Saison".
     "<th>Épisode<th></thead>";
// On interroge la base de donnée en quel-
   ques lignes claires
$episode = new Episode();
$episode->orderBy("number", "title");
$episode->find();
// boucle principale
while ($episode->fetch()) {
    $serie = $episode->getLink("serie");
    $season = $episode->getLink("season");
    echo "<tr>";
    echo "<td>".$serie->name."</td>";
    echo "<td>".$season->number."</td>";
    echo '<td>'.$episode->number.". "
        .'<a href="'.$episode->url.'">'.
        $episode->title."</a></td>";
    echo "<td>".'<a href="edit.php?editid='
        .$episode->idEpisode.'">edit</a>';
}
echo "</table>\n";
echo '<a href="edit.php">new</a>';
?>
```

Quant à edit.php, il peut quasiment se résumer à ceci :

```
// Initialisation du formulaire
$what =& new Episode();
$fg =&DB_DataObject_FormBuilder::create($what);
$form =& $fg->getForm();
// Est-ce que tous les champs sont ok ?
if ($form->validate()) {
 $form->process(array(&$fg,'processForm'),
                false);
 $form->freeze();
}
$form->display();
```

Ce simple code gère l'affichage des champs de l'objet (nouveau ou mis à jour) ainsi que la modification de la base de donnée après soumission et validation.

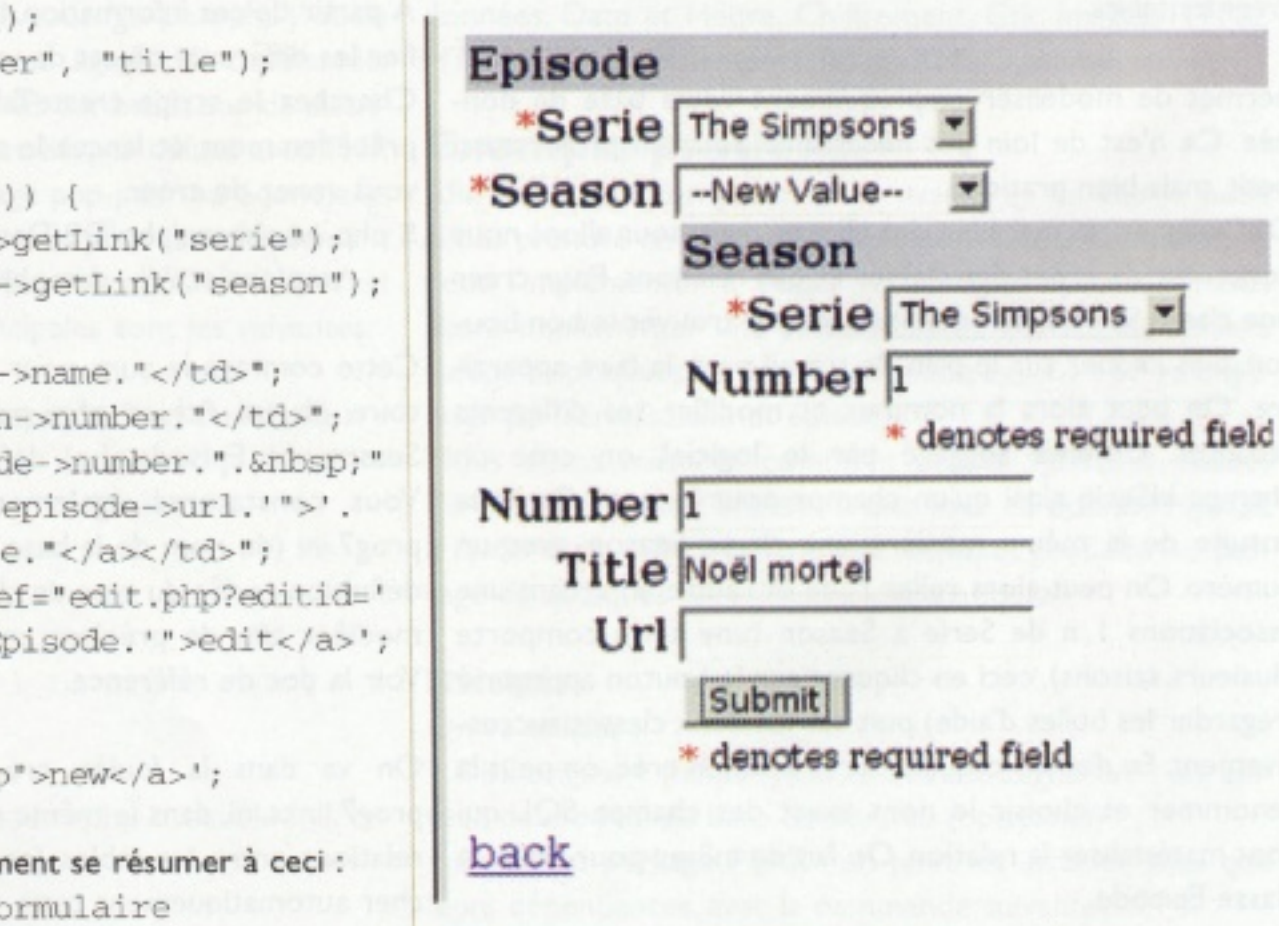
Beaucoup de choses sont paramétrables. Vous pouvez personnaliser entièrement l'allure des formulaires, que ce soit les noms et l'allure des champs, des classes css ou même du html brut. Vous pouvez également implémenter de manière efficace toutes sortes de filtres et de règles de validation des données.

Prenez le temps de vous balader dans la documentation de tous ces modules PEAR. Et prenez également le temps de consulter leurs code source : vous en apprendrez long sur certains aspects avancés de PHP.

dvrasp

Références :

- http://pear.php.net/manual
- http://pear.reversefold.com/dokuwiki/doku.php?id=pear:db_dataobject_formbuilder
- http://driverred.blogspot.com/2005/06/dbdataobjectformbuilder-simple.html

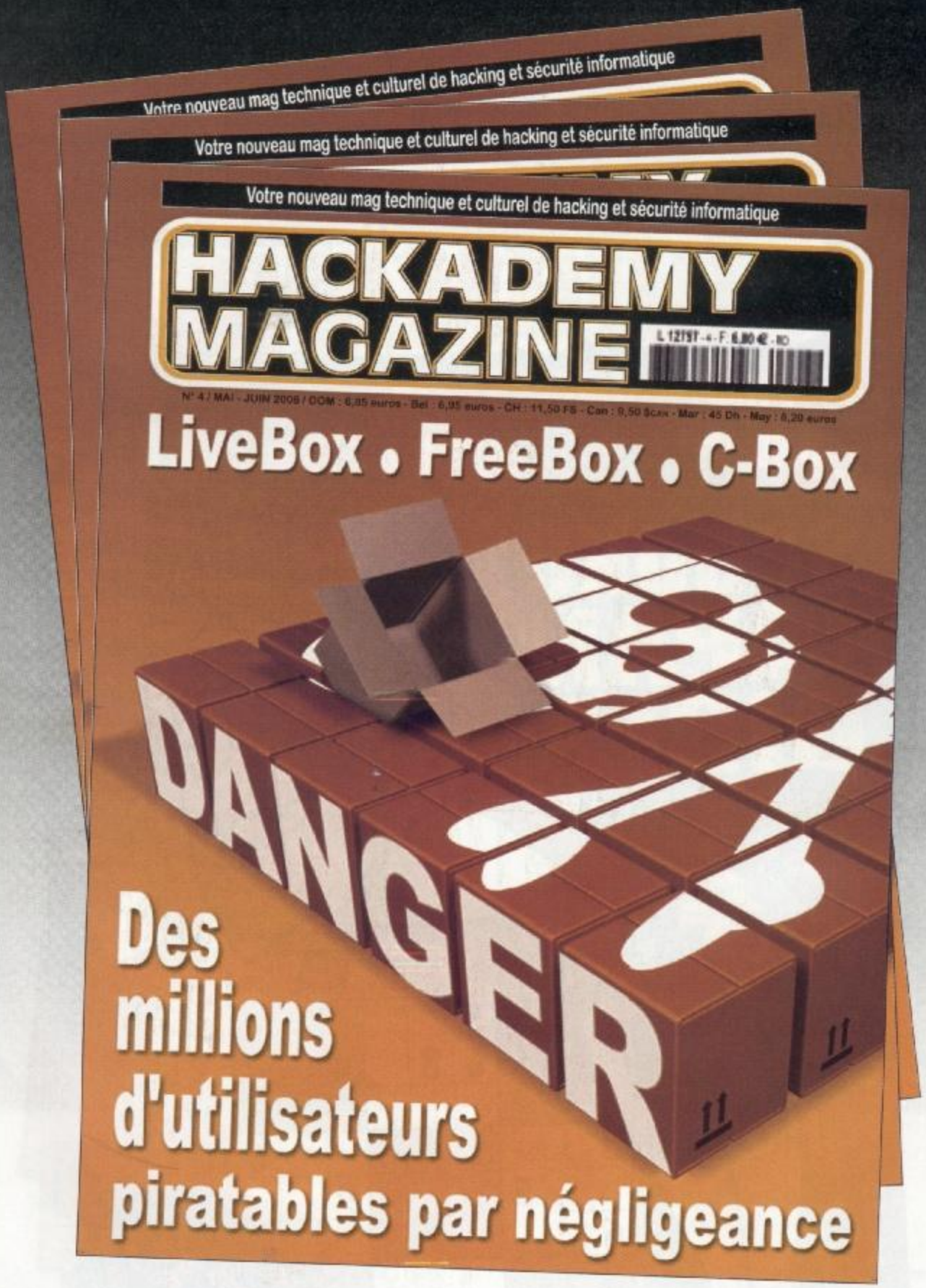


FormBuilder gère automatiquement les sous-éléments

Remarques :

- La distribution binaire de DBDesigner4 ne fonctionne pas sans modification sur certaines distribution trop à jour (ma Debian par exemple) ; poser une question sur http://forum.thehackademy.net si vous rencontrez des problèmes.
- Le code complet de l'exemple est également disponible ; demandez sur le forum.

En vente en kiosque



En vente en kiosque