



Le plus grand magazine sur PHP au monde

FIREBUG FIREPHP DESIGN PATTERNS JAVA FILTERPHP PHP 5.3

phpsolutions 3/2010 (39)

phpsolutions

Nouvelles technologies et solutions pour les développeurs PHP

PHP N° 3/2010 (39) ISSN 1731-4593 Prix 7,50 EUR CD offert France Metro : 7,50 EUR DOM : 8,80 EUR MAR : 80 MAD TOM/S : 990 XPF

PHP ET SÉCURITÉ

phpsolutions



SUR LE CD

EN EXCLUSIVITÉ
BUSINESS ICON SET

PHP 5.3
AMÉLIORATIONS ET NOUVEAUTÉS

GLASSFISH
COUPLER LA PUISSANCE
DE JAVA EE ET PHP

TÉLÉCHARGER UN FICHIER
DEPUIS UN SCRIPT PHP
ARCHITECTURE CLIENT/SERVEUR

EXPRESSIONS RÉGULIÈRES
POSIX ET PCRE

PRATIQUE

DESIGN PATTERNS
UTILISATION DU PATTERN OBSERVATEUR

FICHE TECHNIQUE

GESTION DU MULTILINGUISME
RÉALISEZ VOS PROJETS WEB

L 14389 - 39 - F: 7,50 € - RD





Développeurs PHP/MySQL, Venez nous rejoindre !

Travaillez pour le plus grand site
de ventes d'objets de collection !

Premières sélections très prochaines !

Quelques chiffres...

- 450 000 utilisateurs
- 600 000 objets vendus par mois
- 27 millions d'objets en vente

Compétences et Qualités

Vous maîtrisez le développement :

- PHP/MySQL/JavaScript/DHTML/CSS/Ajax.

Sont des atouts :

- Environnement Linux (Administration système) ;
- Design et ergonomie ;
- Optimisation MySQL ;
- Sécurité des applications Internet ;
- Travail en équipe.

Description du poste

A Enghien (Belgique)

Contrat à temps plein, durée indéterminée.

- Développement PHP de pages et services du site Delcampe ;
- Design et ergonomie des interfaces ;
- Gestion et optimisation des bases de données ;
- Sécurisation logicielle des services Delcampe ;
- Participation active aux idées et analyses de développement ;
- ...



Pour toute information, contactez :

Christophe Gesché, Directeur Technique
christophe@delcampe.com

Envoyez votre candidature à :

Evelyne Lorand, Directrice RH
evelyne@delcampe.com



Rejoignez notre groupe Facebook :
"Delcampe.net recrute !"

<http://www.facebook.com/group.php?gid=21588322484>



Sébastien Delcampe
Fondateur & Senior PHP Developer



ICON STUDIO

a picture is worth a thousand words

Icons are a crucial part of any product, whether that's software or a website. Just a glance at an icon gives users a clear idea of the application and the company that develops it. High-quality icons put you far ahead of your competitors on the software market.



CUSTOM DESIGN www.perfect-icons.com

It's a good decision to order graphics for your product from professional designers. In this case, the product will have its uniqueness, which users like so much. Besides, you can discuss all aspects with the designer, which will result in high-quality individual icons.



STOCK ICONS www.777icons.com

Speed up product development and save your money by choosing to use ready-made stock icons instead of ordering custom graphics. 777icons.com offers complete sets and a la carte icons matching the most demanding tastes.



ICON EDITOR www.aha-soft.com

To go further in your creativity, you can create icons using special software - graphic editors. With IconLover, you can create simple images and edit them by adding effects that make them unique. IconLover can be also used for creating icons out of your photos. It is also suitable for correcting icons you already have.



TABLE DES MATIÈRES

VARIA

6 Actualités

Actualités du monde du développement.

8 Description du CD

Présentation du contenu du CD joint au magazine.

10 Interview de Fabrice Sourdonnier

Fabrice Sourdonnier, co-gérant de Cognix Systems.

OUTILS

12 Tester et nettoyer les données grâce à FilterPHP

Jérémy Rafflin

La sécurité dans PHP est un point qui est très important. Avant d'effectuer la requête vous devez tester l'id pour vérifier qu'il s'agit bien d'un nombre pour éviter de générer une erreur. N'oubliez pas de tester et/ou nettoyer vos données afin d'éviter les erreurs lors de l'exécution de vos scripts et d'éviter les injections de codes malveillants. Dans cet article, nous allons voir que la classe FilterPHP est plutôt bien adaptée aux problèmes courants.

PROJETS

22 La gestion du multilinguisme dans le développement web

Christophe Cadic

Le multilinguisme est un sujet très présent dans le développement web. Or, cette notion peut s'avérer bien plus complexe qu'il n'y paraît. Vous verrez quels cas de figure peuvent être rencontrés et quels sont les moyens d'y répondre.

28 Découvrez SimpleXml

Nicolas Turneau

Le XML est un langage informatique dit de balisage générique. Il est principalement utilisé pour structurer des données. Voyons ensemble comment utiliser la librairie SimpleXml afin de créer et lire des documents XML assez simplement.



DOSSIER

32 Démarrez avec FireBug et FirePHP

Christophe Villeneuve

Être assisté pendant les développements permet de gagner des heures de corrections intensives. De nombreux outils existent permettant d'accélérer l'écriture de vos projets PHP. C'est le cas de FirePHP qui vous est présenté dans ce numéro.

37 La sécurité de base d'un serveur

Eric Vincent

Tout webmaster doit avoir un minimum de connaissances, un minimum de compétences au niveau de la sécurité. Car il existe seulement deux sortes de webmasters : ceux qui ont déjà été victime d'un pirate et ceux qui n'ont jamais eu de soucis mais qui en auront forcément un jour. Les conséquences étant lourdes, il est nécessaire et obligatoire d'y consacrer du temps avant. Après, il est trop tard. Alors, concevons une configuration minimale d'un serveur.

PRATIQUE

42 Observez vos objets et réagissez à leurs événements

Mathieu Gautheret

Le design pattern Observateur est l'un des patterns les plus utilisés car il répond à une problématique souvent rencontrée dans le développement orienté objet : comment formé des objets du changement d'état d'un autre objet tout en respectant le principe d'un couplage faible ? Observateur est donc classé dans la catégorie des patterns comportementaux puisqu'il permet d'organiser des objets pour qu'ils collaborent facilement entre eux. Découvrez les avantages et les inconvénients du pattern Observateur.



50 PHP et les expressions régulières

Dony Chiquel

Les expressions régulières ont été grâce à leur puissance adoptées par de nombreux langages de programmation et ont vu leur champ d'utilisation s'étendre des fichiers *htaccess* aux feuilles de style CSS. Voyons comment tirer profit des nombreux avantages qu'offrent les expressions régulières.

FICHE TECHNIQUE

56 Les nouveautés de PHP 5.3

Fabrice Facorat



PHP 5.3 marque une étape importante pour le basculement du langage dans le monde des langages dit professionnels en apportant de nombreuses nouveautés notamment en ce qui concerne la programmation orientée objet et le déploiement d'applications PHP. Dans cet article nous verrons la migration de PHP 5.2 vers la version 5.3.

60 Coupler la puissance de Java EE et PHP grâce à GlassFish

Jérôme Lafosse

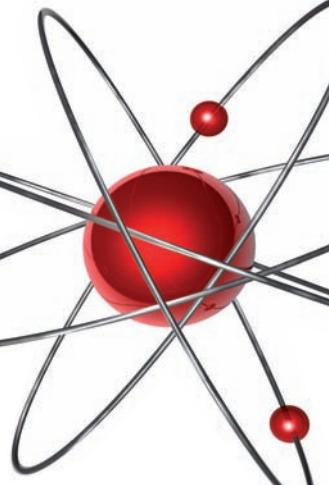
La tendance actuelle est au développement d'applications Internet avec la plate-forme Java EE 6 ou PHP. Mais pourquoi ne pas coupler la puissance de ces deux langages de programmation ? Une telle approche permet d'utiliser un site déjà existant et de le migrer facilement vers un des deux langages, d'utiliser les fonctions adaptées et optimisées de chaque langage ou mieux, de mélanger les deux langages afin d'obtenir un système Java EE 6 / PHP.

POUR LES DÉBUTANTS

68 Télécharger un fichier depuis un script PHP

Magali Contensin, Cécile Odero

Cet article présente trois méthodes alternatives de téléchargement de fichiers à partir d'un script PHP. La première méthode utilise des fonctions PHP natives, la seconde emploie la bibliothèque gratuite et open source CURL, la dernière établit une communication par sockets.



78 PHP et Delphi

Faure Yann

Delphi est un langage de programmation orientée objet (POO) créé et édité par Borland en 1995. Cet article présente les différentes possibilités pour faire interagir votre application locale en Delphi avec PHP et ainsi permettre la communication entre logiciels et serveurs simplement.



PHP 5.2.13

Une nouvelle version de la branche PHP 5.2 vient de sortir. Cette nouvelle version est une version mineure, mais très importante. Elle corrige de nombreux bugs dont certains liés à la sécurité et c'est pourquoi il est encouragé de passer à cette version pour l'ensemble des utilisateurs.

http://www.php.net/releases/5_2_13.php

PHP TV émission 5

PHP TV la web TV, a publié la nouvelle version de son site internet. Une refonte avec un nouveau logo et beaucoup d'améliorations.... Il vous reste juste à le consulter <http://www.phptv.fr>.

Migration PHP 5.2 vers PHP 5.3

Avec le déploiement de PHP 5.3, de nombreux ajouts et de modifications importantes à PHP sont apparus. Pour faciliter le passage de PHP 5.2 à PHP 5.3, Stas Malyshov propose un petit script.

Ce script concerne les points les plus répandus, et répond à une grande partie des évolutions.

<http://github.com/smalyshев/migrate>

Bluefish 2.0

Bluefish est un IDE (éditeur de texte). Cette nouvelle version est orientée développement Web en 17 langues. Son souhait est d'être léger, rapide, libre mais aussi ouvert. Grâce à cette ouverture, de nombreux langages Open Source sont supportés dont PHP. Une large gamme de fonctionnalités est disponible comme la coloration syntaxique, une interface multiple documents, intégration de programmes externes, synchronisation, barre d'outils...

<http://bluefish.openoffice.nl>

Zend Server 5.0

Zend Server 5.0 est la nouvelle version que Zend propose avec de nombreuses évolutions, comme une amélioration des performances et un résolution plus rapide des problèmes. Par ailleurs, cette nouvelle version supporte PHP 5.3

<http://zend.com/fr>

EGGcrm 1.50

EGGcrm est un logiciel de gestion relation en PHP / MySQL. Il permet de gérer des fichiers clients, suivre des affaires, prospection et éditer les documents. Une gestion e-mailing est aussi proposée et compatible PHP 5.3 Les nouvelles fonctionnalités proposées sont une gestion de stocks améliorée, une organisation du menu avec des raccourcis, un éditeur Wysiwyg...
<http://www.eggcrm.net>

Memq

MEMQ est basé sur l'utilisation de Memcache et de PHP. Cette nouvelle approche va vous permettre de mettre en place un système de file d'attente que vous pouvez utiliser dans vos propres applications.
<http://abhinavsingh.com/blog/2010/02/memq-fast-queue-implementation-using-memcached-and-php-only/>

HipHop

HipHop est le nom du nouveau projet de Facebook, réalisé en PHP. Ce nouveau projet est un optimiseur pour améliorer les performances de son réseau social. Depuis 2 ans, le projet a été commencé pour arriver au résultat présenté aujourd'hui. Il transforme le code source de PHP dans un langage C++ optimisé et compiler en utilisant G++. Le résultat obtenu permet de soulager l'affichage des pages sur le serveur et ainsi de gagner 50 % de consommation.

Bien entendu, si les serveurs sont moins sollicités, la consommation des kWh par



an va diminuer car Facebook aura besoin de moins de serveurs : <http://developers.facebook.com/news.php?blog=1&story=358>. Mais lorsqu'un projet majeur apparaît, il n'est jamais seul et c'est le cas. Facebook exploite XHP, une autre façon d'écrire du PHP. XHP est par conséquent une extension utilisée par Facebook qui va augmenter la syntaxe de PHP et permettre d'écrire des fragments HTML dans un même script comme ceci :

```
<?php  
if ($_POST['name']) {  
?>  
    <span>Hello, <?= $_  
POST['name']?>.</span>  
<?php  
} else {  
?>  
    <form method="post">  
        What is your name?<br>  
        <input type="text" name="name">  
        <input type="submit">  
    </form>  
<?php  
}  
?>
```

PHP Firewall

PHP Firewall est une API qui va vous aider à contrôler un certain nombre de points de sécurité connus, touchant entre autres la détection des PROXYs. Il s'agit d'un script à insérer dans votre site internet en PHP. Il va avoir le rôle de parefeu. Bien entendu, il ne faut pas perdre de vue les bonnes pratiques de programmation, car même si vous utilisez ce script, il faut penser qu'il est un complément de votre programmation sécurisée.

Il propose un système de logs et la possibilité d'être alerté par email en cas d'opérations non souhaitées. PHP Firewall repère et détecte les points suivants :

- Les attaques de XSS, les injections SQL, les inclusions de fichiers.
- Une protection contre les robots dangereux.
- Une protection contre les attaques de requêtes à distance.
- Une détection de certains types de vers comme Santy et même pouvant venir de votre serveur d'hébergement.

Bien entendu, il s'occupera du nettoyage des cookies, des variables POST ou GET. A cette occasion, il pourra mémoriser les IP qui essayeront de vous attaquer par l'intermédiaire de Spams ou sous la forme de différentes attaques.

<http://fr.php-firewall.info/>



École Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Télécommunications

L'ENSEEIHT est une école d'ingénieurs centenaire qui délivre 5 diplômes d'ingénieurs dans les domaines du génie électrique, de l'électronique, de l'informatique, de l'hydraulique et des télécommunications, proposant une formation scientifique et technique très fortement reconnue et de renommée nationale ou internationale.

Les différentes évolutions scientifiques de l'école l'ont amenée à anticiper régulièrement les domaines et les besoins du monde industriel. C'est ainsi que, dès le début du 20^{ème} siècle, l'école a répondu aux besoins d'énergie hydro-électrique. Très rapidement, dans les années 50, elle s'est positionnée dans les domaines de l'électronique, au début des années 60, elle a été la première école française à se positionner dans le domaine de l'informatique et tout récemment, elle s'est positionnée dans les domaines des télécommunications et réseaux.

L'école s'ouvre très largement à l'international avec 30 % d'étudiants étrangers en

formation d'ingénieurs et une obligation de mobilité internationale pour tous nos étudiants élèves ingénieurs. Classée dans les 20 premières écoles françaises (220 écoles d'ingénieurs en France), l'ENSEEIHT est un acteur important de la formation d'ingénieurs en particulier dans les domaines relevant des technologies de l'information et de la communication. Dans ces domaines, l'école a su anticiper les besoins industriels en intégrant dans ses formations les nouvelles technologies avant même qu'elles deviennent incontournables ; tel a été le cas dans les années 70 avec le génie logiciel, dans les années 80 avec les technologies objet ou aujourd'hui avec les technologies web 2.0 ou 2.5.

L'école aujourd'hui délivre 380 diplômes d'ingénieurs par an (dont 200 dans les domaines TIC) et 80 doctorats par an. L'ENSEEIHT propose également les formations suivantes : le *Master spécialisé en informatique* et une formation par l'apprentissage en *informatique et réseaux*. Pour en savoir plus www.enseeht.fr.

Construire un site pour événement

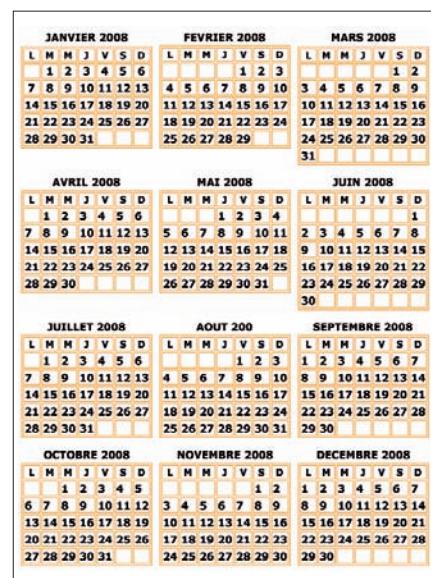
Construire un site web pour un événement n'est pas destiné qu'aux développeurs ou développeuses car lorsque vous désirez organiser un événement, quelque soit le type, et la cible que vous souhaitez toucher, de nombreuses opérations sont nécessaires.

Anna Filina de PHP Quebec donne dix points que toutes équipes organisatrices d'événements doivent connaître à la perfection.

Les différentes étapes à réfléchir sont :

- Logo et design de l'événement.
- Appel à conférenciers.
- Choisir les sélections.
- Vente des billets d'entrées.
- Outils de suivi.
- Ventes flash.
- Planification.
- Répartition des postes avec l'équipe organisatrice.

- Buzz et communication.
- Inscription de l'événement sur des sites spécifiques.



10 étapes pour memcached server

Pour améliorer vos applications PHP, vous devez souvent utiliser un cache. Il existe différentes extensions et différents paquets. Le site *Web Developer juice* montre sous Linux comment installer et configurer Memcached Server en 10 étapes.

<http://www.webdeveloperjuice.com/2010/01/25/10-baby-steps-to-install-memcached-server-and-access-it-with-php/>

Slow show

Slow Show est un outil Open Source qui va vous permettre de prendre différentes mesures de performances de vos sites internet. Ces différentes mesures pourront vous aider à analyser l'ensemble de vos sites sur une période donnée. Il capture les résultats en utilisant *YSlow* et les classe en rapidité sous forme graphique.

<http://groups.google.com/group/showslow>

Drupal 7

Drupal est un CMS écrit en PHP. Sa nouvelle version est maintenant disponible. Elle présente une interface utilisateur repensée. Quelques nouveaux modules très utilisés dans les versions actuelles ont été intégrés en standard dans ce CMS comme les modules CCK et *ImageField*. Maintenant, cette nouvelle version supporte l'ensemble des bases de données en intégrant une couche d'abstraction avec PDO. Par ailleurs, la gestion des fichiers est maintenant basée sur *SimpleTest*. <http://www.drupal.org>

PHPMD – PHP Mess Detector

PHPMD est une API inspirée de l'outil Java PMD. Cette API va essayer de repérer les problèmes éventuels d'une page PHP. Les problèmes détectés sont : les bugs éventuels, le code non optimisé, la détection d'expressions trop compliquées, des parties inutiles comme des paramètres, des propriétés...

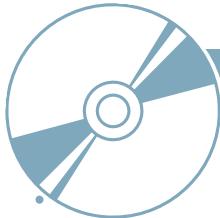
<http://phpmd.org>

HMVC

Le modèle MVC est très utilisé dans de nombreux outils, Frameworks... Ce modèle peut encore évoluer. Si vous modifiez légèrement le modèle existant en appliquant une option hiérarchisée, vous obtenez le modèle HMVC (*Hierarchical-Model-View-Controller pattern*). Sam de Freyssinet publie la mise en place de HMVC. Il nous présente les avantages à utiliser cette technique. Par ailleurs, vous verrez, sous la forme de scripts, différents concepts de base pour les insérer dans vos applications. <http://techportal.ibuildings.com/2010/02/22/scaling-web-applications-with-hmvc/>

Maarch entreprise

Maarch Entreprise est un application écrite en PHP 5 sous licence GPL 3. Elle est basée sur le Framework du même nom. Cette API est destinée pour une utilisation métier et plus exactement pour l'archivage électronique (SAE), la gestion des archives physiques et la gestion documentaire (GED). L'ensemble des possibilités qui vous sont offertes répondent à la norme NF Z42-013. <http://www.maarch.org>



Description de CD

Business Icon Set

Nous vous proposons la version complète de Business Icon Set. Ce logiciel vous permettra de réaliser des icônes de toute sorte comme le travail, la finance et les chiffres et sites Web de manière plus séduisante. Ce *Set d'icônes* est professionnellement dessiné pour partager des couleurs, le style et la gamma.

Business icônes existent dans tous les formats et tailles courantes, ainsi vous trouverez certainement celui correspondant le mieux à vos besoins. Les icônes sont réalisées dans les tailles 16x16, 24x24, 32x32, 48x48 et aussi 256x256. Les icônes sont disponibles en deux variantes de couleur : 256 couleurs et *True Color* avec semi-transparence. Ils ont aussi plusieurs formats de fichiers, comme ICO, PNG, GIF et BMP. Il vous permet de pimenter vos logiciels ou sites web avec des icônes plus *fun*.

Business Icon Set dispose d'une collection d'icônes libres pour une utilisation commerciale et personnelle, y compris les logiciels, sites Web, blogs, et présentations. Les icônes sont divisées en catégories telles que Business, Finances, Transports, l'argent, l'informatique et rapports. Les icônes sont soigneusement créées pixel par pixel par la main d'un artiste professionnel. Elles disposent d'une palette de couleurs aux effets brillants, lisses et arrondis. De super qualité, les icônes aident un développeur à mettre une touche vraiment professionnelle à l' interface de son projet sans avoir à employer un concepteur ou passer des jours voire des semaines sur la conception d'icônes.



Vos produits web et logiciels seront plus modernes et attrayants avec Business Icon Set. L'ensemble inclut des icônes d'envoi ou financières tels que dollar, euro, du yen, la monnaie sac, de l'argent, guichet, caisse enregistreuse, paiement, tirelire, coffre-fort, diamant, la balance, banque, billets de banque, laisse de billets de banque, serviette, calculatrice, calendrier, carte de crédit, télécopieur, poignée de main, livres, bus, voiture, camion, chariot élévateur, *pick-up*, remorque, pelle, hélicoptère, cadis, à pièces, pièces de monnaie, monnaie fengshui, téléphone cellulaire, téléphone, porte ouverte, serrure, cadenas ouvert, clé , maison, réservoir, montre, cœur, avion, bateau, train, camion ou panneau, satellite, base de données, recherche, ingénieur, gestionnaires, options, paramètres, outils, tournevis, pinceau, stylo, crayon, propriétés de la souris, du globe, la Terre, boussole, fauteuil, microscope, caméscope, imprimante, serveur, ordinateur portable, e-mail, recyclage.



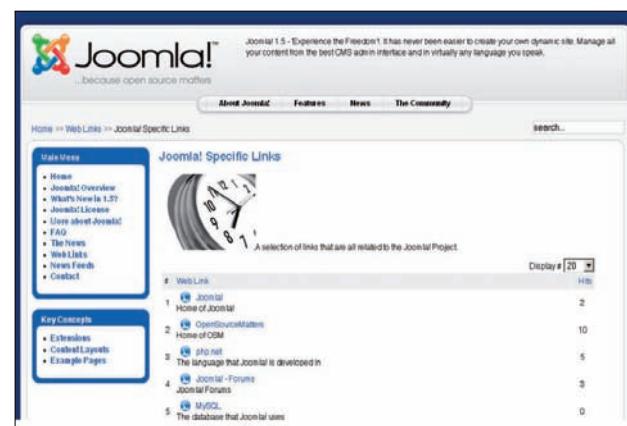
Pour plus d'informations sur le produit, nous vous invitons à visiter le site : <http://www.aha-soft.com/stock-icons/business-icons.htm>.



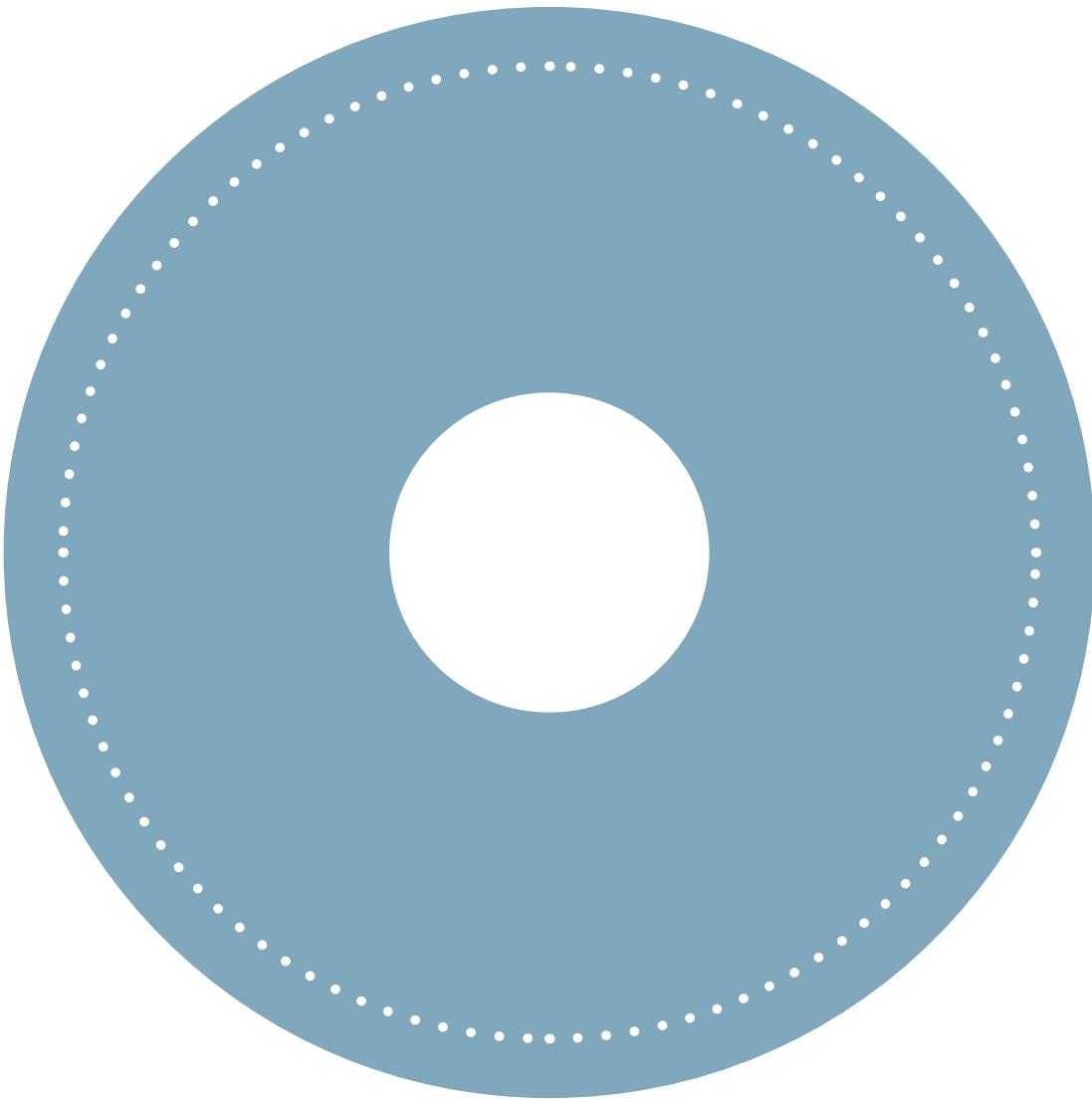
Joomla! 1.5

Joomla! est un système de gestion de contenu (en anglais CMS pour *Content Management System*) libre, open source et gratuit. Il est écrit en PHP et utilise une base de données MySQL. Joomla! inclut des fonctionnalités telles que des flux RSS, des news, une version imprimable des pages, des blogs, des sondages, des recherches. Joomla! est sous licence GNU GPL. Il est créé à partir du CMS Open Source Mambo en août 2005.

Bon apprentissage !



**S'il vous est impossible de lire un CD,
alors qu'il n'a pas de défaut apparent,
essayez de le lire dans un autre lecteur.**



**Pour tout problème concernant les CDs,
écrivez-nous à l'adresse :**

cd@phpsolmag.org



Interview de Fabrice Sourdonnier Co-gérant de Cognix Systems <http://www.cognix-systems.com>



WebGazelle® CMS 2.0

Le CMS génialement simple !

PHP Solutions : Il existe de très nombreux CMS, pourquoi un nouveau CMS ?

Fabrice Sourdonnier : Effectivement beaucoup de CMS, payants ou gratuits, sont présents sur le marché dont certains sont de très bonne qualité. Parmi les CMS OpenSource les plus connus et les plus utilisés, beaucoup sont à l'origine dédiés à la gestion de blogs ou de sites journalistiques. En 2003, avec la toute première version de WebGazelle® CMS, nous voulions développer un outil de mise à jour 100% dédié aux besoins d'un site Institutionnel de présentation d'entreprise. En partant de ce constat, nous avons conçu une ergonomie très largement simplifiée.

PS : Vous venez de lancer la version 2.0 de votre CMS, qu'apporte-t-elle de nouveau ?

FS : Au démarrage du projet WebGazelle® CMS 2.0, nous voulions offrir une vision rajeunie de notre CMS en le faisant bénéficier des atouts ergonomiques de l'AJAX. Il nous semblait naturel de pouvoir modifier l'arborescence d'un site ou l'emplacement de n'importe quel bloc de contenu par simple glisser/déposer. Nous voulions aussi concevoir un CMS très réactif pour qu'il soit un véritable outil de travail et de productivité pour les utilisateurs et les Webmasters.

PS : L'ergonomie semble être l'un des principaux avantages de votre CMS ?

FS : L'ergonomie n'est pas le seul avantage de WebGazelle® CMS 2.0, mais il est vrai que nous y avons apporté un soin tout particulier. En effet, nous avons essayé de comprendre comment était utilisé un CMS. Il faut dire que les besoins sont très variables, entre les entreprises qui mettent à jour leur site 1 ou 2 fois par an et celles qui le font quotidiennement, nous voulions un CMS aux interfaces à la fois intuitives pour les premiers et facteur de productivité pour les seconds.

PS : Quelles sont les autres atouts de WebGazelle® CMS 2.0 ?

FS : Il y en a beaucoup. Si je devais en citer 3, je parlerais tout d'abord de la gestion de vidéos, que nous voulions aussi simple que la mise en ligne de photos. A ce titre, WebGazelle® CMS 2.0 reconnaît automatiquement le format de la vidéo et la ré encode automatiquement au format Flash. Pour le référencement, nous voulions proposer plus qu'une simple solution optimisée. C'est la raison pour laquelle est né notre analyseur de référencement, un outil qui conseille l'utilisateur sur la façon d'optimiser les critères de visibilité naturelle de son site sur Internet. Pour finir, je parlerais de la vitesse d'affichage des sites, point sur lequel l'équipe a vraiment fait un très gros travail, avec de bonnes performances à la clé.

PS : Quelles technologies utilisez-vous ?

FS : Pour bâtir WebGazelle® CMS 2.0, nous avons construit notre propre Framework de développement, baptisé le FrameWork CS 2.0 (CS pour Cognix Systems) sur le socle des Frameworks élémentaires et robustes de Zend pour PHP et Jquery pour l'AJAX. Nous avons eu en fait une démarche inverse à celle



Visualiser la vidéo du CMS

<http://www.youtube.com/watch#!videos=MLsFGMXuTaw&v=FCjlfypue-w>

de beaucoup d'éditeurs qui font évoluer leur CMS vers une vraie plateforme de développement. En concevant WebGazelle® CMS 2.0 comme une brique du FrameWork CS 2.0, cela nous permet de concevoir d'autres applications et modules nativement compatibles les uns avec les autres. Les curieux trouveront un descriptif complet du Framework CS 2.0 sur notre site Internet.

PS : Quelles sont les autres applications et modules dont vous disposez ?

FS : On dispose aujourd'hui de 14 applications et modules qui ont pour vocation de répondre aux problématiques les plus

courantes des sites Internet, portails Web ou Intranet/extranet. Ce nombre va bien sûr augmenter en fonction des retours de nos utilisateurs. On dispose par exemple d'un module Newsletter pour les entreprises qui souhaitent gérer leurs campagnes de mailing. On peut également citer des modules comme notre médiathèque, notre annuaire, notre agenda partagé, ou encore une messagerie interne avec gestion de membres pour les portails communautaires.

PS : Sur quoi travaillez-vous en ce moment ?

FS : Nous continuons à beaucoup investir en R&D sur WebGazelle® CMS 2.0 et le FrameWork CS 2.0. Nous pensons que **nous sommes à l'aube d'une révolution** dans le domaine des logiciels Web dont les CMS font partie. Tout d'abord, les infrastructures deviennent de plus en plus performantes et cela ouvre des perspectives vraiment passionnantes, indispensables pour les calculs lourds comme le traitement des vidéos, l'indexation de contenu, ou le calcul des systèmes de cache. D'un autre côté, les navigateurs deviennent de plus en plus performants et le HTML5 va ouvrir de nouvelles possibilités.

PS : Cognix Systems met beaucoup l'accent sur le développement durable ?

FS : Derrière ce concept marketing actuel, nous pensons qu'il s'agit avant tout d'un état d'esprit qui vise à avoir une réflexion sur le moyen/long terme et non sur le court voir le très court terme. C'est une démarche que nous avons engagée en 2005. Depuis cette date, nous arrivons à avoir des résultats concrets, qui par exemple nous ont permis de réduire de moitié notre parc de serveurs, tout en améliorant la vitesse d'affichage de nos sites. A court terme, cela a représenté un investissement en R&D important, mais qui aujourd'hui permet de réduire l'empreinte écologique et le coût d'hébergement de nos sites.



Notre palette de modules



Cognix Systems, SSII et Web Agency depuis 2002

Expertise autour des technologies Web sur PHP

Développeurs PHP, venez nous rejoindre !

Intégrez notre équipe **jeune et dynamique** présente sur Rennes pour le développement d'applications Internet/intranet/extranet.

recrutement@cognix-systems.com

02 99 27 75 92

Plus d'informations sur
www.cognix-systems.com



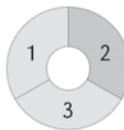
Tester et nettoyer les données grâce à FilterPHP

La sécurité dans PHP est un point qui est très important dans une application. Dans cet article, nous allons voir que la classe FilterPHP est plutôt bien adaptée aux problèmes courants.

Cet article explique :

- Comment nettoyer et tester des données grâce à FilterPHP.
- Les avantages et les inconvénients de FilterPHP.

Niveau de difficulté



Vous le savez sûrement déjà, PHP n'est pas un langage typé. C'est-à-dire que contrairement à d'autres langages, les données n'ont pas de type défini lors de leurs déclarations. C'est un des points qui fait que PHP est plus simple à apprendre qu'un autre langage comme Java ou C++. C'est également un handicap en ce qui concerne la sécurité, en effet, vous ne savez pas ce que contiennent vos variables. Supposons par exemple, qu'une requête SQL vous permette de sélectionner le nom d'un client en fonction de son id. Avant d'effectuer la requête vous devez tester l'id pour vérifier qu'il s'agit bien d'un nombre pour éviter de générer une erreur. Vous devez donc tester et/ou nettoyer vos données afin d'éviter les erreurs lors de l'exécution de vos scripts et d'éviter les injections de codes malveillants.

L'une des questions les plus importantes à se poser lorsque l'on traite la question de la sécurité d'une application PHP est : quelles sont les données de l'application à sécuriser ? En réalité toutes les données entrantes et sortantes sont à sécuriser. Les données entrantes doivent être sécurisées pour que votre application n'encourt aucun risque, cela paraît assez simple à comprendre. Par

Ce qu'il faut savoir :

- Connaître les bases de PHP(principalement les tableaux et les variables superglobales).
- Connaître les types de données existantes.

contre, les données sortantes sont trop souvent oubliées, en effet, il ne faut pas oublier de les sécuriser lorsqu'elles interagissent avec d'autres langages. Par exemple, un script PHP sécurise les données provenant d'un formulaire, en empêchant tout code PHP de s'exécuter et enregistre ensuite ces données dans un fichier, qui va être lu par un script Java. Java ne fonctionne pas comme PHP, un script qui est inoffensif pour PHP ne le sera pas forcément pour Java. Dans cet article, vous verrez principalement comment être certain que les données traitées sont bien du type souhaité. Pour plus de détails concernant les injections et autres problèmes de sécurité, je vous renvoie aux nombreux articles qui sont déjà parus dans *PHP Solutions*.

Contrôles des données

Le contrôle des données est le fait de vérifier les données. Par exemple, lors de la validation d'un formulaire vous devez dans un premier temps tester si tous les champs obligatoires sont remplis et s'ils le sont correctement. Si vous demandez une adresse e-mail, vous devez vérifier qu'elle est au bon format, si vous demandez une année de naissance, vérifier que vous récupérez bien une date valide (c'est-à-dire un nombre entier).

Il existe une extension PHP nommée FilterPHP qui permet d'effectuer une validation ou un nettoyage de données. Pour pouvoir utiliser cette extension vous aurez besoin d'une version de PHP supérieure ou égale à la version 5.2.0. Pour les versions an-

tières, il existait une extension PECL qui a désormais été abandonné. Si votre version de PHP est inférieure à la version 5.2.0, je vous conseille vivement de faire une mise à jour, ne serait-ce que pour les problèmes de sécurité.

FilterPHP propose de nombreux filtres et l'extension n'est toujours pas terminée à 100%. Il y aura de nombreux ajouts

Nom du filtre	ID du filtre
int	257
boolean	258
float	259
validate_regexp	272
validate_url	273
validate_email	274
validate_ip	275
string	513
stripped	513
encoded	514
special_chars	515
unsafe_raw	516
email	517
url	518
number_int	519
number_float	520
magic_quotes	521
callback	1024

Figure 1. Résultat de l'affichage des différents filtres et leurs ID

et modifications par le futur. Le Listing 1 permet d'afficher la liste des filtres disponibles ainsi que leurs ID correspondants. Vous pouvez voir le résultatat de ce listing sur la Figure 1.

La fonction la plus importante de FilterPHP est `filter_var()`, elle permet de contrôler et d'assainir tous les types de données grâce aux différents filtres passés en arguments.

Contrôler une valeur booléenne

La validation d'une valeur booléenne via FilterPHP se fait grâce au filtre `FILTER_VALIDATE_BOOLEAN`. Le Listing 2 montre comment utiliser ce filtre. Si le filtre est validé, ce script affichera 1 (équivalent de `true`), sinon il n'affichera rien. Les autres valeurs acceptables pour `true` sont :

- 1
- "1"
- "yes"
- "true"
- "on"
- TRUE

Les valeurs acceptables pour `false` sont :

- 0
- "0"
- "no"
- "false"
- "off"
- NULL
- FALSE
- "

Contrôler un entier

Le contrôle d'un entier se fait grâce au filtre `FILTER_VALIDATE_INT` passé en argument à la fonction `filter_var()`. Le Listing 3 présente l'utilisation de ce filtre. Ici on teste la variable `$entier` et on retourne le résultat dans `$resultat`, ensuite si le test n'est pas faux, on affiche la valeur renournée par le test (nous sommes donc certains que cette valeur est un entier), sinon on indique que `$entier` n'est pas un nombre entier.

Contrôler un entier dans une plage de valeurs

Pour contrôler un entier dans une plage de valeurs il faut ajouter un tableau d'options comme troisième argument à la fonction `filter_var()`. La plage de valeurs est définie par '`min_range`' et `max_range`'. Comme vous l'avez sûrement deviné, il s'agit du minimum et du maximum de la plage de valeurs. Le Listing 4 présente l'utilisation du tableau d'options afin de contrôler un entier dans une plage de valeurs déterminées. Le

Listing 1. Affichage des différents filtres et leurs ID

```
<table>
<tr><td>Nom du filtre</td><td>ID du filtre</td></tr>
<?php
foreach( filter_list () as $filtre)
{
    Echo '<tr><td>'.
$filtre.'</td><td>'.filter_id($filtre).'</td></tr>'."\n";
}
?>
</table>
```

Listing 2. Exemple de contrôle de valeurs booléennes

```
<?php
$var=true;
echo filter_var ( $var , FILTER_VALIDATE_BOOLEAN );
?>
```

Listing 3. Contrôler un entier

```
<?php
$entier = '12m';
$resultat = filter_var ( $entier , FILTER_VALIDATE_INT ); //dans ce cas
$resultat vaut false
if($resultat != false)
{
    echo $resultat.' est bien un entier';
}
else
{
    echo 'Erreur ceci n\'est pas un nombre entier';
}
//si vous remplacez $entier par 2 par exemple, $resultat vaudra 2 et le
script affichera : 2 est bien un entier
?>
```

Listing 4. Contrôler un entier dans une plage de valeurs

```
<?php
$entier = 22;//l'entier à valider
$min = 1;//le minimum de la plage de validation
$max = 100;//le maximum de la plage de validation
echo filter_var($entier , FILTER_VALIDATE_INT, array("options"=> array(
"min_range"=>$min, "max_range"=>$max)));
?>
```

Listing 5. Contrôler un entier octal ou hexadécimal

```
<?php
$entierOctal = 0666;//l'entier octal à valider(il est égal à 438 en base
décimale)
$min = 1;//le minimum de la plage de validation
$max = 450;//le maximum de la plage de validation
echo filter_var($entierOctal , FILTER_VALIDATE_INT, array("flags" =>
FILTER_FLAG_ALLOW_OCTAL, "options"=> array( "min_range"=>$min, "max_
range"=>$max)));
//affiche 438 car 1<438<450
?>
```

tableau d'options se forme de la façon suivante : `array("options"=> array("min_range"=>$min, "max_range"=>$max))`. Sur tout n'oubliez pas le "`options"=> array(...)`. Si vous n'indiquez pas un tableau dans la case `options` cela ne fonctionnera pas et une erreur sera renournée. Dans ce listing, l'entier est bien dans la plage de valeurs, le nombre vingt-deux est donc affiché. Vous pouvez changer le nombre `$entier` pour qu'il ne soit plus dans la plage de valeurs, ainsi le script n'affichera rien car le résultat du test sera `false`.

Vous pouvez bien entendu n'indiquer qu'une valeur minimale ou maximale dans le tableau d'options. Dans ce cas, le test ne

se fera que par rapport à ce minimum ou ce maximum.

Contrôler un entier octal ou hexadécimal

Les contrôles d'un entier octal et d'un entier hexadécimal se font respectivement grâce aux `flags` (drapeau en français, ce sont des options supplémentaires pour préciser ce que l'on souhaite faire) `FILTER_FLAG_ALLOW_HEX` et `FILTER_FLAG_ALLOW_OCTAL` qui peuvent être passés en option de deux façons différentes :

```
filter_var($valeurHexa,FILTER_VALIDATE_INT,array("flags"=>FILTER_FLAG_ALLOW_HEX ));
```

Listing 6. Contrôler un flottant

```
<?php
//on initialise un tableau de différentes valeurs
$tableau = array(10.6, "deux virgule trois", "0", "-98745,123", "skdfj41,gg2",
-255);
//on filtre le tableau
$tableauDeValidation = filter_var($tableau ,FILTER_VALIDATE_FLOAT
,FILTER_REQUIRE_ARRAY);
//on affiche le résultat
var_dump($tableauDeValidation);
?>
```

Listing 7. Contrôler un flottant en choisissant le séparateur decimal

```
<?php
$flottant='15-9';
$dec_sep='-';
echo filter_var( $flottant , FILTER_VALIDATE_FLOAT , array( "options"
=>array( "decimal" => $dec_sep )) );
//affichera 15.9
?>
```

Listing 8. Contrôler les URL avec Filter PHP

```
<?php
$url = "http://phpsolmag.org" ;
if(filter_var($url, FILTER_VALIDATE_URL) === FALSE )
{
    echo 'l\'URL '.$url.' n\'est pas valide' ;
}
else
{
    echo 'l\'URL '.$url.' est valide';
}
//affiche l'URL http://phpsolmag.org est valide
?>
```

Listing 9. Contrôler des URL avec le Flag FILTER_FLAG_SCHEME_REQUIRED

```
<?php
$url = "x://x" ;
if(filter_var( $url, FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED
) === FALSE)
{
    echo 'l\'URL '.$url.' n\'est pas valide' ;
}
else
{
    echo 'l\'URL '.$url.' est valide';
}
//si vous faites le test, vous remarquez que cette URL est valide, ce qui démontre bien que seulement la syntaxe caractère://caractère est testée.
?>
```

Listing 10. Contrôler des URL avec le flag FILTER_FLAG_HOST_REQUIRED

```
<?php
$url = "http://phpsolmag.org" ;
if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_HOST_REQUIRED) === FALSE)
{
    echo 'l\'URL '.$url.' n\'est pas valide' ;
}
else
{
    echo 'l\'URL '.$url.' est valide';
}
//ici l'url est validée
?>
```

Listing 11. Contrôler des URL avec le Flag FILTER_FLAG_PATH_REQUIRED

```
<?php
$url = "http://phpsolmag.org/dosier/" ;
if(filter_var($url ,FILTER_VALIDATE_URL ,FILTER_FLAG_PATH_REQUIRED) === FALSE )
{
    echo 'l\'URL '.$url.' n\'est pas valide' ;
}
else
{
    echo 'l\'URL '.$url.' est valide';
}
//l'URL est validée car il y a un chemin dans l'URL
?>
```

Ou

```
filter_var($valeurHexa, FILTER_VALIDATE_INT, FILTER_FLAG_ALLOW_HEX
);
```

Vous pouvez bien évidemment combiner le test des entiers octaux et hexadécimaux avec le test dans une plage de valeurs. Le Listing 5 présente le test d'un nombre octal dans une plage de valeurs. Attention vous devez garder à l'esprit que vous ne testez pas un entier décimal, par conséquent vous devez effectuer une conversion pour définir le minimum et le maximum de la plage de valeurs.

Contrôler un nombre flottant**(nombre avec virgule)**

Vous commencez à avoir l'habitude des contrôles de nombres, pour les nombres flottants, il faut utiliser le filtre FILTER_VALIDATE_FLOAT. Pour cet exemple, nous allons voir comment tester plusieurs nombres flottants. Pour cela, nous allons créer un tableau de valeurs et le passer en paramètre à la place de la variable habituelle. Pour que le tableau soit analysé par FilterPHP il faut ajouter un troisième argument, FILTER_REQUIRE_ARRAY, qui permet de signaler à la fonction que le premier argument est un tableau et non un nombre. Le Listing 6 est un exemple de contrôle de plusieurs flottants. Ce code affichera ce résultat :

```
array(6){
[0]=> float(10.6)
[1]=> bool(false)
[2]=> float(0)
[3]=> bool(false)
[4]=> bool(false)
[5]=> float(-255) }
```

Seulement les éléments du tableau qui sont des flottants restent des flottants, les autres sont retournés en booléens avec la valeur *false*.

Le séparateur décimal n'est pas le même dans tous les pays, et il faut savoir en tenir compte. Les créateurs de FilterPHP l'ont également remarqué et ont pensé à donner la possibilité de choisir le séparateur décimal. Pour cela, il suffit de mettre dans le tableau d'options, une clé 'décimal' ayant pour valeur le séparateur utilisé . Attention, le séparateur doit être un caractère unique, en effet, si vous choisissez , comme séparateur, FilterPHP retournera une erreur. Le Listing 7 présente comment contrôler un nombre flottant ayant pour séparateur décimal un -.

Contrôler une URL

Les URL sont très complexes à vérifier de par le nombre de formes qu'elles peuvent prendre. En effet, aucune norme n'a été

faite pour les URL. FilterPHP aide à vérifier les différents composants d'une URL grâce à différents filtres. Le filtre à appliquer est FILTER_VALIDATE_URL. Plusieurs *flags* sont disponibles pour pouvoir valider telle ou telle chose. Je vous les décris ci-dessous :

- FILTER_FLAG_SCHEME_REQUIRED – indique qu'il faut que l'URL soit de la forme *x://x*, couramment on souhaite avoir *http://x* (ou *x* est un caractère quelconque). Ce *flag* est l'équivalent du filtre FILTER_VALIDATE_URL sans *flag* supplémentaire.
- FILTER_FLAG_HOST_REQUIRED – d'après les tests effectués, ce *flag* fait exactement la même chose que le *flag* FILTER_FLAG_SCHEME_REQUIRED.
- FILTER_FLAG_PATH_REQUIRED – ce *flag* permet de tester si un chemin est présent dans l'URL.
- FILTER_FLAG_QUERY_REQUIRED – ce *flag* vérifie qu'une requête est de la forme *php?id*.

Des exemples s'imposent. Le Listing 8 est le contrôle d'une URL avec FILTER_VALIDATE_URL sans aucun *flag*. Dans l'exemple l'URL est validée. Comme dit précédemment l'URL sera validée si elle est de la forme *x://x* (ou *x* est un caractère quelconque).

Le Listing 9 est l'exemple avec le *flag* FILTER_FLAG_SCHEME_REQUIRED, ce *flag* fait la même chose que la fonction sans *flag*, en fait c'est le *flag* par défaut de FILTER_VALIDATE_URL. Pour l'exemple, *x://x* est bien validé. Le Listing 10 est l'application du *flag* FILTER_FLAG_HOST_REQUIRED. Comme son nom l'indique, ce *flag* vérifie qu'un nom d'hôte est spécifié, il vérifie également la syntaxe *caractère://caractère*. Le *flag* FILTER_FLAG_PATH_REQUIRED est utilisé dans le Listing 11. Ici, le test est validé car l'URL contient un chemin après l'URL de base. Pour finir le contrôle des URL, le Listing 12 présente l'utilisation du *flag* FILTER_FLAG_QUERY_REQUIRED qui permet de vérifier qu'il y a des options à la requête d'un fichier PHP.

Contrôler une adresse IP

Le contrôle d'adresse IP se fait également grâce à la fonction filter_var(), et s'utilise avec le filtre FILTER_VALIDATE_IP. De nouveaux *flags* sont utilisés comme arguments. Bien entendu, il est possible de contrôler simplement une adresse IP via des expressions régulières, mais conserver le même outil de contrôle est parfois plus pratique voire cohérent. Voici les *flags* utilisés pour contrôler différentes IP :

- FILTER_FLAG_IPV4 – permet de tester si l'adresse IP est une IPv4,

Listing 12. Contrôler des URL avec le Flag FILTER_FLAG_QUERY_REQUIRED

```
<?php

$url = "http://phpsolmag.org/info.php?id=1" ;
if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED) === FALSE)
{
    echo 'L\'URL '.$url.' n\'est pas valide' ;
}
else
{
    echo 'L\'URL '.$url.' est valide';
}
?>
```

Listing 13. Contrôler une adresse IP

```
<?php

$ip = "192.168.1.1"; //une adresse IPV4
if(filter_var($ip, FILTER_VALIDATE_IP) === FALSE)
{
    echo "$ip est une adresse IP valide";
}
else
{
    echo "$ip n'est pas une adresse IP valide";
}
//se script affichera : 192.168.1.1 est une adresse IP valide
?>
```

Listing 14. Contrôler une adresse IPV4

```
<?php

$ip = "192.168.0"; //une adresse IPV4
if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV4) === FALSE)
{
    echo "$ip est une adresse IPV4 valide";
}
else
{
    echo "$ip n'est pas une adresse IPV4 valide";
}
//se script affichera : 192.168.0 n'est pas une adresse IPV4 valide
?>
```

Listing 15. Contrôler une adresse IPV6

```
<?php

$ip = "ffff:0000:0a88:85a3:0000:0000:ac1f:8001"; //une adresse IPV6
if(filter_var($ip, FILTER_VALIDATE_IP) === FALSE)
{
    echo " $ip est une adresse IP valide";
}
else
{
    echo " $ip n'est pas une adresse IP valide";
}
//se script affichera : ffff:0000:0a88:85a3:0000:0000:ac1f:8001 est une adresse IPV6 valide
?>
```

Listing 16. Contrôler si une adresse IP est privée

```
<?php

$ip = "192.168.1.1"; //une adresse IPV4 privée
if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE) === FALSE)
{
    echo " $ip est une adresse IPV4 privée";
}
else
{
    echo " $ip est une adresse IPV4 publique";
}
//se script affichera : 192.168.1.1 est une adresse IPV4 privée
?>
```

Listing 17. Contrôler si une adresse IP est dans une plage réservée

```
<?php
$ip = "255.255.255.255"; //une adresse IPV4 réservée
if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_RES_RANGE) === FALSE)
{
    echo " $ip est une adresse IPV4 réservée";
}
else
{
    echo " $ip n'est pas une adresse IPV4 réservée";
}
//ce script affichera : 255.255.255.255 est une adresse IPV4 réservée
?>
```

Listing 18. Contrôler une adresse e-mail avec FilterPHP

```
<?php
$email = "jeremy.rafflin@laposte.net" ;
if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
{
    echo " $email n'est pas valide";
}
else
{
    echo " $email est valide" ;
}
//ce script affichera : jeremy.rafflin@laposte.net est valide
?>
```

Listing 19. Exemple d'utilisation des expressions régulières dans FilterPHP

```
<?php
$texte = "Hello world!" ;
//on vérifie si $texte commence par H
if(filter_var($texte, FILTER_VALIDATE_REGEXP,
array("options"=>array("regexp"=>"/^H(.*)/")) ) === false)
{
    echo "L'expression n'a pas été trouvée";
}
else
{
    echo "La chaîne de caractères commence bien par un H";
}
//ce script affichera : La chaîne de caractères commence bien par un H
?>
```

Listing 20. Contrôler une adresse e-mail grâce aux expressions régulières

```
<?php
$adresse = 'jeremy.rafflin@laposte.net';
//l'expression régulière permettant de valider une adresse e-mail
$Syntaxe='#^[\w.-]+@[^\w.-]+\.[a-zA-Z]{2,6}$#';
if(preg_match($Syntaxe,$adresse))//si le mail correspond à la syntaxe
{
    echo "l'adresse e-mail est valide";
}
else//si le mail ne correspond pas à la syntaxe
{
    echo "l'adresse e-mail n'est pas valide";
}
//le mail est validé dans notre cas
?>
```

Listing 21. Contrôler une URL

```
<?php
// L'URL du site web
$url = "http://phpsolmag.org/fr";
if (@fopen($url, 'r'))
{
    echo 'URL valide !';
}
else
{
    echo 'URL non valide !';
}
?>
```

- **FILTER_FLAG_IPV6** – permet de tester si l'adresse IP est une IPv6,
- **FILTER_FLAG_NO_PRIV_RANGE** – permet de tester si l'adresse n'est pas dans la plage d'adressage privée déterminée par la RCF. Ce flag fonctionne avec les adresses IPV4 et IPV6,
- **FILTER_FLAG_NO_RES_RANGE** – permet de tester si l'adresse n'est pas dans la plage d'adressage privée. Ce flag fonctionne avec les adresses IPV4 et IPV6.

Le Listing 13 permet de tester une adresse IP, l'adresse est validée car correcte. Le Listing 14 permet de tester une adresse IPV4. Le Listing 15 permet de tester une adresse IPV6. Le Listing 16 permet de tester si une adresse IPV4 est privée ou publique. Le Listing 17 permet de tester si une adresse IPV4 est dans une plage réservée ou non.

Contrôler une adresse e-mail

Contrairement aux URL, il existe une règle d'écriture pour les adresses électroniques. Cela simplifie l'usage de FilterPHP. En s'appuyant sur la description suivante :

- les adresses doivent contenir une chaîne de caractères contenant des lettres minuscules ou majuscules, des chiffres, quelques caractères spéciaux tel que **_**,
- suivi de **@**,
- puis de nouveau une chaîne de caractères et pour finir un point suivi de deux à six lettres .

Il est facile d'exploiter FilterPHP pour tester la validité d'une adresse. FilterPHP ne valide que le format **caractère@caractère**, nous verrons plus loin dans cet article comment contrôler plus efficacement une adresse e-mail. Pour filtrer une adresse e-mail, il faut utiliser le filtre **FILTER_VALIDATE_EMAIL**. Le Listing 18 permet le contrôle des adresses e-mails grâce à FilterPHP.

Effectuer un contrôle grâce aux expressions régulières et FilterPHP

Il est possible d'utiliser les expressions régulières conjointement à FilterPHP. Pour ce faire, les créateurs de la classe ont prévu un filtre **FILTER_VALIDATE_REGEXP**, qui permet grâce au tableau d'options de rechercher l'expression régulière dans une chaîne de caractères. Le tableau d'options se présente de la façon suivante :

```
array("options" =>array("regexp" =>
$syntaxe))
```

Il faut donc remplacer **\$syntaxe** par la syntaxe de l'expression régulière ou créer une variable **\$syntaxe** contenant l'expression régulière. Le Listing 19 teste si la chaîne de

caractères contenue dans \$texte commence par H. C'est donc le cas puisque *Hello world!* commence par H.

Affiner le contrôle des URL et des adresses e-mail en conjungant FilterPHP et expressions régulières.

Dans la partie Contrôler une URL et Contrôler une adresse e-mail nous avons vu que FilterPHP ne contrôlait pas la totalité de l'URL ou de l'e-mail. Pour des contrôles plus précis, je vous conseille d'utiliser les expressions régulières.

Pour les mails, l'expression régulière est assez simple et le Listing 20 permet de contrôler une adresse e-mail en utilisant les expressions régulières dans la fonction `preg_match()`. Cette fonction prend comme premier argument l'expression régulière et comme second argument la chaîne de caractères à tester. Ici, l'expression régulière correspond à tester : une chaîne de caractères composée de : chiffres, lettres minuscules ou majuscules et certains caractères spéciaux comme , _ suivie d'un @ puis une chaîne de caractères composée des mêmes éléments que précédemment, puis un . et qui se termine par une chaîne de caractères de deux à six caractères composée de lettres minuscules et majuscules.

Malheureusement l'expression régulière permettant de valider les URL est très compliquée à trouver, dû au très grand nombre de possibilités. Je vous recommande donc de rechercher les expressions régulières qui permettent de contrôler les URL acceptées pour votre site (par exemple, il est possible que votre site n'accepte que les URL valides effectuant une requête sur les fichiers .gif). Il existe cependant une méthode simple pour vérifier si une URL existe. Elle nécessite l'utilisation de `fopen()` (si votre serveur ne le permet pas, vous ne pourrez pas utiliser cette méthode) et demande plus de ressources au serveur. Le but de cet article n'étant pas de présenter la lecture de fichiers, je vous renvoie à l'article sur la lecture de fichiers du magazine PHP Solutions 4/2009. Le Listing 21 vous présente cette méthode. Quelques explications sur ce code :

- dans un premier temps on définit l'URL à tester,
- ensuite on teste si on arrive à lire le fichier correspondant à l'URL grâce à `fopen()` (le second argument est le mode à effectuer, ici 'r' veut dire que l'on souhaite lire le fichier (il existe d'autres méthodes)),
- l'arobase permet de ne pas afficher les erreurs s'il y en a, en effet si l'URL ne peut pas être lue, une erreur sera affichée si vous ne mettez pas le '@'.

Avec l'URL du listing, le test est validé.

Listing 22. Temps d'exécution des expressions régulières

```
<?php

for($n=0;$n<5;$n++)
{
    $string = "Hello world!";
    $debutF = microtime();
    for($i=0;$i<10000;$i++)//une grosse boucle pour pouvoir minimiser les erreurs aléatoires
    {
        //on effectue le test avec FilterPHP
        $b=filter_var($string, FILTER_VALIDATE_REGEXP,
array("options"=>array("regexp"=>"/^H(.*)/")));
    }
    $finF= microtime();
    $debutR = microtime();
    for($i=0;$i<10000;$i++)//une grosse boucle pour pouvoir minimiser les erreurs aléatoires
    {
        //on effectue le test avec preg_match()
        $c=preg_match("/^T(.*)/",$string);
    }
    $finR=microtime();
    echo "Test ".$n."  


```

Listing 23. Assainir un nombre entier

```
<?php

//on déclare un faux nombre entier
$entier = "72+5sldslkd€" ;
echo filter_var($entier ,FILTER_SANITIZE_NUMBER_INT);
//affiche 72+5 (notez que le '+' est conservé)
?>
```

Listing 24. Assainir un nombre flottant grâce aux différents Flags

```
<?php

/*on déclare un faux nombre flottant avec un séparateur des milliers, un séparateur décimal et une puissance e */
$flotant = "87,472.598+5e12 et des caractères qui seront supprimés" ;
echo filter_var($flotant , FILTER_SANITIZE_NUMBER_FLOAT)."  


```

Listing 25. Assainir et encoder une URL

```
<?php

//une URL avec des caractères interdits
$url = "http://wYww.googlE.fr" ;
echo filter_var($url, FILTER_SANITIZE_URL);

// affiche http://www.google.fr
$url = "http://test.free.fr/dossier/fichier.php?id=1&nom=prenom";
echo filter_var($url, FILTER_SANITIZE_ENCODED);

//affiche http%3A%2F%2Ftest.free.fr%2Fdossier%2Ffichier.php%3Fid%3D1%26nom%3D
$prenom
?>
```

Listing 27. Assainir une chaîne de caractères de toutes les balises

```
<?php
//utilisation de FILTER_SANITIZE_STRING seul
$string = chr(165).' ? <>!#$%^&*()"<toto> "\'HIHI\'</toto>{~bobthebuilder
'.chr(20);
echo filter_var ($string, FILTER_SANITIZE_STRING);
/*affiche : ¥ ? !@#$%^&*()" "HIHI"{'~bobthebuilder
code source : ¥ ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
Notez que les guillemets sont encodés dans le code source
*/
?>
```

Listing 28. Assainir une chaîne de caractères sans encoder les guillemets

```
<?php
$string = chr(165).' ? <>!#$%^&*()"<toto> "\'HIHI\'</toto>{~bobthebuilder
'.chr(20);
echo filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_NO_ENCODE_QUOTES);
/*affiche : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
Notez que les guillemets ne sont pas encodés dans le code source
*/
?>
```

Listing 29. Assainir une chaîne de caractères en encodant les caractères dont le code ASCII est inférieur à 32 et supérieur à 127

```
<?php
$string = chr(165).' ? <>!#$%^&*()"<toto>"\HIHI\"</toto>{~bobthebuilder
'.chr(20);
echo filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_ENCODE_LOW);
/*affiche : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : ¥ ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
&#20;
Notez que les guillemets et le caractère dont le code ASCII est inférieur à
32 sont codés
*/
echo '<br />'.filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_ENCODE_HIGH);
/*affiche : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : &#165; ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
Notez que les guillemets et le caractère dont le code ASCII est supérieur
à 127 sont codés
*/
?>
```

Listing 30. Assainir une chaîne de caractères en supprimant les caractères dont le code ASCII est inférieur à 32 et supérieur à 127

```
<?php
$string = chr(165).' ? <>!#$%^&*()"<toto> "\'HIHI\'</toto>{~bobthebuilder
'.chr(20);
echo filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_LOW);
/*affiche : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : ¥ ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
Notez que le caractère dont le code ASCII est inférieur à 32 est supprimé
*/
echo '<br />'.filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_HIGH);
/*affiche : ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
Notez que le caractère dont le code ASCII est supérieur à 127 est supprimé
*/
?>
```

Listing 31. Assainir une chaîne de caractères en encodant le caractère &

```
<?php
$string = chr(165).' ? <>!#$%^&*()"<toto> "\'HIHI\'</toto>{~bobthebuilder
'.chr(20);
echo filter_var ($string, FILTER_SANITIZE_STRING, FILTER_FLAG_ENCODE_AMP);
/*affiche : ¥ ? !@#$%^&*()" "'HIHI"{'~bobthebuilder
code source : ¥ ? !@#$%^&*()" &#34;&#39;HIHI&#39;&#34;{~bobthebuilder
Notez que le caractère & a été codé
*/
?>
```

Listing 26. Assainir et encoder une URL

```
<?php
$email = "moi&mesfamis@toto(.fr\\
titi.com";
echo filter_var($email, FILTER_SANITIZE_EMAIL);

//affiche moi&mesamis@toto.frtiti.
com

?>
```

Comparaison des expressions régulières avec FilterPHP et preg_match()

Maintenant que nous avons vu rapidement que les expressions régulières peuvent être utilisées via `preg_match()` ou FilterPHP, parlons performance. L'extension FilterPHP est vraiment pratique, mais face à la fonction `preg_match()` elle est vraiment beaucoup trop lente. Le Listing 22 permet de tester le temps d'exécution d'une expression régulière via les deux méthodes présentées ici. Le résultat de ce code est présent sur la Figure 2. On voit clairement que `preg_match()` est nettement plus rapide à chacun des cinq tests. FilterPHP n'est cependant pas complètement terminé et il y a régulièrement des mises à jour, on peut donc espérer une mise à jour pour que l'utilisation des expressions régulières via FilterPHP soit concurrente vis-à-vis des autres fonctions équivalentes.

Filtrer les données

Contrôler les données est très pratique dans le cas où l'on peut demander une correction de celles qui ne sont pas correctes. Par exemple, dans le cas du traitement d'un formulaire, on peut demander à l'utilisateur de corriger les données. Mais, si les données proviennent

Test 0

temps d'exécution avec FilterPHP = 0.03338
temps d'exécution avec pregmatch = 0.016442

Test 1

temps d'exécution avec FilterPHP = 0.034189
temps d'exécution avec pregmatch = 0.014783

Test 2

temps d'exécution avec FilterPHP = 0.033512
temps d'exécution avec pregmatch = 0.015474

Test 3

temps d'exécution avec FilterPHP = 0.035693
temps d'exécution avec pregmatch = 0.014675

Test 4

temps d'exécution avec FilterPHP = 0.033314
temps d'exécution avec pregmatch = 0.017342

Figure 2. Résultat du temps d'exécution des expressions régulières

d'une base de données ou d'une autre application que l'on ne contrôle pas, on ne peut pas forcément demander une correction en cas d'erreur. Il faut alors filtrer les données, c'est-à-dire éliminer tout ce que l'on ne souhaite pas. Par exemple, une application nous fournit le prix d'un article, ce prix doit être un entier et ne doit pas contenir de lettres. Je vous conseille toutefois d'utiliser les filtres en supplément du contrôle de données (il vaut mieux deux protections plutôt qu'une, cela compliquera la tâche d'un éventuel *hacker*). Comme pour le contrôle de données, le filtrage se fait grâce à la fonction `filter_var()`.

Filtrer un entier

L'assainissement d'un entier est plutôt simple car il suffit d'utiliser le filtre `FILTER_SANITIZE_NUMBER_INT`. Le Listing 23 présente la méthode à utiliser.

Filtrer un flottant

L'assainissement d'un flottant est similaire à celui d'un entier, il se fait grâce au filtre `FILTER_SANITIZE_NUMBER_FLOAT`. Attention, avec ce simple filtre, le flottant est converti en nombre entier. Par exemple, si vous utilisez ce filtre sur un flottant tel que 72.56, la fonction retournera 7256. Cependant, il existe différents *flags* pour ce filtre :

- `FILTER_FLAG_ALLOW_FRACTION` – permet de conserver le séparateur décimal (le point chez les anglo-saxons),
- `FILTER_FLAG_ALLOW_THOUSAND` – permet de conserver le séparateur des milliers (la virgule),
- `FILTER_FLAG_ALLOW_SCIENTIFIC` – permet de conserver le 'e' de l'écriture scientifique.

Vous pouvez vous référer au Listing 24 pour les exemples. Je vous passe les commentaires puisqu'ils sont présents dans le listing.

Assainir ou encoder une URL

Fort heureusement pour nous, l'assainissement d'une URL est beaucoup plus simple que leurs contrôles. Les caractères qui peuvent être présents dans une URL sont assez restreints, en effet, les caractères autorisés sont les lettres, les chiffres et les caractères suivants : `$_.+!*'0,{|\\^~[]><#%>?:@&=`. Pour nettoyer une URL de tous les autres caractères vous devez utiliser le filtre `FILTER_SANITIZE_URL`. Vous trouverez dans la première partie du Listing 25 ce que fait ce filtre.

La seconde partie du listing permet d'encoder l'URL grâce au filtre `FILTER_SANITIZE_ENCODED`, ainsi vous pouvez passer des URL en paramètre d'URL. Quatre *flags* sont dis-

Listing 32. Sécuriser les guillemets

```
<?php

$string = " des guillemets ' ' \"\"";
echo filter_var($string, FILTER_SANITIZE_MAGIC_QUOTES);

/* affiche : des guillemets \' \' \\\"\" */

?>
```

Listing 33. Utilisation des filtres de rappels

```
<?php

//1ère partie
function espaceEnTiret($string)
{
    return str_replace(" ", "-", $string);
}

//on déclare un numéro de téléphone
$string = "03 35 96 84 56";
echo filter_var($string, FILTER_CALLBACK, array("options" => "espaceEnTiret"));

//affiche 03-35-96-84-56

//2ème partie
class test
{
    function testMail($mail) {
        if(!preg_match('#^[\w.-]+@[\\w.-]+\.[a-zA-Z]{2,6}$#', $mail))
        {
            throw new Exception( "Exception, l'adresse n'est pas valide");
        }
    }
    $mail="un faux mail";
    try
    {
        filter_var($mail, FILTER_CALLBACK, array("options" =>array('test','test
Mail')));
    }
    catch ( Exception $e )
    {
        echo $e ->getMessage();
    }
}

//affiche Exception, l'adresse n'est pas valide
?>
```

Listing 34. test.php?lien=http://lien 1.php

```
<?php

$url = filter_input(INPUT_GET, 'lien', FILTER_SANITIZE_ENCODED);
echo '<a href="test.php?lien='.$url.'">Le lien</a>';

//code source : <a href="test.php?lien=http%3A%2F%2Flien%201.php">Le lien</a>
?>
```

ponibles comme options à ce filtre. Je vous les présenterai un peu plus tard dans cet article. Retenez que les *flags* disponibles sont :

- `FILTER_FLAG_STRIP_LOW` – supprime tous les caractères dont le code ASCII est inférieur à 32,
- `FILTER_FLAG_STRIP_HIGH` – supprime tous les caractères dont le code ASCII est supérieur à 127,
- `FILTER_FLAG_ENCODE_LOW` – encode tous les caractères dont le code ASCII est inférieur à 32,
- `FILTER_FLAG_ENCODE_HIGH` – encode tous les caractères dont le code ASCII est supérieur à 127.

Assainir une adresse e-mail

Comme pour l'assainissement d'une URL, le filtrage d'une adresse e-mail se fait grâce à un filtre : `FILTER_SANITIZE_EMAIL`. Celui-ci n'accepte donc aucun *flag* optionnel puisqu'il supprime tous les caractères excepté les chiffres, les lettres et les caractères suivants : `$_.+!*'0,{|\\^~[]><#%>?:@&=`. Le Listing 26 vous montre comment effectuer ce filtrage.

Filtrer et encoder une chaîne de caractères

À la fin de cette partie, vous saurez contrôler et donc sécuriser totalement les chaînes de caractères. Vous saurez encoder

Listing 35. test.php

```

<form method="post" action="test.php">
<input type="text" name="nom" value="jeremy" /><br />
<input type="text" name="age" value="21a" /><br />
<input type="text" name="email" value="jeremy.rafflin@laposte.net" /><br />
<input type="text" name="url" value="http://phpsolmag.org/fr" />
<input type="submit" />
</form>

<?php

//on déclare les variables à récupérer avec leurs filtres correspondants
$filtre = array(
"age" =>array("filter" => FILTER_VALIDATE_INT, "options"=>array("min_range"=>0,"max_range"=>100)),
"nom" => FILTER_SANITIZE_SPECIAL_CHARS ,
"email" => FILTER_SANITIZE_EMAIL ,
"url" => array("filter" => FILTER_CALLBACK, "options" => "testUrl")
);

//on déclare une fonction qui va permettre le test d'une URL
function testUrl($url)
{
    if(@fopen($url, 'r'))
    {
        //on retourne l'URL si elle est valide
        return $url;
    }
    else
    {
        //si l'URL n'est pas valide on retourne false
        return false;
    }
}

//on effectue les filtres aux variables récupérées via INPUT_POST
$tableauFiltre = filter_input_array(INPUT_POST, $filtre);

//on affiche le résultat
echo $tableauFiltre['nom'].'<br />';
echo $tableauFiltre['age'] .'<br />';
echo $tableauFiltre['email'].'<br />' ;
echo $tableauFiltre['url'];

?>

```

et supprimer tous les caractères spéciaux suivant vos besoins. Dans cette partie, je vais utiliser la fonction chr(CODE_ASCII) qui retourne le caractère correspondant au code ASCII passé en argument. Je me servirai de cette fonction pour générer les caractères que l'on ne trouve pas sur les claviers traditionnels. Vous pouvez trouver le code ASCII des caractères sur <http://www.whatasciicode.com/>.

Suppression de balise XHTML

Vous pouvez supprimer ou encoder les balises XHTML. Le filtre FILTER_SANITIZE_STRING supprime les balises XHTML (attention si les signes < ou > sont présents dans la chaîne de caractères, du texte peut être supprimé de façon non souhaitée) et encode les guillemets. Vous pouvez voir l'effet de ce filtre dans le Listing 27.

Étant donné que vous devriez commencer à bien comprendre le fonctionnement des *flags*, je ne vais pas vous détailler chaque *flag* disponible pour ce filtre, les commentaires présents dans les listing devraient suffire. Les options disponibles sont :

- FILTER_FLAG_NO_ENCODE_QUOTES – permet de ne pas encoder les guillemets (Listing 28),
- FILTER_FLAG_ENCODE_LOW – encode tous les caractères dont le code ASCII est inférieur à 32 (Listing 29),
- FILTER_FLAG_ENCODE_HIGH – encode tous les caractères dont le code ASCII est supérieur à 127 (Listing 29),
- FILTER_FLAG_STRIP_LOW – supprime tous les caractères dont le code ASCII est inférieur à 32 (Listing 30),
- FILTER_FLAG_STRIP_HIGH – supprime tous les caractères dont le code ASCII est supérieur à 127 (Listing 30),
- FILTER_FLAG_ENCODE_AMP – encode le caractère & (Listing 31).

Encodage de balise XHTML

Il existe aussi un filtre qui permet d'encoder les caractères '<', '>', '&', les guillemets ainsi que tous les caractères dont le code ASCII est inférieur à 32. Le nom de ce filtre est FILTER_SANITIZE_SPECIAL_CHARS. Il peut prendre seulement trois *flags* en options (FILTER_FLAG_ENCODE_LOW n'est pas utile puisque cette action est effectuée par défaut) :

- FILTER_FLAG_ENCODE_HIGH – encode tous les caractères dont le code ASCII est supérieur à 127,
- FILTER_FLAG_STRIP_LOW – supprime tous les caractères dont le code ASCII est inférieur à 32,
- FILTER_FLAG_STRIP_HIGH – supprime tous les caractères dont le code ASCII est supérieur à 127.

Je ne détaille pas ce filtre afin de vous laissez vous exercer.

Le filtre qui ne fait rien

La documentation décrit le filtre FILTER_UNSAFE_RAW comme un filtre qui ne fait rien. En effet, ce filtre seul ne fait absolument rien. Seul les *flags* optionnels vous permettent d'effectuer des actions. Les options disponibles sont les mêmes que pour le filtre FILTER_SANITIZE_STRING excepté du *flag* FILTER_FLAG_NO_ENCODE_QUOTES (cela paraît logique puisque ce *flag* permettait de ne pas encoder les guillemets, or le filtre FILTER_UNSAFE_RAW n'encode pas les guillemets, il n'est donc pas nécessaire de le préciser). Je vous laisse vous référer à la partie qui concerne la suppression de balise XHTML pour comprendre comment vont fonctionner les *flags* optionnels. La seule différence sera que cette fois, le filtre ne supprimera pas les balises XHTML.

Gérer les Magic Quotes

Les 'magic quotes' sont en fait les guillemets. En PHP les guillemets peuvent permettre d'interpréter des variables, ce qui est très dangereux. Il faut donc ne pas laisser passer une telle faille de sécurité. Pour cela, vous pouvez soit les désactiver de votre serveur PHP, soit les échapper (les rendre inoffensives).

Nous avons vu précédemment qu'il est possible d'encoder les guillemets, mais FilterPHP permet de faire l'équivalent de la fonction addslashes() grâce au filtre FILTER_SANITIZE_MAGIC_QUOTES. Le Listing 32 vous montre l'utilité de ce filtre. Vous remarquez que des antislash ont été ajoutés devant chaque caractère qui doit être échappé.

Le filtre de rappel

Comme son nom l'indique le filtre FILTER_CALLBACK permet d'appeler une fonction de rappel. C'est-à-dire appliquer une fonction à la chaîne de caractères passée en argument à filter_var(). Vous pouvez appeler une fonction ou une méthode de classe. Pour appeler une fonction, la syntaxe est la suivante :

```
filter_var($variable, FILTER_CALLBACK,
array("options" => "Fonction"));
```

Il est à noter que le rappel multiple n'est pas autorisé. `filter_var($variable, FILTER_CALLBACK, array("options" => array('Function1', 'Fonction2')))` retournera une erreur.

Pour appeler une méthode de classe, il faut utiliser : `filter_var($Variable, FILTER_CALLBACK, array("options" =>array("MaClasse", "maMethodeDeClasse")))`. Point important, les exceptions sont retournées par ce filtre. Le Listing 33 est un exemple d'utilisation de ce filtre. La première partie permet de convertir les espaces en tirets afin de formater un numéro de téléphone. La seconde partie appelle une méthode de classe qui permet de tester une adresse e-mail, qui elle-même retourne une exception si l'e-mail n'est pas valide.

Utilisation des variables super-globales et des tableaux

Les variables super-globales sont les variables PHP telles que `$_POST`, `$_COOKIE`, `$_SERVER`,... FilterPHP va nous permettre de contrôler ou nettoyer le contenu de ces variables dès leur arrivée, nous serons donc certains de leur contenu. Les tableaux vont nous permettre de regrouper les tests afin d'éviter d'avoir un code trop lourd.

Comment utiliser les variables super-globales avec FilterPHP ?

L'utilisation des variables super-globales ne se fait pas avec `filter_var()` mais avec `filter_input()`. La syntaxe de cette fonction est assez similaire à ce que nous avons vu jusqu'à présent, en effet, elle est la suivante : `filter_input(type, 'nom_de_la_variable', filtre))`. Filtre est le filtre que vous souhaitez utiliser, donc tous les filtres que nous avons vus dans cet article. Type est une constante parmi : `INPUT_GET`, `INPUT_POST`, `INPUT_COOKIE`, `INPUT_SERVER`,

Figure 3. Résultat du listing 35

Sur Internet

- http://www.aly-abbara.com/utilitaires/convertisseur/convertisseur_chiffres.html – Un convertisseur multiple(hexadécimal, décimal, octal, binaire).
- <http://www.commentcamarche.net/contents/php/phpreg.php3> – page expliquant avec précision les expressions régulières,
- <http://www.siteduzero.com/tutoriel-3-14608-les-expressions-regulieres-partie-1-2.html> – tutoriel créé par un internaute expliquant de façon 'sympathique' les expressions régulières,
- <http://www.whatasciicode.com/> – la correspondance entre symbole, code ASCII, code hexadécimal et le codage HTML des caractères,
- <http://www.php.net/manual/fr/book.filter.php> – page du manuel de FilterPHP.

`INPUT_ENV`, `INPUT_SESSION` (pas encore implémenté) ou `INPUT_REQUEST` (pas encore implémenté). Le Listing 34 est une page nommée `test.php`. On fournit dans l'URL un élément `lien=http://lien 1.php` pour montrer un exemple de l'utilisation de `INPUT_GET`. Dans ce cas, l'intérêt est de pouvoir encoder une URL. L'utilisation des autres types se fait de la même manière.

Exemple concret en utilisant les tableaux

Avant de vous montrer un exemple concret de tout ce que nous avons vu dans cet article, complété par les tableaux, voyons comment combiner les tableaux et FilterPHP. Pour filtrer un tableau, vous devez utiliser `filter_var_array()`. Pour vous présenter cette fonction, voici un exemple simple :

- Déclaration d'un tableau : `$var=array("nom"=>"Jérémie Rafflin!","age"=>"21a");`
- Déclaration d'un tableau de filtre : `$filtre=array("age"=>array('filter'=>FILTER_SANITIZE_NUMBER_INT),"nom"=>array('filter'=>FILTER_SANITIZE_MAGIC_QUOTES));`
- On applique les filtres aux variables : `$tableauFiltre=filter_var_array($var, $filtre);`
- Vous récupérez alors un tableau `$tableauFiltre` qui contient `$tableauFiltre['age']=21` et `$tableauFiltre['nom']=Jérémie \'Rafflin\'`.

Passons désormais à un exemple plus concret. Le Listing 35 comprend un formulaire dans lequel j'ai mis des valeurs par défaut pour exemple. En ce qui concerne le PHP, dans un premier temps, on déclare un tableau qui contient les noms des variables `POST` à récupérer avec les filtres à leur appliquer. Nous allons donc vérifier si l'âge est un entier compris entre 0 et 100 et si l'URL peut être ouverte grâce à la fonction `testUrl()`. Nous allons également assai-

nir les caractères spéciaux présents dans le nom et les caractères qui ne doivent pas être présents dans les e-mails. Vous pouvez voir le résultat sur la Figure 3. L'âge n'est pas affiché car le filtre a retourné `false` au test puisque dans le formulaire l'âge vaut `21a`.

Conclusion

FilterPHP a le mérite d'être vraiment simple à comprendre, à implémenter et propose de nombreux filtres pour contrôler et assainir des variables. Vous devriez maintenant être capable de créer une classe vous permettant de sécuriser vos données entrantes et sortantes. Comme nous l'avons vu, ce n'est pas une extension 'miracle' puisque son exécution est parfois plus lente que d'autres extensions. Souvent en programmation, nous nous retrouvons devant un compromis entre les performances du code et le temps de réalisation des scripts. La plupart du temps, les ressources machines sont négligées face au temps humain pour des raisons économiques. FilterPHP est donc dans ce cas bien adapté puisque l'extension ne nécessite que peu de temps humain pour l'apprendre et l'implémenter. Gardez en tête que FilterPHP est une extension qui est toujours en développement et qui a de beaux jours devant elle. Je vous conseille donc d'aller voir de temps en temps les nouveautés de l'extension si vous l'utilisez.

JÉRÉMY RAFFLIN

L'auteur est étudiant en première année de master. Il est autodidacte en PHP depuis plus de trois ans et s'intéresse particulièrement à la sécurité et à la création d'applications multi-utilisateurs, notamment les jeux vidéos sur navigateur. Il participe au moment de l'écriture de cet article à la création d'un site d'hébergement web.

La gestion du multilinguisme dans le développement web

Le multilinguisme est un sujet très présent dans le développement web. Or, cette notion peut s'avérer bien plus complexe qu'il n'y paraît. Nous allons voir dans cet article quels cas de figure peuvent être rencontrés et quels sont les moyens d'y répondre.

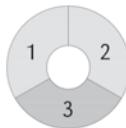
Cet article explique :

- Les problématiques de multilinguisme en développement web et des exemples de techniques de prise en charge.

Ce qu'il faut savoir :

- Les noms de domaine
- Les modèles de base de données
- L'url rewriting
- Le référencement
- L'IOC
- L'ORM Doctrine.

Niveau de difficulté



Le multilinguisme est souvent assimilé à une simple traduction de texte. Or, nous allons voir que la réalité des projets web nous expose à une plus grande complexité. Nous verrons que des variations linguistiques peuvent être appliquées aussi bien aux contenus qu'à la structure et au comportement du site. En guise de premier exemple, imaginons un site contenant des images. Il est très fréquent qu'une image contienne du texte. La localisation du site nous imposera de produire une version *traduite* de l'image. De même, un site peut contenir des vidéos. La localisation de vidéos devra passer soit par un ré-encodage avec doublage dans une autre langue, soit par la mise en place de sous-titres.

La traduction quant-à-elle des textes pourra être sujette à des jeux de caractères exotiques ou à des sens d'écriture différents. Les documents structurés (multi-attributs) affichés sur le site peuvent aussi subir des variations. Ainsi, le classique binôme *FirstName – LastName* dans un pays pourra devenir *FirstName – MiddleName – LastName* dans un autre pays.

Lors du référencement par les moteurs de recherche, les urls des pages sont lues par les

robots. Une même page présente dans plusieurs langues devra donc disposer d'une url spécifique à chaque langue. Il en est de même pour les méta-information de la page comme le *title*, les *meta-description*, *meta-keywords*, ... tout comme les *titles* des hyperliens, les *alt* des images, etc ...

La structure d'une page peut également varier d'un pays à l'autre (instances différentes d'un même site sur des domaines différents). Un site qui vend des produits soumis à des textes de lois locaux va devoir dans certains cas afficher un bloc contextuel d'information. Dans ce cas la localisation de la page revient à afficher ou à masquer certaines zones en fonction du pays d'où elle est consultée. De plus, un même pays peut avoir plusieurs langues. Il en est de même pour les processus métier. Le tunnel d'achat d'un site eCommerce peut comporter un nombre d'étapes différent d'un pays à un autre.

La structure globale d'un site peut également varier d'un pays à un autre. Une page d'informations légales peut être ajoutée sur seulement certaines versions pays d'un site. En plus de la traduction et des variations de structure et de processus, les données de la base peuvent être filtrées différemment d'une localisation à une autre. Sur le site corporatif d'une entreprise ayant des bureaux dans différents pays, on peut imaginer une fenêtre de *News* sur la page d'accueil qui affiche des informations locales au bureau du

pays. Dans ce cas, il ne s'agit pas d'avoir une traduction des news mais bien d'avoir une liste de news différente pour chaque pays. Cela revient à afficher des lignes différentes d'une même table en fonction du pays qui demande la page.

Les quelques exemples que nous venons de voir concernent le *front-office* des sites web, mais on peut très bien imaginer des besoins de localisation au niveau du *back-office* ou du CMS.

Notons que tous les CMS (Open Source ou commerciaux) ne sont pas égaux devant le multilinguisme. Nous ne rentrerons pas dans le détail de l'implémentation des fonctionnalités mais dresserons dans cet article un aperçu des principaux enjeux de la discipline. Nous laisserons au lecteur la liberté d'évaluer les capacités natives de son CMS favori.

Les principaux enjeux du multilinguisme

Nous voyons dans ce paragraphe les principaux enjeux du multilinguisme. Il est assez rare qu'un site doivent assumer tous ces enjeux, mais nous avons ici une couverture assez complète de l'univers des possibles.

Proposer une liste de langues

La première responsabilité d'un site multilingue est de nous proposer la liste des langues supportées. Cette liste peut être proposée sous forme d'une série de petits drapeaux, d'une liste déroulante (*combo*), de radio-boutons, ... l'ergonomie étant à adapter en fonction du nombre de langues et du look'n feel du site. Il sera parfois plus pratique de proposer une liste déroulante pour un site offrant un grand nombre de langues, plutôt que d'afficher la série de drapeaux qui occuperait une place conséquente sur la page.

De plus, à la sélection de la langue, celle-ci devra être persistante afin que l'internaute la

conserve durant la navigation. Il faudra également qu'elle soit reconnue à chaque fois qu'il revient sur le site. Les *cookies* et la *session serveur* sont utilisés pour accomplir cette tâche. La majorité des CMS actuels l'encapsulent correctement et cette gestion est quasi-transparente pour le développeur.

DéTECTER LA LANGUE DE L'INTERNAUTE

Lorsqu'un internaute arrive sur un site, le serveur doit être capable de lire des informations relatives à ses préférences ou caractéristiques linguistiques. Si l'internaute n'est jamais venu sur ce site, le système doit être capable de proposer une langue par défaut. Un bon candidat est alors la langue utilisée pour le navigateur.

De manière générale, il y a plusieurs façons de récupérer un paramètre de langue depuis le serveur. La première technique revient à lire une variable de cookie qui stocke la préférence de l'utilisateur. Il est également possible d'utiliser une variable de session qui maintient la langue sur chaque page.

Mais la méthode qui reste la plus efficace revient à modifier les URL pour passer un paramètre supplémentaire décrivant la langue. Cela permet de gérer très facilement le contenu de la page et fournit une adresse différente pour chaque langue. L'avantage ici est que les moteurs de recherche pourront référencer chaque langue différemment, ce qui est plus optimal.

GÉRER LES LABELS

Après avoir lu le paramètre de langue de l'internaute, il faut lui délivrer le contenu associé. Le premier niveau de localisation d'un site revient à utiliser la notion de label. L'idée est de ne pas écrire d'expressions linguistiques directement dans le HTML, mais plutôt de les référencer par des codes.

Les mots de vocabulaire propres à chaque langue seront par exemple stockés dans un fichier *langfr.inc* ou *langen.inc* ou dans une base de données si les volumes sont importants. Le moteur de cache jouera ici un rôle fondamental pour minimiser les IO ou le nombre de requêtes SQL. Le piège étant dans ce genre de situation de ne pas penser aux accès faits par le serveur pour remonter les valeurs des labels, ce qui peut fortement pénaliser les performances. Pour gérer les labels, le code PHP contiendra ainsi les appels à une fonction qui prend en paramètre le code du label et la langue et retourne l'expression linguistique associée.

GÉRER LES ÉLÉMENTS EN BASE DE DONNÉES

Les données métier d'un site web sont stockées dans une base de données modelée spéci-

Listing 1. Exemple de récupération de la langue du navigateur dans la variable *selectedLanguage* parmi une liste prédéfinie de possibilités

```
$selectedLanguage = 'fr';
$languages = array('fr', 'en', 'es');
if(!empty($_SERVER['HTTP_ACCEPT_LANGUAGE'])){
    $browserLanguages = explode(',', $_SERVER['HTTP_ACCEPT_LANGUAGE']);
    foreach($browserLanguages as $browserLanguage) {
        $lang = strtolower(substr($browserLanguage, 0, 2));
        if(in_array($lang, $languages)) {
            $selectedLanguage = $lang;
        }
    }
}
```

fiquement pour le site. Les tables contiennent des colonnes qui représentent les attributs des objets métier. Ainsi, un site eCommerce pourra afficher des produits avec des attributs localisables et des attributs non localisables.

Typiquement, le numéro de référence d'un produit est universel alors que sa description doit être traduite dans chaque langue.

Ceci est différent d'une gestion de labels. Les labels représentent une notion de contenu non structuré comme les textes sur les boutons, les messages de bienvenue, ... alors que les éléments en base de données sont sujets à des contraintes d'intégrité référentielle et à des volumes qui peuvent atteindre plusieurs millions de lignes. Pour gérer correctement ce cas de figure, nous pouvons faire appel au pattern translation. Le pattern translation s'appuie sur l'éclatement d'un objet métier en deux tables en base de données : la table des informations non localisables (dont la clé primaire fait partie) et la table des informations localisables. La Figure 1 illustre cette architecture.

Dans ce cas, pour un produit affiché sur un site en 4 langues, nous aurons une ligne dans la table principale du produit qui contiendra la clé primaire, la référence produit, ainsi que d'autres caractéristiques non localisables, puis 4 lignes dans la table translation qui contiendront chacune les informa-

tions traduites du produit dans chacune des 4 langues. L'architecture applicative devra encapsuler cette fonctionnalité avec une API prenant en paramètre l'identifiant du produit et la langue dans laquelle on veut l'information, exactement sur le même principe que l'accès à la valeur d'un *label*.

GESTION DU CROISEMENT PAYS / LANGUE

Il est important de faire la différence entre une version linguistique et une version pays d'un site.

Une version pays pourra contenir plusieurs version linguistiques et chaque version pays sera sur un nom de domaine propre. Comme vu précédemment, les déclinaisons linguistiques sont basées sur la détection côté serveur des préférences de langue de l'internaute. Il s'en suit un affichage des labels et des données dans la langue sélectionnée.

Tout ceci se passe sur un seul et même nom de domaine. Par exemple, toutes les requêtes vont converger vers un domaine *www.mon-site.com* et vont se différencier par la variable *lang=en* ou *lang=fr*, ...

Or, de nombreux sites doivent offrir plus qu'une traduction de labels et de données, et doivent adapter leurs templates et leurs traitements à des caractéristiques pays qui leurs sont propres. C'est souvent le cas pour

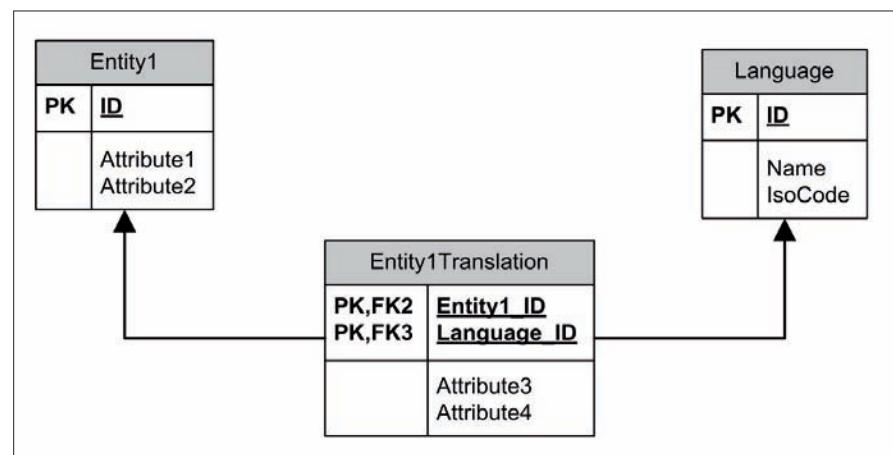


Figure 1. Architecture physique de localisation

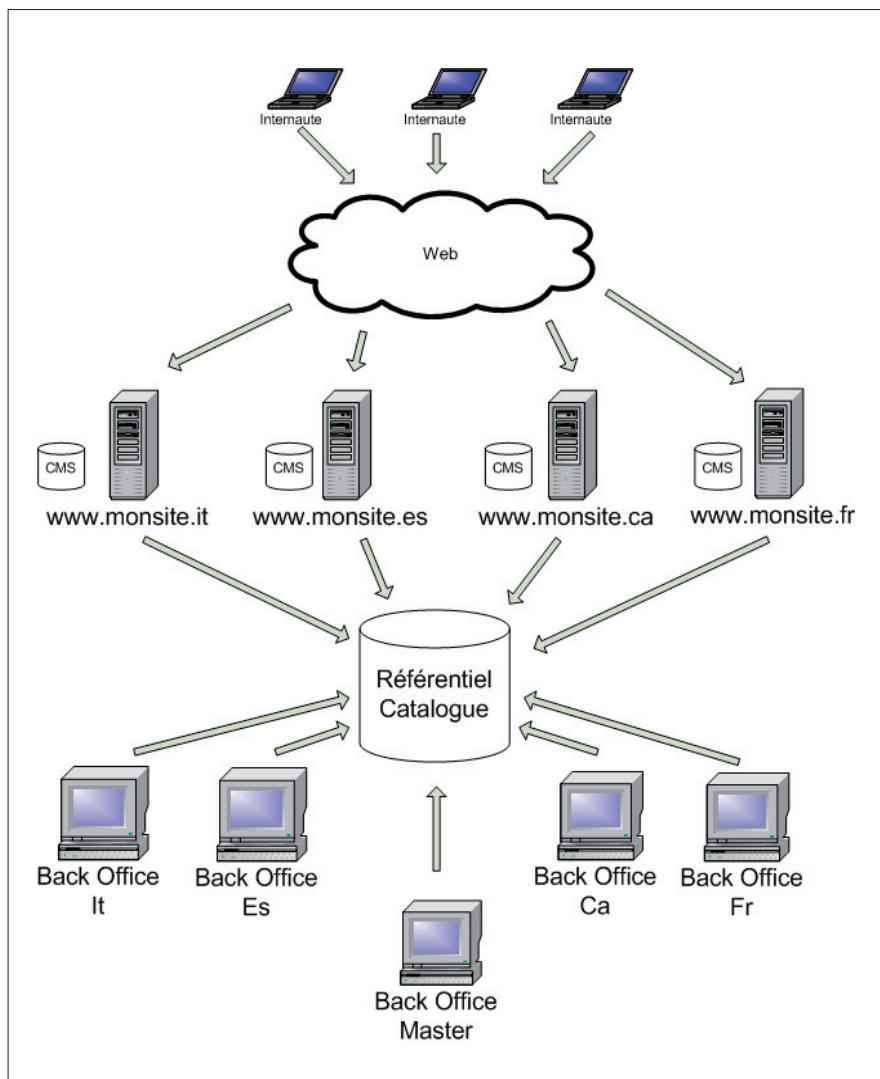


Figure 2. Architecture web d'un site multi-pays

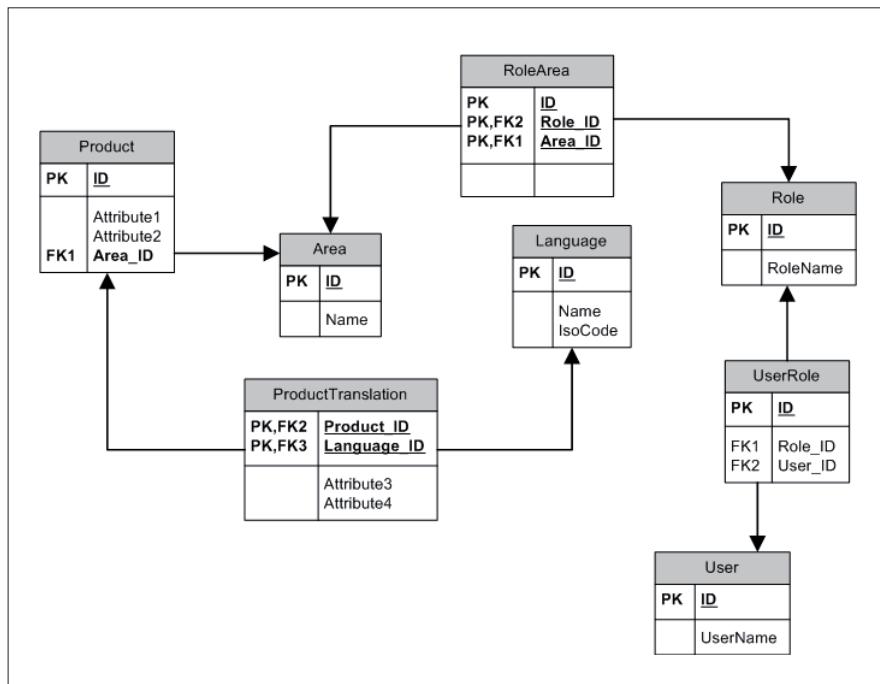


Figure 3. Architecture de base de données centrale permettant la localisation linguistique et la localisation pays

les sites de grandes marques implantées dans de nombreux pays sur plusieurs continents. Dans ce cas, l'approche revient parfois à avoir une *page dispatch* qui sera le point d'entrée des différentes versions du site.

Imaginons que notre site *www.monsite.com* possède une localisation italienne, une localisation espagnole, une localisation canadienne et une localisation française. Nous allons devoir créer 4 domaines qui seront respectivement *www.monsite.it*, *www.monsite.es*, *www.monsite.ca* et *www.monsite.fr*.

A chaque domaine correspondra une instance du site. Mais chaque instance du site pourra elle-même posséder plusieurs variantes linguistiques. On peut par exemple imaginer une version française et une version anglaise de la localisation pays *www.monsite.ca*. De cette façon, le site canadien répondra des pages en français si on le consulte par *http://www.monsite.ca?lang=fr* alors qu'il répondra des pages en anglais si on le consulte par *http://www.monsite.ca?lang=en*.

De plus, si *www.monsite.com* propose un catalogue de produits, il est judicieux de pouvoir l'administrer à partir d'un seul endroit pour la création des références produit, mais de pouvoir l'administrer localement pour les déclinaisons linguistiques de ces références.

La Figure 2 illustre cette organisation.

Selon cette architecture, chaque version pays du site, de par la séparation physique des instances web, peut proposer ses propres templates de présentation. Cette approche doit néanmoins être la plus factorisée possible de façon à ne jamais dupliquer 2 fois le même traitement. Nous rentrerons plus en détail sur cette stratégie un peu plus bas. Ce mode d'organisation nous permet d'administrer localement les contenus éditoriaux (via les fonctions CMS), d'administrer localement les traductions des produits du catalogue, et d'avoir une administration centrale du référentiel catalogue. L'efficacité de ce dispositif réside dans la mise en place d'un système de filtres par pays sur la BDD centrale. C'est ce que nous détaillons dans le paragraphe suivant.

Gérer les filtres sur les données

Comme nous venons de le voir, une gestion efficace d'un site multi-pays passe par l'utilisation d'une base centrale (en plus des bases locales). Cette base centrale permet d'administrer depuis un point unique les éléments structurants (création du référentiel métier) et autorise l'administration locale des déclinaisons linguistiques du référentiel.

Pour ce faire, nous mettons en place le pattern *Data Visibility* sur la base de données.

Ce pattern consiste à associer chaque données à une *Area*. Une *Area* est une zone de regroupement sur laquelle les administrateurs ont des droits de lecture et de modification paramétrables. Concrètement, cela revient à avoir une table nommée *Area* vers laquelle toutes les autres tables ont une clé étrangère. A chaque entrée dans cette table va correspondre une *région* de la base de données. En fonction des droits, lorsque le système interroge la base, l'*area* du demandeur est passée en paramètre pour filtrer les données retournées.

Dans notre exemple à 4 pays, nous devons insérer 5 lignes dans cette table : l'*area Italia*, l'*area Espagne*, l'*area Canada*, l'*area France* et l'*area All* (l'*area All* servira à identifier les objets visibles par tout le monde).

Les produits visibles pour seulement l'Italie auront un lien vers l'*area Italia*. Les produits visibles par tous les pays auront un lien vers l'*area All*. Si des produits doivent être visibles pour seulement la France et l'Italie, alors on créera une area *France/Italie* et les produits concernés auront un lien vers cette area. La simple gestion du paramètre *area* permet ainsi de filtrer systématiquement au niveau d'une couche DAO les données à remonter pour un pays. Cela repose sur la génération dynamique d'une restriction SQL liée à l'*area*. De cette manière, un produit peut être partagé ou non entre différents pays, et, par pays, peut subir des déclinaisons linguistiques. La Figure 3 illustre le modèle de base de données qui permet une telle gestion.

Que la base de données soit accédée depuis un serveur web (*front-office*) ou depuis un *back-office*, le système appelant doit passer 2 informations : l'*area* courante et la langue courante. Les API de la couche DAO ont la responsabilité de n'autoriser l'accès aux données que si ces 2 informations sont communiquées.

Cinématique : un user (*front* ou *back*) possède un ou plusieurs rôles (dont le rôle anonyme pour l'internaute non logué). Chaque rôle possède une ou plusieurs *area*(s) qui lui donne(nt) son espace de visibilité sur les données. Chaque produit est stocké dans une *area* via sa clé étrangère vers la table *Area*. Chaque attribut localisable du produit est stocké dans la table translation correspondante. L'architecture multi-pays multi-langues est donc posée : les données remontées de la base sont obtenues par une réunion ensembliste de droits sur les areas, croisée avec un éventuel paramètre de langue. Toutes les combinaisons pays/langue sont alors possibles avec cette architecture. En ajoutant un écran d'administration des areas dans le *back-office* central, nous obtenons un système souple et puissant.

Listing 2. Initialisation de la langue et du pays

```
// Fonction qui initialise la langue et le pays (market) - à implémenter
dans un plugin Joomla par exemple
public function initLanguageAndMarket()
{
    // Identification du pays (market) et de la langue si absents de la
    session
    if (empty($_SESSION['market']) || empty($_SESSION['lang'])) {
        // Détection de la langue du browser
        $acceptLang = explode(",", strtolower($_SERVER['HTTP_ACCEPT_LANGUAGE']));
        $prefLang = explode("-", $acceptLang[0]);

        if (empty($prefLang[1])) $prefCtry = $prefLang[0]; else $prefCtry
        = $prefLang[1];
        $prefLang = $prefLang[0];

        // Ecriture des valeurs en session
        $_SESSION['market'] = $prefCtry;
        $_SESSION['lang'] = $prefLang;
    }

    // Récupérer la langue des préférences de l'utilisateur passées en
    paramètre de la requête
    if (!empty($_GET['lang'])) {
        $_SESSION['lang'] = $_GET['lang'];
    }

    // Si le pays n'est pas défini, on le positionne par défaut à "int"
    (international)
    if (!empty($_SESSION['market'])) $market = $_SESSION['market']; else
$market = "int";

    // Si la langue n'est pas définie, on la positionne à "en" (anglais)
    if (!empty($_SESSION['lang'])) $lang = $_SESSION['lang']; else $lang
= "en";

    // On teste si l'area existe
    $query = Doctrine_Query::create()
        ->from('Area a')
        ->where('a.code = ?', strtoupper($market));

    if($query->count() == 0)
        $market = $_SESSION['market'] = "int";

    // On teste si la langue existe pour l'area courante
    $query = Doctrine_Query::create()
        ->from('Language l')
        ->innerJoin('RoleArea ar')
        ->innerJoin('Area a')
        ->where('l.isocode = ?', strtoupper($lang))
        ->andWhere('a.code = ?', strtoupper($market));

    if($query->count() == 0)
    {
        $query = Doctrine_Query::create()
            ->select('a.defaultLanguage')
            ->from('Area a')
            ->where('a.code = ?', strtoupper($market));

        // Récupération de la langue par défaut de l'area
        $lang = $_SESSION['lang'] = $query->execute()[0]->defaultLanguage;
    }
    LanguageHelper::init($lang, $market);
}
```

Listing 3. Accès à la langue et au pays

```
// Accès à la langue
LanguageHelper::getCurrentLanguage();

// Accès au pays
LanguageHelper::getCurrentMarket();
```

Listing 4. Lecture de données localisée

```
$query = Doctrine_Query::create()
    ->from('Entity e')
    ->innerJoin('e.Translation et')
    ->where('et.lang = ?', strtoupper($lang))
    ->andWhereIn('e.area', array ($market, 'int'));
```

Sur Internet

- <http://www.webrankinfo.com/> – Communauté francophone du référencement,
- http://fr.wikipedia.org/wiki/Nom_de_domaine – Explications sur les noms de domaines,
- <http://apache.developpez.com/cours/urlrewriting/> – Cours sur la réécriture d'url,
- <http://sqlpro.developpez.com/cours/modelisation/merise/> – Cours sur la modélisation de bases de données,
- <http://www.w3.org/International/O-charset.en.php> – Informations liées à l'encodage de caractères,
- <http://www.w3.org/International/questions/qa-htaccess-charset> – Paramétriser les informations de charsets depuis le fichier .htaccess
<http://martinfowler.com/articles/injection.html> – Inversion de contrôle

blématiques de traduction de labels, de traduction de contenus, de gestion de filtres sur les données, de mutualisation de bases de données, et de réécriture d'url. Il reste un point important qui n'a pas encore été évoqué. Ce point concerne la variation de template (présentation), de structure de site (arborescence), et de processus métier.

C'est ici qu'apparaît la combinatoire la plus complexe.

Une règle universelle doit être respectée : factoriser au maximum. En effet, il n'est pas rare que certains sites se déclinent sur plus de 15 langues. Si nous ne pratiquons pas un bon niveau de factorisation du code, n'importe quelle évolution fonctionnelle commune à toutes les langues peut devenir un vrai cauchemar. En effet, le risque est que chaque correction ou évolution nécessite le report de 15 fois le même bloc de code. Ceci est bien-sûr à proscrire, sans quoi aucune TMA n'est plus envisageable. Plusieurs techniques peuvent nous aider à gérer ces problématiques.

Cas n°1

Le premier cas de figure traite d'une instance unique du site (déclinaisons linguistiques mono-site).

Dans ce cas il est conseillé de créer des templates et processus dont le comportement est conditionné par le paramètre langue. Sans rentrer dans le détail d'architectures applicatives (ce qui nécessiterait un article à part entière), nous mentionnons juste ici que l'idée est de mettre en place des providers d'HTML et de règles (classes statiques exposant des librairies) qui centralisent la gestion des nuances linguistiques. Une page pourra ainsi demander son template au provider qui répondra par exemple en mode 2 colonnes si la langue est *fr* ou en mode 3 colonnes si la langue est *it*. Il en est de même pour un processus métier qui demanderait une liste de devises pour le paiement : la fonction qui retourne les devises prendra en paramètre la langue et retournera la liste correspondante.

Cas n°2

Le deuxième cas de figure traite de plusieurs instances de site installées sur des domaines différents.

Dans ce cas, la démarche consiste à créer un unique site contenant l'ensemble des comportements possibles et dont le comportement local est piloté par fichier de configuration. On parle ici d'un site tronc commun configurable et possédant des pages satellites pouvant être utilisées spécifiquement en fonction du pays. Une approche IOC (*Inversion Of Control*) s'avérera pratique pour remplacer une page standard par une page spé-

Centraliser l'accès aux informations de langue et de pays

De nombreuses classes doivent pouvoir accéder aux informations de langue et de pays pour conditionner le comportement du site. Il est donc important de les initialiser proprement et centraliser l'accès à ces valeurs.

Si l'on travaille avec Joomla!, l'utilisation d'un plugin pour l'initialisation des valeurs est un bon choix.

Le Listing 2 illustre le code qui initialise la langue et le pays avant de les exposer à l'ensemble du code du site.

L'initialisation alimente un LanguageHelper qui expose la langue et le pays via 2 accessseurs statiques.

Le Listing 3 montre la syntaxe de récupération de la langue et du pays depuis n'importe quelle partie du code.

Enfin, le Listing 4 donne un exemple de requête Doctrine qui récupère des données localisées par le pattern Translation.

Assurer le référencement

Il est fortement recommandé, pour un référencement optimisé, de disposer d'une url spécifique pour chaque page de chaque localisation. Ainsi, la page de détail d'un produit devra disposer d'une url propre réécrite.

Sur des variations pays, le changement de domaine va apporter le premier niveau de différenciation dans l'url. La caractérisation du produit apportera le deuxième niveau de différenciation avec des informations liées à son nom, son code, ... Ensuite, pour les variations-

linguistiques sur le même domaine, il est conseillé de faire apparaître explicitement la langue dans l'url. La plupart des CMS gèrent nativement l'url rewriting. Attention cependant à la souplesse de ceux-ci, notamment par rapport à la liberté de mise en forme des urls réécrites.

En plus de la réécriture d'url, un bon référencement passe aussi par un HTML qualitatif. Il faudra donc s'attacher à ce que les métainformations d'une page (title, meta-description, meta-keywords, ...) soient déclinées par localisation. Si le CMS ne propose pas nativement ce type d'administration, il faudra prévoir le développement de cette fonctionnalité. Les liens et les médias devront également être localisables depuis le CMS. Ainsi, les *alt* des images, les *titles* des liens, etc ... devront être renseignés pour chaque localisation.

Il est également fondamental que les urls des hyperliens soient écrites en version friendly dans le code html de la page. Les robots Google exploitent cette version réécrite lorsqu'ils crawlent le site. Le mécanisme de publication des pages devra donc prendre en compte cette contrainte. Pour plus de détail sur un référencement optimal, nous renvoyons le lecteur vers les guidelines Webrankinfo (<http://www.webrankinfo.com>).

Gérer les mises en pages, structures de site et processus spécifiques : un rôle pour l'IOC

Nous avons vu jusqu'à présent que les localisations langues ou pays soulevaient des pro-

Terminologie

- Localisation : Déclinaison linguistique ou pays d'un site web.
 - Url Rewriting : Remplacement des urls techniques d'un site par des friendly url permettant à Google une meilleure indexation.
 - Pattern Data Visibility : Structuration des tables et relations d'une base de données pour filtrer les données en fonction de droits de visibilité.
 - TMA : Tierce Maintenance Applicative. Il s'agit des actions de maintenance corrective et évolutive qui peuvent être réalisées après la mise en production d'un site.
- DAO : Data Access Object ou couche d'accès aux données.
IOC : Inversion Of Control ou technique d'injection de dépendance par configuration.

Rejoignez le Club .PRO

Pour plus de renseignement : editor@phpsolmag.org

cifique à un pays. Il en est de même pour les classes implémentant la logique métier. Le site est alors déployé à l'identique sur tous les domaines et varie seulement par ses fichiers de configuration.

Gérer les jeux de caractères

Enfin, n'oublions pas que les jeux de caractères peuvent varier fortement d'une langue à une autre. Il est très important de toujours libeller les pages Web de manière explicite. La norme HTTP 1.1 indique que le codage par défaut est ISO-8859-1. Voici un exemple de ligne typique présente dans l'entête HTTP :

```
Content-Type: text/html; charset=utf-8
```

Avant d'avoir renvoyé du contenu, on peut par exemple utiliser la fonction header() :

```
header('Content-type: text/html; charset=utf-8');
```

Le fichier *.htaccess* d'Apache permet aussi de préciser les encodages spécifiques des pages servies. Dans la majorité des cas, le format UTF-8 sera à privilégier.

Conclusion

Nous avons vu dans cet article que les problématiques multilingues peuvent nécessiter parfois la mise en place d'architectures complexes. Même si les cas les plus simples se résument à de la traduction de labels, une vraie réflexion est nécessaire au niveau base de données si l'on veut garder un maximum de souplesse. Il est important de planifier ce type de méthode avant de commencer tout développement. La question initiale étant d'identifier quel sera l'impact du multilinguisme sur le comportement global du site.

Dans tous les cas, il faudra résister à la tentation de dupliquer simplement un site pour le traduire car cette action génère de la redondance et complexifie les évolutions post-localisations. La règle principale sera donc de toujours viser un niveau de factorisation le plus haut possible pour ne pas s'enliser dans les maintenances ultérieures.

CHRISTOPHE CADIC

Christophe Cadic est Directeur Technique France de l'agence de marketing interactif Nurun. Il rencontre régulièrement les problématiques du multilinguisme lors de la mise en place de dispositifs digitaux pour ses clients et partenaires.

Contact : christophe.cadic@nurun.com



Stonfield Inworld

Stonfield Inworld propose aux entreprises des solutions globale d'intégration d'Internet et des Univers Virtuels dans leur stratégie de développement. Au-delà de ses services, la société consacre 30% de ses ressources à des travaux de R&D sur le e-Commerce et le e-Learning dans les Mondes Virtuels.



COGNIX Systems

Conseil, conception et développement d'applications évoluées pour les systèmes d'informations Internet/intranet/extranet. Alliant les compétences d'une SSII et d'une Web Agency, Cognix Systems conçoit des applicatifs et portails web à l'ergonomie travaillée et des sites Internet à forte valeur ajoutée.
<http://www.cognix-systems.com>



Alter Way GROUP

Anaska Formation

Anaska est le spécialiste des formations sur les technologies OpenSource. En partenariat avec MySQL AB, Mandriva, Zend et d'autres acteurs de la communauté, Anaska vous propose un catalogue de plus de 50 formations dédiés aux technologies du Libre.
<http://www.anaska.com>



WEB82

Création et hébergements de sites web pour particuliers, associations, entreprises, e-commerce. Développement entièrement aux normes W3C (www.w3.org) de sites web de qualité, au graphisme soigné et employant les dernières technologies du web (PHP5, MySQL5, Ajax, XHTML, CSS2).
<http://www.web82.net>



Core-Techs

Expert des solutions de gestion et de communication d'entreprise en Open Source, Core-Techs conçoit, intègre, déploie et maintient des systèmes de Gestion de Contenu Web, de Gestion Documentaire, de Gestion de la Relation Client (CRM), d'e-commerce et de travail collaboratif.
<http://www.core-techs.fr>



POP FACTORY

Pop Factory, SSII spécialisée Web. Développement de solutions applicatives spécifiques ; offre de solutions packagées : catalogue numérique, e-commerce, livre/magazine numérique, envoi SMS. Nous accompagnons nos clients tout au long de leur projet : audit, conseil, développement, suivi et gestion.
<http://www.popfactory.com/info@popfactory.fr>



Blue Note Systems

Spécialistes en CRM Open Source, nous proposons une offre complète de prestations sur la solution SugarCRM. Notre valeur ajoutée réside dans une expertise réactive et une expérience des problématiques de la GRC. Nous vous aidons à tirer le meilleur parti de votre solution CRM.
<http://www.bluenote-systems.com>



Intelligence Power

Conseil, Expertises, Formations et Projets E-business centrés au tour du cœur de métier : la Business Intelligence. Intelligence Power vous propose des solutions innovantes pour aligner la technologie sur la stratégie de votre entreprise.
<http://www.intelligencepower.com>



Web Alliance

Vous souhaitez être en première page des moteurs de recherche ? Rejoignez-nous, 100% des clients Web Alliance sont en 1ère page de Google. Web Alliance, société de conseil spécialisée dans le référencement internet, vous propose son expertise (référencement, liens sponsorisés, web-marketing).
www.web-alliance.fr

Club.PRO

Découvrez SimpleXml

Le XML est un langage informatique dit de balisage générique. Il est principalement utilisé pour structurer des données. Voyons ensemble comment utiliser la librairie SimpleXml afin de créer et lire des documents XML assez simplement.

Cet article explique :

- Comment utiliser la librairie SimpleXml.
- Un lecteur de flux RSS.

Ce qu'il faut savoir :

- Bases de PHP.
- Des connaissances en Xml sont un plus.

Niveau de difficulté



Commençons par un exemple simple : Le Listing 1 présente un fichier Xml appelé famille.xml. Comme vous pouvez le voir, un fichier xml possède une structure bien définie en suivant plusieurs règles. Le Tableau 1 rassemble quelques unes de ces règles. L'on pourrait ajouter d'autres règles à respecter lorsqu'on construit un fichier Xml.

Revenons à notre Listing 1. Comme l'on peut tout de suite le remarquer, la simplicité du Xml fait qu'un humain comprend facilement le contenu du fichier. Ainsi l'on retrouve les membres principaux de la famille *Simpson* caractérisés par plusieurs données : *Nom*, *prénom*, *âge* pour tous et nom de jeune fille pour *Marge* uniquement. Il est évident que la liste aurait pu être bien plus longue mais celle-ci sera suffisante à notre étude.

Débutons SimpleXml

La première chose à vérifier, c'est que SimpleXml est activée dans php.ini. Ensuite, nous pourrons écrire un petit script qui aura pour objet d'ouvrir le fichier Xml, et de lister les membres de la famille en affichant seulement le nom et le prénom. C'est ce qui est réalisé avec le code du Listing 2. La première ligne de code charge le contenu du fichier *famille.xml* dans la variable \$personnes. La

suite est on ne peut plus simple : pour chaque personne présente, on affiche le nom et le prénom. Il est possible que le document Xml soit mal construit et que la lecture du fichier Xml ne soit pas possible. Pour éviter des erreurs, dès la lecture du fichier Xml, la fonc-

tion `simplexml_load_file()` renverra `False` si elle détecte un problème de lecture. Ainsi, vous pouvez inclure la boucle `foreach()` dans un `if` comme fait dans le code du Listing 3. Comme vous pouvez le voir, cela risque de poser des soucis dans la localisation des erreurs : `simplexml_load_file()` s'occupe de l'ouverture du fichier, de l'analyse XML ainsi que la création de la ressource. Une seule fonction pour gérer tout cela et dont les sources d'erreurs sont énormes ce qui peut générer des affichages d'erreurs indésirables. Il existe la méthode bien connue du `@` devant la fonction mais cela ne réglera pas les soucis

Listing 1. Une petite famille: famille.xml

```
<?xml version="1.0" encoding="utf-8"?>
<personnes>
    <personne>
        <nom>Simpson</nom>
        <prenom>Homer</prenom>
        <age>40</age>
    </personne>
    <personne>
        <nom>Simpson</nom>
        <prenom>Marge</prenom>
        <nom>JeuneFille</nom>
        <nom>Bouvier</nom>
        <age>41</age>
    </personne>
    <personne>
        <nom>Simpson</nom>
        <prenom>Bart</prenom>
        <age>10</age>
    </personne>
    <personne>
        <nom>Simpson</nom>
        <prenom>Lisa</prenom>
        <age>8</age>
    </personne>
    <personne>
        <nom>Simpson</nom>
        <prenom>Maggie</prenom>
        <age>1</age>
    </personne>
    <personne>
        <nom>Simpson</nom>
        <prenom>Abraham</prenom>
        <age>83</age>
    </personne>
</personnes>
```

Tableau 1. Règles à respecter en Xml

Règle 1	Un document Xml commence toujours par une déclaration XML qui précise la version de la norme XML utilisée. Exemple : <?xml version="1.0" encoding="utf-8"?> L'encodage peut y être précis. Par défaut, il est supposé que le fichier est encodé en UTF-8.
Règle 2	Un fichier Xml est constitué de balises contenant des informations : <balise>information</balise>
Règle 3	Une information est encadrée par une balise ouvrante et une balise fermante qui possède un slash : <balise>info</balise>
Règle 4	Un fichier Xml peut contenir plusieurs éléments et donc plusieurs balises.
Règle 5	L'ordre d'ouverture des balises doit être respecté. Le nom de la balise de fin d'un élément doit correspondre à celui de la balise de début.
Règle 6	Une balise seule se termine par un slash : <balise/>
Règle 7	Les noms des balises peuvent comporter des lettres, des chiffres, des tirets, des underscores, des deux-points ou des points.
Règle 8	Une balise peut contenir des attributs contenant eux mêmes des informations : <balise attribut="valeur1" attribut2="valeur2">information</balise>
Règles 9	Du commentaire peut être inséré entre des balises : le commentaire est encadré de cette manière : <!-- commentaire -->
Règle 10	Des sections CDATA permettent de ne pas traiter les blocs de texte comportant des caractères qui seraient normalement identifiés comme du balisage. Les sections CDATA commencent par "<![CDATA[" et se terminent par la chaîne "]]>". <exemple> <![CDATA[<truc>info&cie<<<]> </exemple>

et vous permettra encore moins de gérer ces erreurs. Il y a donc deux cas possibles : le fichier XML est valide, ou ne l'est pas.

Puisons nos informations

Nous partons désormais dans le cas où le fichier Xml est supposé valide. Il nous reste maintenant la possibilité de puiser les informations nécessaires dans le contenu retourné par la fonction `simplexml_load_file()`. Là où SimpleXml simplifie les choses, c'est que le contenu retourné est à la fois un tableau et un objet dont l'arborescence ressemble à la structure du fichier Xml. Le Listing 4 présente un fichier Xml qui contient plusieurs personnes regroupées en 2 familles : la famille *Simpson* et la famille *Flanders*. Le Listing 5 permet d'afficher avec un `print_r`, à la manière d'un tableau, l'ensemble d'une famille. Dans l'exemple, nous avons affiché la seconde famille. Nous vous rappelons que l'on commence à compter à partir de 0 et non 1. La famille *Simpson* serait donc la *famille[0]*.

Engageons nous dans la recherche avec Xpath

Là où SimpleXml simplifie le traitement du Xml, c'est qu'il fonctionne comme si les balises étaient des dossiers ou fichiers. Ainsi, une balise qui contient une ou plusieurs autres balises est un dossier sinon c'est un fichier. L'on va continuer notre exercice sur le Xml du Listing 4 appelé *famille2.xml*. Nous voudrions lister toutes les personnes. Pour cela nous allons travailler avec Xpath. Une personne est contenue dans une famille elle même contenue dans la liste des personnes. Une architecture dossiers donnerait ceci : */personnes/famille/personne*. Le Listing 6 est un exemple

simple pour réaliser la recherche. Le résultat retourné est un tableau d'objet de type personne. A titre informatif, les personnes sont listées suivant l'ordre de lecture et les balises *famille* ne contenant rien sont bien entendu ignorées. Cependant elle ignorera les personnes qui ne sont pas contenues dans une architecture qui n'est pas */personnes/famille*. Supposons maintenant que nous nous aurions un Xml dénaturalisé avec des personnes qui ne sont pas incluses dans des familles. (le vendeur de Bds ferait l'affaire). Via le code précédent, il n'aurait pas pu être trouvé. Nous avons donc besoin de rechercher simplement toutes les *<personne>*. Au lieu de spécifier une architecture du type : */dossier/sous-dossier/ele-*

ment, nous n'avons qu'à donner le nom de la balise à prendre : *//element*. Le Listing 7 présente cette utilisation.

Lecture d'un flux RSS : Le Monde

Les flux rss proposent des informations dans un fichier xml mis à jour fréquemment. Ainsi de grands sites (tels que les quotidiens) proposent ces flux. Nous allons étudier la lecture d'un flux RSS distant avec l'utilisation de SimpleXml. Le Listing 8 démontre la structure d'un fichier RSS.

La première ligne est la ligne de déclaration `xml`. Ensuite nous avons l'ouverture d'une balise RSS qui spécifie la version à laquelle le document RSS est conforme. Puis, se trouve une

Listing 2. Comment lister les membres de la famille

```
<?php
$personnes = simplexml_load_file('famille.xml');
foreach ($personnes->personne as $personne)
{
    //print_r($personne);
    echo "Nom : ".$personne->nom;
    echo " Prenom : ".$personne->prenom." <br />";
}
?>
```

Listing 3. Gestion d'une éventuelle erreur

```
<?php
$personnes = @simplexml_load_file('famille.xml');
if($personnes)
{
    foreach ($personnes->personne as $personne) {
        print "Nom : {$personne->nom} ";
        print "Prenom : {$personne->prenom} <br />\n";
    }
}
else
{
    echo 'il semble que le fichier Xml soit incorrect';
}
?>
```

Listing 4. Un Xml de familles

```
<?xml version="1.0" encoding="utf-8"?>
<personnes>
    <famille nom="Simpson">
        <personne>
            <prenom>Homer</prenom>
            <age>40</age>
        </personne>
        <personne>
            <prenom>Marge</prenom>
            <nomJeuneFille>Bouvier</nomJeuneFille>
            <age>41</age>
        </personne>
        <personne>
            <prenom>Bart</prenom>
            <age>10</age>
        </personne>
        <personne>
            <prenom>Lisa</prenom>
            <age>8</age>
        </personne>
        <personne>
            <prenom>Maggie</prenom>
            <age>1</age>
        </personne>
        <personne>
            <prenom>Abraham</prenom>
            <age>83</age>
        </personne>
    </famille>
    <famille nom="Flanders">
        <personne>
            <prenom>Ned</prenom>
            <age>60</age>
        </personne>
        <personne>
            <prenom>Maude</prenom>
        </personne>
        <personne>
            <prenom>Todd</prenom>
            <age>8</age>
        </personne>
        <personne>
            <prenom>Rod</prenom>
            <age>10</age>
        </personne>
    </famille>
</personnes>
```

Listing 5. Affichage de la seconde famille

```
<?php

$personnes = simplexml_load_file('famille2.xml');
foreach($personnes->famille[1] as $familles)
{
    echo print_r($familles). "<br>";
}
?>
```

Listing 6. Recherche de toutes les personnes dans '/personnes/famille'

```
<?php

$personnes = simplexml_load_file('famille2.xml');

// on recherche <personnes><famille><personne>
$liste_personnes = $personnes ->xpath('/personnes/famille/personne');
print_r($liste_personnes);
?>
```

Listing 7. Recherche de toutes les personnes

```
<?php

$personnes = simplexml_load_file('famille2.xml');

// on recherche <personnes><famille><personne>
$liste_personnes = $personnes ->xpath('//personne');
print_r($liste_personnes);
?>
```

Listing 8. Exemple de flux rss

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
  <channel>
    <title>Mon site</title>
    <description>Ceci est un exemple de flux RSS 2.0</description>
    <lastBuildDate>Wed, 27 Jul 2005 00:30:30 -0700</lastBuildDate>
    <link>http://www.example.org</link>
    <item>
      <title>Actualité N°1</title>
      <description>Ceci est ma première actualité</description>
      <pubDate>Tue, 19 Jul 2005 04:32:51 -0700</pubDate>
      <link>http://www.example.org/actul</link>
    </item>
  </channel>
</rss>
```

Listing 9. Lecture du flux rss du journal Le Monde

```
<?php
try
{
    //on charge le fichier xml
    if(!@$fluxrss=simplexml_load_file('http://www.lemonde.fr/rss/une.xml')){
        throw new Exception('Flux introuvable'); //on lève une exception si erreur
    }

    // L'on vérifie le contenu du flux (non vide)
    // puis on affiche les informations de base du flux (title et description)
    if(empty($fluxrss->channel->title) || empty($fluxrss->channel->description))
        throw new Exception('Flux invalide');

    echo '<h3>' . $fluxrss->channel->title . '</h3>
<p>' . $fluxrss->channel->description . '</p>';

    $i = 0;//on compte les articles affichés
    $nb_affichage = 15; // on bloque à 15 titres
    echo '<ul>';
    foreach($fluxrss->channel->item as $item)
    {
        echo '<li><a href="'.(string)$item->link.'">'.(string)$item->title.'</a> <i>publié le
'.(string)date('d/m/Y à G\hi',strtotime($item->pubDate)).'</i></li>';
        if(++$i==$nb_affichage)
            break;
    }
    echo '</ul>';
}
catch(Exception $e)
{
    echo $e->getMessage();
}
?>
```

unique balise `<channel>` qui contiendra les diverses informations du flux RSS, obligatoires ou non, ainsi que la liste des différents articles. Chaque article est identifié comme un objet `<item>` qui pourra contenir une balise `<title>` et `<description>`. Il peut y avoir d'autres balises comme `<pubDate>` qui spécifie la date de publication, `<guid>` qui spécifie l'identifiant unique de l'article, `<link>` qui donne l'adresse url de l'article complet en ligne. Ces trois dernières balises sont optionnelles.

Passons maintenant au Listing 9. Grâce à la trentaine de lignes de ce listing, il est possible de lister les articles à la une du site du journal *Le Monde*. En effet, le site propose un flux rss disponible à cette adresse : <http://www.lemonde.fr/rss/une.xml>. Le site propose d'autres flux rss spécifiques (international, planète, france, économie ...). L'objectif du Listing 9 est donc de lire le fichier xml et d'en extraire certaines informations à savoir le titre, l'url et la date de publication.

Comme vous pouvez le voir, nous utilisons ici un try/catch pour gérer les erreurs. Dans le try, on commence d'abord par charger le fichier contenant le flux rss du journal via l'adresse indiquée un peu plus haut. Si lors de la lecture du rss, il y a la moindre erreur (le fichier n'existe pas, ou structure xml incorrecte) on lève une exception. Par la suite l'on vérifie le contenu des balises `<title>` et `<description>` contenues dans `<channel>`. Si ces balises sont vides, on lève là aussi une exception. Enfin, dans une boucle nous allons parcourir l'ensemble des `items` pour afficher leur `<title>`, la date de publication ainsi que le lien menant à l'article sur le site du journal.

Dans le code, nous nous sommes limités à l'affichage des 15 premiers articles.

Conclusion

Nous avons vus ici de manière aussi courte que simple comment utiliser la librairie Sim-

pleXml pour lire un fichier Xml ou rechercher dans son contenu. Le Xml prend aujourd'hui une part de plus en plus importante et savoir travailler avec est aujourd'hui un avantage facile à intégrer au développement d'un projet. Sachant que la lecture dans un fichier est plus rapide que la réalisation d'une requête Sql, il convient de se poser la question si parfois, des données constantes seraient mieux dans un fichier au lieu d'une BDD.

NICOLAS TURMEAU

Passionné du Web, Nicolas Turneau est étudiant à l'Institut d'Informatique Appliquée de Laval. Il utilise le langage PHP tous les jours depuis plusieurs années et participe à divers projets en ligne dont Numénor Online. Il aime partager ses connaissances et écrit par passion différents livres liés ou pas à l'informatique.

Contact : nturneau@iia-laval.fr

Démarrez avec FireBug et FirePHP

Être assisté pendant les développements permet de gagner des heures de corrections intensives.

De nombreux outils existent permettant d'accélérer l'écriture de vos projets PHP. C'est le cas de FirePHP qui vous est présenté dans ce numéro.

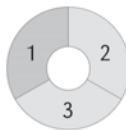
Cet article explique :

- Firebug.
- FirePHP.
- Deux méthodes.

Ce qu'il faut savoir :

- Quelques notions de PHP.
- Utiliser Mozilla Firefox.

Niveau de difficulté



Lors de la réalisation d'un site internet, les développeurs sont souvent confrontés à des erreurs de saisie ou de frappes, donc des bugs. Ces bugs, pour être corrigés, peuvent être résolus en utilisant différentes techniques et outils :

- Le débogage humain.
- Le débogage machine.

Le débogage humain

Le débogage humain correspond à une vérification ou relecture de votre code visuellement et si vous connaissez parfaitement le langage, vous pourrez trouver le léger problème rapidement. Cette technique peut aussi se traduire en utilisant des affichages bruts à l'écran comme :

- L'utilisation des fonctions standards PRINT ou ECHO.
- L'utilisation des fonctions évoluées comme VARDUMP() ou PRINT_R().

Bien entendu, d'autres moyens existent pour obtenir des données comme l'envoi d'e-mail sans que l'internaute se rende compte des remontées d'erreurs. Mais cette opération peut également s'effectuer par le biais de fichier de Log.

Le débogage assisté

Le débogage assisté est complètement différent du débogage précédent car vous allez vous faire aider par des outils, qui sont tous variés et différents. Certains vous seront utiles avec votre IDE, et d'autres avec les navigateurs. Les principaux outils de débogage du marché les plus connus sont :

- xDebug.
- Zend Debug.
- FirePHP.

L'article va se pencher sur l'utilisation de FirePHP, car c'est un outil qui évolue régulièrement. Il est important de faire un état des différentes possibilités mais aussi d'appréhender la possibilité de l'utiliser régulièrement.

Introduction

Pour utiliser FirePHP, un logiciel et une application sont nécessaires :

- Mozilla Firefox,
- Firebug.

Mozilla Firefox

Mozilla Firefox est un navigateur, qui fonctionne sur l'ensemble des systèmes d'exploitation.

Firebug

Le but principal pour le *plug-in* Firebug est d'inspecter et d'éditer du code HTML et CSS. Il permet par ailleurs de voir, de moni-

torer et déboguer toute la partie client (par exemple du code Javascript), en gardant le principe de console de commande. Bien sûr, vous pouvez obtenir une vue des échanges effectués entre le serveur et le navigateur.

Il existe d'autres extensions se positionnant autour de Firebug comme :

- Firelog : gestion des logs dans Firebug.
- Firecookie : gestion des cookies dans Firebug.



Figure 1. Résultat des icônes

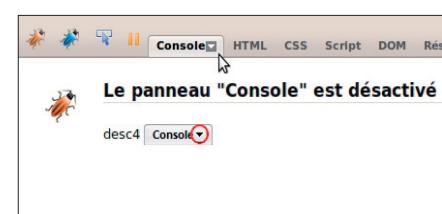


Figure 2. Activation

- Yslow : analyse une page web.
- FirePHP : obtenir des informations PHP.

L'article va se pencher sur l'utilisation de FirePHP car c'est le point principal qui nous intéresse.

FirePHP

FirePHP est aussi un plug-in de Firefox, s'appuyant sur Firebug et va vous permettre d'effectuer du débogage plus optimisé. Pour l'utiliser, vous devez utiliser Firebug. Le principe d'utilisation de FirePHP, est de vous permettre de placer dans un script PHP des informations qui seront affichées dans la console de Firebug. Ces informations sont transmises par les en-têtes HTTP. Le *plug-in* FirePHP est une classe PHP dont vous définissez les données débuggées à afficher. Grâce à cela, les données seront au format *Json*, permettant d'afficher simplement les données.

Installation

Pour utiliser FirePHP, certaines opérations d'installation sont nécessaires. Vous devez installer Firebug et FirePHP.

Firebug

Pour installer Firebug, vous devez vous rendre sur un des deux liens suivants à partir de votre navigateur Firefox :

- <https://addons.mozilla.org/fr/firefox/addon/1843>,
- <http://getfirebug.com>.

Vous téléchargez ce *plug-in* et vous devez redémarrer votre navigateur pour que celui-ci soit pris en compte.

FirePHP

Pour installer FirePHP, vous devez vous rendre sur un des liens suivants à partir de votre navigateur Firefox :

- <https://addons.mozilla.org/en-US/firefox/addon/6149>,
- <http://www.firephp.org/>.

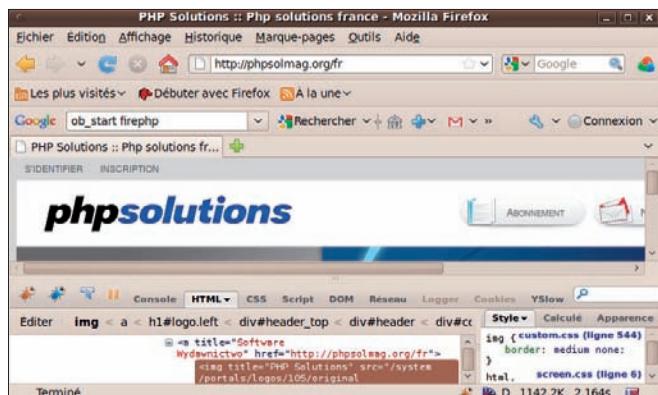


Figure 3. Source d'une page

Listing 1. Test installation procédure

```
<?php
require_once('FirePHPCore/fb.php');

$var="PHP Solutions";
fb($var, FirePHP::getInstance(true));
?>
```

Listing 2. Test installation objet

```
<?php
require_once 'FirePHPCore/FirePHP.class.php';
$var="PHP Solutions";
$Firebug = FirePHP::getInstance(true);
$Firebug->fb($var);
?>
```

Listing 3. Les méthodes

```
<?php
require_once 'FirePHPCore/FirePHP.class.php';
$var="PHP Solutions";

$firephp = FirePHP::getInstance(true);

$firephp->info($var.' signale une condition fausse');
$firephp->warn($var.' signale un warning');
$firephp->error($var.' signale une erreur');
?>
```

Listing 4. Les exceptions

```
<?php
require_once 'FirePHPCore/FirePHP.class.php';
$firephp = FirePHP::getInstance(true);

try
{
    $firephp->fb("DÃ©but du script pour PHP Solutions");
    $firephp->fb($o);

    throw new Exception("Signale un probleme");
}
catch(Exception $e)
{
    $firephp->fb($e);
}

$firephp->fb("fin du script");
?>
```

Comme précédemment, vous devez télécharger ce *plug-in* et redémarrer votre navigateur pour que celui-ci soit aussi pris en compte.

Résultat

Le résultat obtenu, va vous faire apparaître une nouvelle icône dans votre barre de navigation comme le montre la Figure 1.

Activation

Après avoir réalisé les différentes installations, nous devons activer la console. La Figure 2 montre comment réaliser cette opération. Maintenant que ces deux plug-ins ont été installés, que vous possédez des nouvelles icônes en bas de votre navigateur, et que votre console est activée, vous

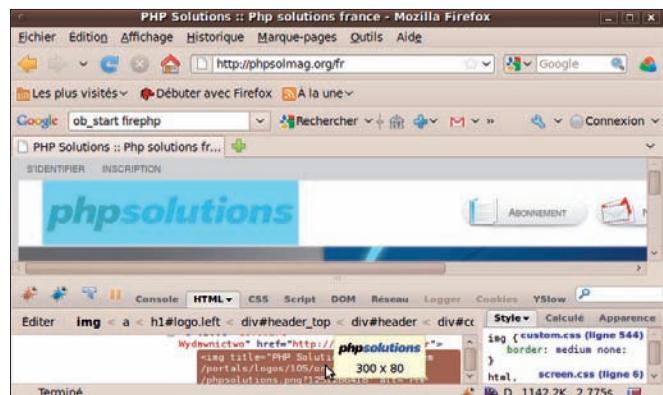


Figure 4. Inspection

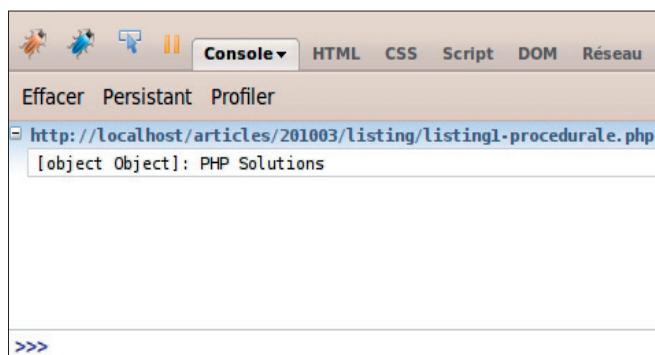


Figure 5. Résultat installation procédurale

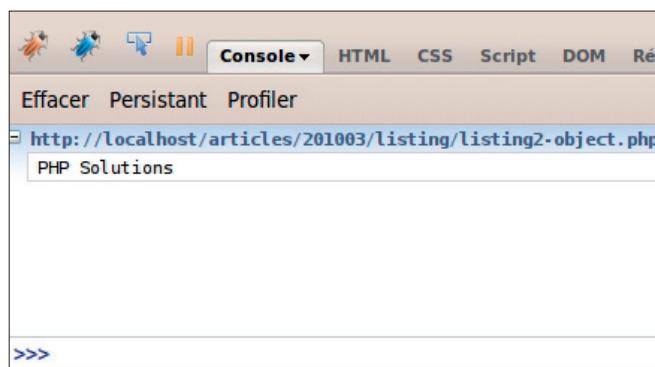


Figure 6. Résultat installation objet

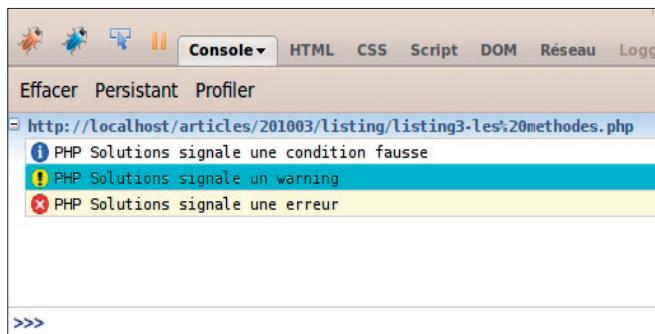


Figure 7. Les méthodes

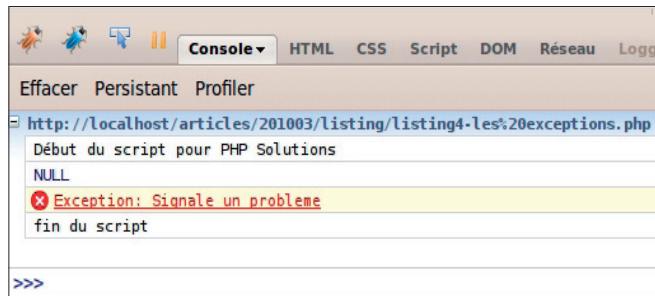


Figure 8. Les exceptions

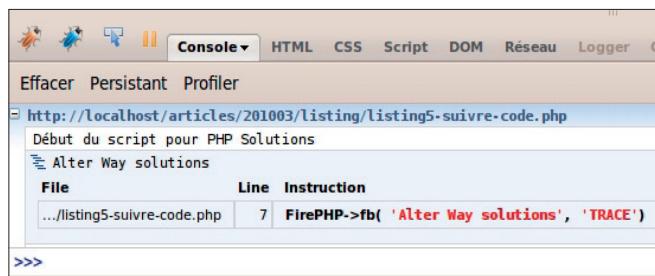


Figure 9. Suivre le code

pouvez passer aux différents tests d'utilisation avec Firebug et FirePHP.

FireBug, utilisation

Avec FireBug, vous pouvez contrôler le code HTML, le Javascript et aussi le CSS via la console. Cette console va lister les différentes actions à chaque opération lorsque la page s'affiche. Pour illustrer cet exemple, l'article va charger la page suivante : `http://www.php-solutions.fr/`. Lorsque que vous ouvrez la console, vous visualisez la source de la page internet, voir le résultat à la Figure 3 (source d'une page).

Inspection

Lorsque vous désirez consulter une partie d'une page, vous devez activer *Inspecter les éléments* et lorsque vous passez la souris sur la zone souhaitée, vous obtenez directement le code source de la partie concernée comme le montre la Figure 4 (*Inspection*).

Décomposition

La console est décomposée en différents onglets. Elle va vous montrer :

- la partie HTML,
- la partie Javascript (JS),
- la partie CSS.

Pour chaque partie de la page, des couleurs différentes vont apparaître. Les informations que vous obtenez correspondent aux CSS utilisés avec des précisions sur le padding, border, margin. Les couleurs ont toutes une signification comme :

- Bleu ciel pour la taille de base l'élément.
- Violet pour le padding en CSS.
- Jaune pour la partie margin.

Bien entendu, vous obtenez une signalisation différente suivant l'onglet qui a été activée.

FirePHP

Avec FirePHP, il existe plusieurs façons d'utiliser ce plug-in, soit en mode procédural soit en mode objet. La méthode procédurale se décompose en différentes étapes :

- Chargement de la librairie *FirePHPCore*.
- Test une variable en déclarant une `getInstance`.

La représentation PHP se présente par le Listing 3 – Résultat installation procédurale. Vous devez voir le résultat montré dans la Figure 5. Le résultat que nous obtenons montre, que FirePHP nous renvoie un résultat sous la forme d'un objet. Ici, la valeur de la variable : `PHP Solutions`. Pour la méthode objet, elle se décompose aussi en différentes étapes :

- Chargement de la librairie *FirePHPCore*.
- Déclaration du `getInstance` en objet.
- Test une variable.

La représentation PHP se présente par le Listing 4 – résultat installation objet.

Vous devez voir le résultat montré dans la Figure 6. Le résultat que nous obtenons montre, que FirePHP nous renvoie un résultat sans signaler qu'il s'agisse d'un objet puisque nous déclarons un objet. Ici, la valeur de la variable : `PHP Solutions`.

Utilisation classique

FirePHP propose une seule fonction pour aider à utiliser l'outil. La syntaxe proposée est `fb()` qui se décompose de la façon suivante : `fb (object, [type])`. Le premier paramétrage concerne l'objet variable et le deuxième est une option paramétrable qui indique comment le contenu sera montré.

Les différentes méthodes qui existent actuellement sont : `info`, `warn`, `error`.

- méthode `info` : La méthode `info` ressemble aux notices et informations générales. Elle se présente de la façon suivante `FirePHP::INFO`,
- méthode `warn` : La méthode `warn` ressemble aux `warning`. Elle se présente de la façon suivante `FirePHP::WARN`,
- méthode `error` : La méthode `error` ressemble au `fatal error`. Elle se présente de la façon suivante : `FirePHP::ERROR`.

Nous mettons en place le Listing 3 (les méthodes), pour voir les différentes méthodes décrites ci-dessus. Grâce à ces méthodes, nous pouvons obtenir des informations différentes avec des petits symboles pour faciliter le repérage des erreurs éventuelles.

La source se présente sous la forme objet pour éviter d'avoir des informations supplémentaires qui ne sont pas utiles lors de l'affichage du résultat. Comme le montre la Figure 5, le résultat obtenu affiche des symboles qui vont vous permettre de repérer facilement les erreurs éventuelles.

Bien sûr, d'autres options existent pouvant être utile de connaître avec FirePHP comme :

- `FirePHP::LOG`
- `FirePHP::DUMP`
- `FirePHP::TRACE`
- `FirePHP::EXCEPTION`
- `FirePHP::TABLE`

Les exceptions

Les exceptions vont vous aider dans la gestion des erreurs, car en utilisant la fonction `FIREPHP::EXCEPTION`, elle va vous permettre d'afficher les erreurs si vous utilisez les fonctions `Try/Catch`, mais au lieu de l'afficher à l'écran avec la fonction `die`, ces informations seront affichées dans votre barre de console comme le présente le Listing 4 (les exceptions).

Nous forçons la main au code pour signaler une erreur, comme ceci, le résultat obtenu est représenté par la Figure 8. Nous pouvons rajouter un message de début et de fin, ce qui permet de repérer plus facilement la partie du code à vérifier.

Suivre le code

L'identification des lignes ou de la fonction qui pose problème, peut se repérer de différentes

Listing 5. Les méthodes

```
<?php

require_once 'FirePHPCore/FirePHP.class.php';
$firephp = FirePHP::getInstance(true);
$var = "Alter Way solutions";

$firephp->fb("Début du script pour PHP Solutions");
$firephp->fb($var,FirePHP::TRACE);

$firephp->fb("fin du script");
?>
```

Listing 6. Les groupes

```
<?php

require_once 'FirePHPCore/FirePHP.class.php';
$firephp = FirePHP::getInstance(true);

$firephp->fb("Début du script pour PHP Solutions");

$firephp->group('Les actualités sont sur...');
$firephp->log('PHP Solutions');
$firephp->log('Nexen');
$firephp->log('Planète PHP');
$firephp->groupEnd();

$firephp->group('Des outils');
$firephp->log('Firefox');
$firephp->log('FirePHP');
$firephp->log('Firebug');
$firephp->groupEnd();

$firephp->fb("fin du script");
?>
```

Listing 7. Les tableaux

```
<?php

require_once 'FirePHPCore/FirePHP.class.php';
$firephp = FirePHP::getInstance(true);

$tableau = array();
$tableau[] = array('Actualités','Outils');
$tableau[] = array('PHP solutions','Firefox');
$tableau[] = array('Nexen','Firebug');
$tableau[] = array('Planète PHP','FirePHP');

$firephp->table('Tableau de présentation', $tableau);
?>
```

Listing 8. Mixe

```
<?php

require_once 'FirePHPCore/FirePHP.class.php';
ob_start();

$firebug = FirePHP::getInstance(true);
?>
<html>
<head>
<style type="text/css">
h1 { color : #FF0000; }
</style>
<script>
document.write ("votre magazine : ");
</script>
</head>
<body>

<?php
$var=<h1>PHP Solutions</h1>;
$firebug->fb($var);
echo $var;
?>
</body>
</html>
```

The screenshot shows the Firefox browser interface with the FirePHP extension active. The title bar says "Console". Below it, there are tabs for "HTML", "CSS", "Script", "DOM", and "Réseau". The main content area displays a table titled "Tableau de présentation" with two columns: "Actualités" and "Outils". Under "Actualités", there are three rows: "PHP solutions" (with a link to "Firefox"), "Nexen" (with a link to "Firebug"), and "Planete PHP" (with a link to "FirePHP"). Under "Outils", there is one row: "Firebug" (with a link to "FirePHP"). At the bottom left, there is a "">>>>" button.

Figure 10. Les groupes

manières. Vous pouvez afficher des messages de repères qu'il faudra bien entendu supprimer lors du passage en production.

L'autre solution consiste à utiliser l'option FirePHP::TRACE. Elle va vous donner les informations supplémentaires comme la ligne où se trouve le problème comme le montre le Listing 5 (suivre le code). Le résultat obtenu est représenté par la Figure 9 (suivre le code)

Avancement

Après avoir réalisé l'installation et les vérifications du bon fonctionnement, vous pouvez maintenant utiliser ce nouvel outil pour vos développements. Une fonction groupe est intégrée dans FirePHP. Elle permet d'effectuer des regroupements et des analyses d'études pour optimiser vos scripts. L'utilisation de cette fonction se présente dans le Listing 6 (les groupes).

L'exemple va créer deux groupes avec des contenus différents :

- un groupe avec les magazines qui proposent des actualités
- un groupe outils

Le résultat obtenu est affiché à la Figure 10. Les groupes sont éclatés pour montrer le contenu, mais lors du chargement de la page, ces groupes sont fermés et vous obtenez un résultat condensé. Attention lorsque vous

utilisez différents groupes, l'analyse des informations devient moins pertinente. La solution consiste donc à placer de la couleur sur un groupe. Cette possibilité se présente comme ceci :

```
<?php
$couleur= array('Collapsed' => true,
    'Color' => '#00FF00');
$firephp->group('Les actualités sont
    sur...',$couleur);
?>
```

Si vous ajoutez ce code au Listing 6, vous obtenez une ligne de couleur verte.

Les tableaux

Lors du traitement des données, vous pouvez obtenir des informations sous la forme d'un tableau pour avoir une meilleure clarté des données. Le script se présente sous la forme du Listing 7 (les tableaux). Pour générer ce tableau, vous devez commencer par les titres de colonnes pour ensuite envoyer des données. Le résultat obtenu est représenté par la Figure 11 (les tableaux)

Le mixe

Vous avez vu au cours de l'article, l'utilisation séparée de ces deux outils (Firebug et FirePHP). Un dernier exemple (Listing 8 – mixe) pour illustrer le mix entre tous les langages utilisés (HTML, PHP, JS et CSS) dans vos pa-

ges et qui va afficher avec une même variable un message dans la page HTML et dans la partie console de FirePHP.

La source pour fonctionner doit être structurée de la façon suivante :

- Le chargement de la librairie FirePHP en lui déclarant l'instance d'activation.
- La définition d'une feuille de style comme une couleur rouge à la balise H1.
- L'utilisation du javascript peut être aussi possible pour afficher un texte d'introduction.
- Dans le corps de notre page PHP, une variable est définie, elle sera affichée une fois dans la page. HTML avec la fonction *echo* et une fois dans la console FirePHP

Le résultat obtenu à la Figure 12 montre le résultat entre la page HTML et la console. Il est tout à fait possible de voir dans la console la partie du code HTML, CSS ou JS en sélectionnant les onglets désirés.

L'avenir

L'article d'aujourd'hui a été réalisé et testé avec Firefox 3.5.7, Firebug 1.5.0 et FirePHP 0.4.3, dont l'ensemble des sources est disponible sur le CD-ROM. Ces outils évoluent régulièrement comme la position des boutons et des fonctions. Un autre projet est en cours de développement, il est considéré comme le successeur de FirePHP et s'intitule *FireConsole*. Actuellement aucune date de sortie n'est annoncée.

Conclusion

D'autres extensions complémentaires à FirePHP existent comme des extensions pour les frameworks Symfony et Zend. Elles se basent sur le même principe : faciliter la réalisation de vos projets.

Bien sur, il est fortement déconseillé d'utiliser et de laisser actif FirePHP lorsque vous passez votre site en production car n'importe quels développeurs qui utilisent ce plug-in, pourra consulter très facilement le contenu de votre site internet. C'est pourquoi, vous devez juste vous en servir pendant la période de développement.

The screenshot shows the Firefox browser interface with the FirePHP extension active. The title bar says "Console". Below it, there are tabs for "HTML", "CSS", "Script", "DOM", "Réseau", and "Logger". The main content area displays a table with two columns: "Actualités" and "Outils". Under "Actualités", there are three rows: "PHP Solutions" (with a link to "Firefox"), "Nexen" (with a link to "Firebug"), and "Planete PHP" (with a link to "FirePHP"). Under "Outils", there is one row: "Firebug" (with a link to "FirePHP"). At the bottom left, there is a "">>>>" button.

Figure 11. Les tableaux

The screenshot shows the Firefox browser interface with the FirePHP extension active. The title bar says "Console". Below it, there are tabs for "HTML", "CSS", "Script", "DOM", "Réseau", and "Logger". The main content area displays a mix of HTML and FirePHP logs. It starts with "Début du script pour PHP Solutions", followed by "Les actualités sont sur...". Under "Actualités", there are three rows: "PHP Solutions" (with a link to "Firefox"), "Nexen" (with a link to "Firebug"), and "Planete PHP". Under "Outils de débogage", there is one row: "Firebug" (with a link to "FirePHP"). The log ends with "fin du script". At the bottom left, there is a "">>>>" button.

Figure 12. Résultat mixe

CHRISTOPHE VILLENEUVE

Consultant, auteur du livre *PHP & MySQL-MySQLi-PDO*, *Construisez votre application, livre français aux Éditions ENI*, et spécialiste des nombreux secteurs PHP (CMS, CRM...) pour Alter Way solutions et contributeur pour de nombreux sites touchant PHP dont Nexen, PHP Team, PHPTV... Contacter l'auteur : <http://www.hello-design.fr>

La sécurité de base d'un serveur

Tout webmaster doit avoir un minimum de connaissances, un minimum de compétences au niveau de la sécurité. Car il existe seulement deux sortes de webmasters : ceux qui ont déjà été victime d'un pirate et ceux qui n'ont jamais eu de soucis mais qui en auront forcément un jour.

Cet article explique :

- Des points de sécurité d'un serveur.

Ce qu'il faut savoir :

- Connaître l'existence du fichier `phpinfo`.

Le premier conseil

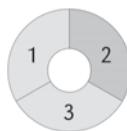
Tout au long de cet article, vous aurez des conseils pour mieux paramétrer votre serveur. Alors, commençons par le premier. Il peut paraître anodin mais comme le dit l'adage : *C'est évident mais c'est mieux en le disant*.

Voici ce conseil :

- il ne faut jamais publier la fonction `phpinfo`, il ne faut jamais publier un fichier `phpinfo.php` avec cette fonction. Même si vous permettez l'accès à une page spéciale à partir d'un mot de passe, il faut être toujours prudent.

Par souci de simplicité, de nombreux webmasters publient ce document. Ceci peut leur permettre de préparer leurs modifications à un endroit avant de les répercuter en réel plus tard sur leur propre serveur. Ceci est une faute importante car si vous, vous voyez bien votre `phpinfo`, tout le monde peut le voir, tout le monde peut y accéder et donc voir des informations cruciales de

Niveau de difficulté



Soyons francs et sincères : combien de fois avez-vous lu des articles sur la sécurité d'un serveur ? Combien de temps avez-vous passé à améliorer la sécurité de votre serveur ? Combien de fois vous vous êtes dit *Ce site a été pirate, pas le mien* ? A partir de ces questions simples, on peut conclure votre état d'esprit sur la sécurité. Si vous avez déjà passé pas mal d'heures à batailler dans les fichiers de configuration, si vous avez lu dans le dernier trimestre plusieurs articles sur la sécurité, votre sensibilité semble correcte. Mais il y a un hic et il est de taille : la grande majorité des webmasters qui ont un serveur dédié ne s'occupent pas assez – voire pas du tout – de cet aspect du métier. Ceci est une faute, il faut y remédier le plus tôt possible et du mieux possible.

Examinons une dizaine de directives, faisons connaissance avec les mesures de sécurité de base. 3, 2, 1, c'est parti !

PhpInfo()

Tout part de la fonction `phpinfo()`. Cette fonction affiche de nombreuses informations sur PHP, concernant sa configuration courante : options de compilation, extensions, version, informations sur le serveur, et l'environnement (lorsqu'il est compilé comme module), environnement PHP, informations sur le système, chemins, valeurs générales et locales de configuration, en-têtes HTTP et la licence PHP.

Les systèmes et les serveurs sont configurés différemment, `phpinfo()` sert donc à vérifier la configuration pour une plate-forme donnée. Si vous installez un serveur Apache en local (avec *EasyPHP*, *Wamp* ou *XAMP* par exemple), vous avez facilement accès à cette fonction. Par exemple, avec *Wamp*, la fonction `phpinfo` est accessible à partir de la partie *Outils* de la page d'accueil. Voir les Figures 1, 2 et 3 et le Listing 1.

The screenshot shows the WampServer configuration page. At the top, it displays the WampServer logo and the text "Version 2.0 English Version". Below this, the "Configuration Serveur" section lists the Apache version (2.2.11) and PHP version (5.3.0). It then lists "Extensions Chargées" (loaded extensions) including Core, date, ereg, json, Reflection, zip, Phar, mbstring, mbxml, mbiconv, mbpre, tokenizer, PDO, xmlreader, and mbstring. To the right of these, other extensions like calendar, filter, mcrypt, session, zlib, SimpleXML, apache2handler, mysql, and mysqli are listed. At the bottom of the configuration section, it says "Version de MySQL: 5.1.36". Below this, there's an "Outils" (Tools) section with links to "phpinfo()" and "phpmyadmin".

Figure 1. Un exemple de page d'accueil

vos serveurs. Le plus grave n'est pas que *tout le monde peut voir ces informations*. Le plus grave est que n'importe quel pirate peut le faire ! Le méchant est maintenant débusqué : le pirate. Car c'est bien contre lui que vous luttez, c'est bien contre lui que vous allez passer des heures à mieux gérer votre serveur. Lorsque vous composez votre numéro de carte bleue sur un terminal, il est maintenant clairement stipulé que vous devez de le faire à l'abri des regards. Pour le `phpinfo`, il doit en être de même.

A ce jour, des dizaines de milliers de `phpinfo` sont totalement visibles sur le Net. Faites une recherche sur le mot `phpinfo` sur votre moteur de recherche préféré et en moins de 3 minutes, vous aurez accès à plusieurs exemples de `phpinfo` de sites Internet. Une remarque : vous constaterez en regardant les `phpinfo` trouvés qu'une grande partie des serveurs est toujours en version PHP4. Quand on connaît les failles de PHP4, on se dit que les pirates ont encore beaucoup de travail devant eux. Alors, autre conseil : si vous êtes toujours en PHP4, migrez très vite en PHP5. Car le délai d'attaque de votre serveur diminue de jour en jour, soyez-en convaincu et persuadé.

allow_url_fopen et allow_url_include

Ces options permettent de traiter les URL comme des fichiers. A partir de la version 5 de PHP, il est largement conseillé de désactiver ces deux directives. Pour faire ce travail, vous devez modifier le fichier `php.ini`. Pour savoir exactement où se trouve ce fichier sur votre serveur, il est préférable de contacter votre hébergeur. Le plus souvent, ses deux directives sont valorisées à `On` (ou 1). Il est préférable – quand on écrit préférable au niveau sécurité, on peut en conclure que c'est obligatoire – de les passer toutes les deux à `Off` (ou 0). Mettez plutôt `Off`, cette option permet une meilleure compréhension. Voici le code attendu dans le Listing 2.

Maintenant, votre sécurité est déjà renforcée mais ce changement a peut-être modifié le comportement de votre site. En effet, ce type d'appel n'est plus possible, voir le Listing 3.

Les fonctions `require()`, `require_once()`, `include()` et `include_once()` sont de fonctions que l'on qualifie d'inclusion. Par définition, elles permettent d'atteindre des fichiers distants. Evidemment, qui dit appel à des fichiers distants dit menace si l'appel est piraté. Dans ce cas, une injection par ces fonctions est tout à fait possible.

Les includes sont dangereux quand ils sont fait sur une variable qui reçoit une valeur en `POST` ou bien encore en `GET`. Même si l'utilisation de variables en dur permet de réduire le risque, il n'est pas annihiler pour autant.

PHP Version 5.3.0

System	Windows NT PC2004460 5.1 build 2600 (Windows XP Professional Service Pack 2) i586
Build Date	Jun 29 2009 21:23:30
Compiler	MSVC8 (Visual C++ 6.0)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--with-oci8-11g=D:\php-sdk\oracle\instantclient11\ sdk,shared" "--with-enchant=shared"
Server API	Apache 2.0 Handler

Figure 2. Un extrait du résultat

Core

Core		
Directive	Local Value	Master Value
<code>allow_call_time_pass_reference</code>	Off	Off
<code>allow_url_fopen</code>	Off	Off
<code>allow_url_include</code>	Off	Off
<code>always_populate_raw_post_data</code>	Off	Off
<code>arg_separator.input</code>	&	&
<code>arg_separator.output</code>	&	&
<code>asp_tags</code>	Off	Off
<code>auto_append_file</code>	<i>no value</i>	<i>no value</i>
<code>auto_globals_jit</code>	On	On
<code>auto_prepend_file</code>	<i>no value</i>	<i>no value</i>
<code>browscap</code>	<i>no value</i>	<i>no value</i>
<code>default_charset</code>	<i>no value</i>	<i>no value</i>
<code>default_mimetype</code>	text/html	text/html
<code>define_syslog_variables</code>	Off	Off
<code>disable_classes</code>	<i>no value</i>	<i>no value</i>
<code>disable_functions</code>	<i>no value</i>	<i>no value</i>
<code>display_errors</code>	Off	Off
<code>display_startup_errors</code>	On	On
<code>doc_root</code>	<i>no value</i>	<i>no value</i>
<code>docref_ext</code>	<i>no value</i>	<i>no value</i>
<code>docref_root</code>	<i>no value</i>	<i>no value</i>
<code>enable_dl</code>	Off	Off
<code>error_append_string</code>	<i>no value</i>	<i>no value</i>
<code>error_log</code>	<i>no value</i>	<i>no value</i>
<code>error_prepend_string</code>	<i>no value</i>	<i>no value</i>
<code>error_reporting</code>	30719	30719
<code>exit_on_timeout</code>	Off	Off
<code>expose_php</code>	Off	Off
<code>extension_dir</code>	c:/wamp_pc/bin/php/php5.3.0/ext/	c:/wamp_pc/bin/php/php5.3.0/ext/
<code>file_uploads</code>	Off	Off

Figure 3. Un autre extrait du résultat, avec des lignes que l'on va modifier

Ainsi, il est conseillé de désactiver les directives `allow_url_fopen` et `allow_url_include` et pour communiquer, on utilisera la bibliothèque cURL, dont c'est le principe de communiquer avec l'extérieur d'une façon sécurisée.

Pour faire un travail correct au niveau sécurité, il faut maintenant activer l'extension `cURL`. Là aussi, il faut modifier le fichier `php.`

`ini`. Ouvrez ce fichier et cherchez l'extension `cURL`. Si elle est en commentaire (le caractère ; en premier), enlevez le. On doit ainsi obtenir le code : `extension=php_curl.dll`. Si elle avez fait une modification, relancez votre serveur afin de prendre en compte cette modification. Mais cette partie n'est pas terminée. Il faut maintenant que vous modifiez le

Listing 1. Sauvegarder ces lignes de programmation dans un fichier .php à la racine de votre serveur local puis exécutez le

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

On peut également extraire une partie du résultat.
On affiche que les paramètres de configuration du serveur.
```

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

Listing 2. Modifications des directives allow_url_fopen et allow_url_include

```
Estat initial
allow_url_open = On
allow_url_include = On

Modifications des directives « allow_url_fopen » et « allow_url_include ».
Estat final
allow_url_open = Off
allow_url_include = Off
```

Listing 3. Appel à require/include avec les directives allow_url_fopen et allow_url_include à On

```
//URL à afficher.
$url = "mon_URL"

//Affichage du lien.
require_once($url);
```

Listing 4. Le programme modifié

```
<?php

//Initialisation de la cURL.
$ch = curl_init();

//URL à afficher avec maintenant la technique cURL.
$url = "mon_URL";

//Préparation de la commande cURL.
curl_setopt($ch, CURLOPT_URL, $url);

//Exécution de la commande cURL.
curl_exec ($ch);

//Fermeture de la commande cURL et libération des ressources.
//Cette commande est obligatoire pour ne pas surcharger le serveur.
curl_close($ch);

?>
```

Listing 5. Les informations affichées

```
Not Found
The requested URL /testURL.php was not found on this server.

Apache/2.2.3 (Debian) mod_python/3.2.10 Python/2.4.4 PHP/5.2.0-8+etch15
mod_ssl/2.2.3 OpenSSL/0.9.8c mod_perl/2.0.2 Perl/v5.8.8 Server at www.
MONSITE.com Port 80
```

Listing 6. Modification de la directive open_basedir

```
open_basedir = « C:\wamp\www\test;c:\wamp\www\scripts » // Sous Windows
open_basedir = « /home/local/www/test:/home/local/www/scripts » // Sous Unix
```

Listing 7. Modification de la directive disable_functions

```
disable_functions = symlink, proc_open , popen, disk_free_space, set_time_
limit, putenv, set_magic_quotes_runtime, sys_get_temp_dir, import_request_
variables, leak, tmpfile, exec, system, shell_exec, passthru, ignore_user_
abort, register_shutdown_function, register_tick_function
```

code qui est du style du Listing 3. Vous avez un exemple concret et précis d'un code après les modifications de configuration dans le Listing 4.

display_errors

PHP dispose d'une gestion d'erreur et suivant le paramétrage, il est tout à fait possible d'afficher ouvertement des erreurs. Et si les erreurs sont totalement visibles, elles le sont également pour le pirate. Si voir des erreurs est primordial pour le développement, ce n'est pas le cas quand l'application est déployée en production. Il faut donc adapter ce paramètre `display_errors` suivant l'endroit où est l'application, suivant le stade d'avancement de celle-ci.

Ce paramètre est valorisé à `On` par défaut. Donc, il n'est pas nécessaire de le modifier lors de la phase de développement. Par contre, lorsque l'application est mise en production, il faut le changer et le valoriser à `Off`. La méthode de modification est la même que pour les directives `allow_url_fopen` et `allow_url_include`.

expose_php

Cette directive indique ou non si PHP doit être affiché comme étant installé sur le serveur en ajoutant sa signature dans les en-têtes du serveur web. On peut rapprocher cette directive de `display_errors` sur le fait que mal interprétée cette valeur donne de l'information au pirate. Mais contrairement à `display_errors` qui donne des informations au niveau développement, `expose_php` donne de précieux renseignements au pirate au niveau technique.

Ainsi, le pirate connaîtra votre type de serveur, connaîtra votre version de PHP. Avec ces informations, le pirate pourra ainsi mieux et plus rapidement cibler son attaque. Voir le Listing 5 pour un exemple d'information affichée.

Dans cet exemple, on voit la version du serveur Apache (2.2.3), son type (Debian) ainsi que la version précise de PHP (5.2.0-8+etch15). Le pirate connaît donc maintenant mieux votre configuration et donc les failles car de nombreux sites Internet les répertorient. A noter également que la gestion du code 404 (*Not Found - Pas trouvé*) peut aider à améliorer cette directive. En effet, dans ce cas, une page que l'on a créé est affichée.

file_uploads

Cette directive autorise ou non le chargement de fichiers par HTTP. Un pirate peut s'attaquer à vos scripts, à vos programmes, à vos bases de données mais il a d'autres angles d'attaques. Celui du déni de service en est un exemple. Il est tout aussi important d'en avoir conscience.

Par exemple, un pirate va tenter de faire télécharger des fichiers, le plus souvent avec une taille importante. Et si votre serveur reçoit une forte et soudaine demande de téléchargement, il aura autant moins de ressources pour gérer le quotidien. Donc, votre site sera perturbé et ralentit. Si ce ralentissement dure trop longtemps, les internautes quitteront votre site, las-sés par tant de lenteur. Un internaute perdu ne reviendra probablement pas sur votre site.

Ceci dit, suivant la nature de votre site, il est possible que vous proposiez des téléchargements. Dans ce cas, il faut avoir cette directive `file_uploads` valorisée à `On` et c'est au niveau des paramètres `post_max_size` et `upload_max_filesize` qu'il faut agir. Pour modifier cette directive, il faut également passer par le fichier `php.ini`. Modification de la directive `file_uploads`.

```
file_uploads = Off.
```

Un nouveau conseil : maintenant, il existe des centaines de sites de téléchargement. Alors, pour limiter au maximum le risque décrit ici, positionnez la directive `file_uploads` à `Off`, connectez-vous à un site de téléchargement, déposez vos fichiers et mettez simplement le lien de téléchargement sur votre site Internet. Ainsi, vous gagnez encore sur votre sécurité.

magic_quote_gpc

Ah, la fameuse directive `magic_quote_gpc` ! Pour ceux qui ont un site sur un serveur dédié depuis 10 ans comprendront l'exclamation concernant cette directive. Il y en a eu des débats sur Internet la concernant. Le passé vieux de 10 ans et celui vieux de 5 ans ne nous intéressent pas, regardons ce qu'il en est en 2010. Pour autant, pour les plus curieux, il est tout à fait possible de faire des recherches sur cette directive pour connaître son historique.

Cette directive permettait de se prémunir contre les injections à l'aide de guillemets en ajoutant systématiquement des barres obliques inverses dans les chaînes de caractères qui arrivent au programme PHP. Mais il a été prouvé que cet ajout pouvait avoir l'effet inverse suivant le jeu de caractères utilisé. Ainsi, il a donc été décidé de la déprécier en PHP5 à partir de la version 5.3.0 puis de la supprimer en PHP6. Cette directive est par défaut à `On`. Procédez ainsi pour la valoriser à `Off` :

```
magic_quote_gpc = Off
```

Pendant que vous êtes sur le traitement des directives `magic`, désactivons les directives cousines de `magic_quote_gpc` : `magic_quote_runtime`, `magic_quote_sybase` en les positionnant à `Off`.

Sur Internet

- <http://www.wampserver.com/> – Pour installer l'outil Wamp,
- <http://www.apachefriends.org/fr/xampp.html> – Pour installer l'outil Xampp,
- <http://www.easyphp.org/> – Pour installer l'outil EasyPHP,
- <http://fr.php.net/manual/fr/book.curl.php> – Pour la bibliothèque cURL.

memory_limit

Cette directive sert à plafonner la quantité de mémoire qu'un programme peut utiliser. Plus cette limite est haute, moins le serveur pourra faire exécuter de scripts simultanément. Il faut donc gérer cette limite correctement. Par défaut, la valeur de cette directive est 8M, soit 8 Mo. Mais depuis cette configuration par défaut, Internet a beaucoup évolué et offre de plus en plus de lecture de photos, de vidéos, d'animation *Flash*, de production de fichiers *PDF* notamment. Heureusement, la taille des serveurs a aussi évolué et la mémoire proposée a suivi cette hausse.

Ainsi, si vous installez PHP à partir de la version 5.2, vous constaterez que cette directive est initialisée à 16 Mo, voire 128 Mo pour la version suivante. Valorisez `memory_limit` à 16 Mo dans un premier temps puis adaptez la valeur à la taille de la mémoire de votre serveur, toujours en modifiant le fichier `php.ini` :

```
memory_limit = 16M
```

open_basedir

Ce qu'il sauvage en premier sur cette directive est son importance. En effet, grâce à cette directive, on n'a plus de restriction d'accès aux fichiers par le propriétaire mais par une racine, par des répertoires. Tous les fichiers, tous les scripts accessibles seront ceux présents dans l'arborescence décrite par `open_basedir`. Voir l'exemple décrit dans le Listing 6.

Avec cette valeur, l'appel à un fichier sous le répertoire *calcul* ne fonctionnera pas alors qu'un appel à un fichier sous le répertoire *scripts* fonctionnera. Mais le traitement de cette directive n'est pas terminé car il existe un problème de sécurité. En effet, en jouant avec la fonction `symlink()`, un pirate peut causer des problèmes au serveur. Cette fonction `symlink()` permet de créer un lien symbolique vers un fichier ou répertoire.

Exemple : deux liens symboliques relatifs sont créés via `symlink()`. Le premier servant de base au second qui pointe vers le fichier à lire dans le répertoire du serveur. `open_basedir` autorise l'opération car le fichier pointé existe. Cependant, si le premier lien symbolique est remplacé par un répertoire de même nom, alors le second lien symbolique pointe maintenant vers le fichier cible hors du répertoire de base du serveur, contournant alors la restriction.

Un attaquant peut donc lire et/ou créer un fichier hors de la racine du site web. Pour contrer le pirate, il est conseillé de désactiver cette fonction `symlink`. Ici, on va apprendre une nouvelle directrice : `:disable_functions`. Voir le paragraphe consacré à cette directive.

register_globals

Cette directive indique si les variables EGPCS (*Environment, GET, POST, Cookie, Server*) seront enregistrées comme des variables globales. Heureusement, depuis la version 4.2 de PHP, la valeur par défaut de cette directive est `Off` mais étant donné l'importance de cette

Tableau 1. Valorisation des directives pour une sécurité de base

Style	Application
<code>allow_url_fopen</code>	<code>Off</code>
<code>allow_url_include</code>	<code>Off</code>
<code>display_errors</code>	<code>Off</code>
<code>expose_php</code>	<code>Off</code>
<code>file_uploads</code>	<code>Off</code>
<code>magic_quote_gpc</code>	<code>Off</code>
<code>memory_limit</code>	<code>16M</code>
<code>open_basedir</code>	A vous de la définir
<code>register_globals</code>	<code>Off</code>
<code>disable_functions</code>	<code>symlink, proc_open, popen, disk_free_space, set_time_limit, putenv, set_magic_quotes_runtime, sys_get_temp_dir, import_request_variables, leak, tmpfile, exec, system, shell_exec, passthru, ignore_user_abort, register_shutdown_function, register_tick_function</code>

Chez votre marchand de journaux

faille et la parfaite connaissance de celle-ci par les pirates, il est préférable d'en vérifier la bonne valeur.

disable_functions

Au-delà des directives que l'on a vu dans cet article, il est intéressant de connaître disable_functions. Celle-ci permet d'en désactiver toute une série en faisant simplement une liste. En effet, avec la version 5 de PHP, toute une série de fonctions peuvent être désactivées pour une meilleure sécurité. Citons les fonctions :

```
proc_open , popen, disk_free_space, set_time_limit, putenv, set_magic_quotes_runtime, sys_get_temp_dir, import_request_variables, leak, tmpfile, exec, system, shell_exec, passthru, ignore_user_abort, register_shutdown_function, register_tick_function.
```

Ces fonctions permettent tantôt de manipuler des processus, tantôt d'accéder au réseau, tantôt d'administrer PHP ou bien encore de gérer un script. Comme les autres, disable_functions se trouve dans le fichier *php.ini*. Voir le listing 7.

Conclusion

Il est obligatoire de passer du temps pour s'occuper de la sécurité de son serveur. Bien sûr, il ne faut pas le faire tous les jours mais il ne faut pas non plus le faire une fois dans l'année, histoire de se donner bonne conscience. Internet évolue tous les jours et à ce titre un bon webmaster doit faire de la veille technologique, y compris au niveau de la sécurité. Car si ce n'est pas avec ce temps passé qu'il gagnera de l'argent, c'est en minimisant cette partie qu'il en perdra énormément, tout comme des heures dans un autre temps et ses nerfs dans un dernier temps. Une passe de sécurité par trimestre ou par semestre (au maximum) est une bonne fourchette de temps.

Pour vous convaincre de passer du temps pour mettre en place cette configuration de base de votre serveur et donc d'avoir la sécurité minimale, faites des recherches avec l'expression *mon site piraté* sur votre moteur de recherche favori : on obtient près de 700 000 articles ! Lisez en détails quelques exemples et vous constaterez que la mise en pratique de cet article est essentiel.

ERIC VINCENT

Eric VINCENT est informaticien de gestion à l'origine et il s'est orienté vers l'Internet aux débuts des années 2000. Il a créé plusieurs sites Internet thématiques et s'intéresse aujourd'hui à la sécurité. Pour contacter l'auteur : ericvincent@nomade.fr



<http://hakin9.org/fr>

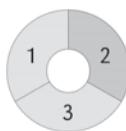
Observez vos objets et réagissez à leurs évènements

Le pattern Observateur définit une relation entre objets de type un-à-plusieurs, de façon à ce que si un objet change d'état, tous ceux qui en dépendent en soient informés et mis à jour automatiquement.

Cet article explique :

- L'utilisation du pattern Observateur.
- Les avantages et les inconvénients de ce pattern.

Niveau de difficulté



Le design pattern Observateur est un des patterns les plus utilisés car il répond à une problématique souvent rencontrée dans le développement orienté objet : comment tenir informé des objets du changement d'état d'un autre objet tout en respectant le principe d'un couplage faible ? Observateur est donc classé dans la catégorie des patterns comportementaux puisqu'il permet d'organiser des objets pour qu'ils collaborent facilement entre eux. Pour illustrer ce pattern, nous allons développer une application autour d'une station de ski.

Exemple d'une station de ski

Cette application sera un extranet afin d'être accessible depuis n'importe quel équipement pouvant se connecter à Internet (PC, smartphone...). Le choix du langage côté serveur se portera évidemment sur PHP dans sa version 5.

La première brique de notre extranet sera la classe `StationSki` qui possède un attribut booléen nommé `$stationOuverte` et qui permet de savoir si la station de ski est ouverte ou fermée. On souhaite qu'un certain nombre de classes puissent facilement se tenir informé du changement

Ce qu'il faut savoir :

- Connaître la programmation orientée objet en PHP 5.
- Quelques notions du langage UML.

d'état de la station de ski. C'est notamment le cas de la classe `Skieurs` qui avertit par mail les habitués de la station de son ouverture ou de sa fermeture. Mais on trouve aussi la classe `SiteStation` qui publie l'état de la station sur le site Internet de la station de ski.

Deux solutions s'offrent alors à nous pour tenir informé les classes `Skieurs` et `SiteStation` lors de l'ouverture ou de la fermeture de la station :

- Soit ce sont les classes `Skieurs` et `SiteStation` qui vont régulièrement lire l'état de la station.
- Soit c'est la classe `StationSki` qui prévient `Skieurs` et `SiteStation` lorsque son état change.

Si l'on souhaite implémenter la première solution on va être confronté à la question suivante : quand les classes `Skieurs` et `SiteStation` devront-elles questionner la station pour connaître son état ? Toutes les minutes ? Toutes les secondes ? Suivant l'intervalle que l'on choisit, soit il y aura un décalage entre le changement d'état de la station et son utilisation par les classes `Skieurs` et `SiteStation`, soit on va surcharger la classe `StationSki` en lisant son état alors que dans la plupart des cas il n'aura pas évolué.

La deuxième solution est donc meilleure puisque c'est la classe `StationSki` qui prévient du changement d'état. Cette information est transmise en temps réel et seulement quand cela est nécessaire, c'est-à-dire quand l'état de la station change. Lors du développement, cela va se traduire par un attribut dans `StationSki` qui va contenir les objets à informer lors d'un changement d'état.

A noter, qu'il serait intéressant qu'un nombre quelconque de classes puisse rester informé de l'état de la station de ski. Actuellement, seules les classes `Skieurs`

Lexique

Design pattern : Présente une solution pour répondre à un problème récurrent dans la conception d'applications orientées objets.

Couplage faible : On parle de couplage faible lorsque deux composants (classes, interfaces...) échangent peu d'informations. On peut alors faire évoluer un composant sans que cela impacte l'autre composant.

Objet Observateur : Objet qui se tient à l'écoute des changements d'états d'une classe afin d'effectuer certaines actions.

Objet Observable : Objet qui facilite l'observation de son état car il est susceptible d'intéresser d'autres objets.

MVC : Le *Modèle Vue Contrôleur* est une architecture qui vise à séparer une application en trois couches : le traitement des données (modèles), l'affichage des données (vues) et la gestion des événements (contrôleurs). Cette séparation permet de faire évoluer une couche en minimisant les impacts sur les autres (par exemple modifier la présentation des données).

et SiteStation sont intéressées par cette information mais dans un futur proche, le site de la mairie souhaite également publier l'état de la station de ski. Vraisemblablement, l'attribut ajouté dans la classe `stationski` sera donc un tableau et pourra donc contenir plusieurs objets.

De plus, il faudrait concevoir l'application de sorte à découpler au maximum la classe `stationSki` des classes qui souhaitent rester informées de l'état. En effet, les classes `skieurs` et `siteStation` sont susceptibles d'évoluer et cela ne doit pas avoir de répercussions dans la classe `stationski` qui est utilisée par de nombreuses classes. Par exemple, les skieurs sont informés par mail de l'ouverture ou de la fermeture de la station. Mais si ce système fait ses preuves on peut envisager que les alertes soient envoyées par SMS pour toucher un public plus large. D'expérience, nous savons que pour découpler des classes, il faut mettre en relation des classes abstraites ou même mieux des interfaces et non des classes concrètes.

Le pattern Observateur

La problématique que nous venons de rencontrer correspond exactement à celle que propose de résoudre le pattern Observateur (en anglais Observer) : Le pattern Observateur définit une relation entre objets de type un-à-plusieurs, de façon que, si un objet change d'état, tous ceux qui en dépendent en soient informés et mis à jour automatiquement.

Le diagramme UML du pattern Observateur présenté dans la Figure 1 schématise la solution à mettre en place. Celui-ci est composé de deux interfaces et deux classes. Toute classe qui souhaite être tenue informée d'un état doit implémenter l'interface Observateur (c'est le cas de la classe ObservateurConcret) et donc déclarer la méthode `actualiser(Observable)`. C'est cette méthode qui sera appelée automatiquement lors du changement d'état de la classe observée.

A l'inverse, toute classe disposant d'un état et qui souhaite pouvoir informer d'autres classes lors de son changement doit implémenter l'interface `Observable` (c'est le cas de la classe `observableConcret`). Cette classe dispose alors d'au moins deux attributs, le premier correspond à l'état et le deuxième à un tableau des observateurs de cet état. Les méthodes `ajouterObservateur` (`Observateur`) et `supprimerObservateur` (`Observateur`) permettent respectivement à des objets de s'abonner ou se désabonner pour rester ou arrêter d'être informés des changements d'un état. En effet, pour qu'un objet puisse observer un état il faut que sa classe implémente l'interface `Observable` et qu'il ait été ajouté

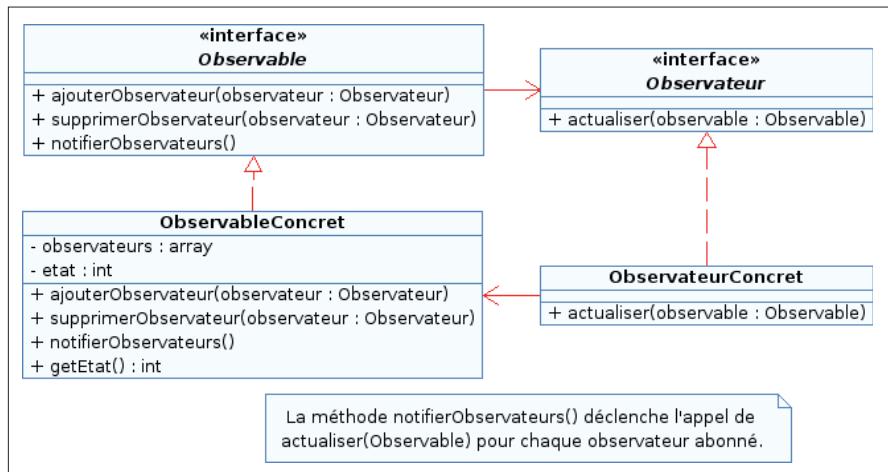


Figure 1. Le diagramme UML du pattern Observateur

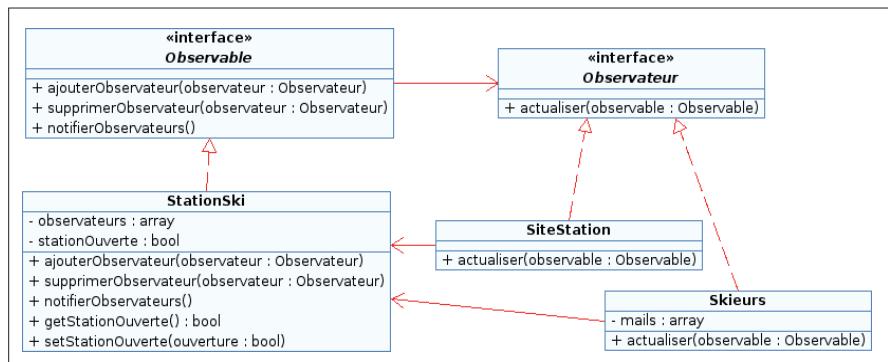


Figure 2. Le diagramme UML de la station de ski basé sur le pattern Observateur

té comme observateur grâce à la méthode ajouterObservateur(Observateur). La méthode notifierObservateurs() de la classe ObservableConcret est appelée lors du changement de valeur de l'état, c'est-à-dire de l'attribut, et appelle toutes les méthodes actualiser(Observable) des observateurs abonnés. Pour finir la méthode getEtat() est un simple accesseur en lecture qui permettra aux observateurs de récupérer la valeur de l'état, l'attribut correspondant dans la classe

ObservableConcret étant déclaré en privé. Voyons maintenant comment appliquer cette solution à notre extranet pour gérer la station de ski.

Le pattern Observateur pour la station de ski

On reprend le diagramme UML du pattern Observateur et on l'applique à notre station de ski (voir Figure 2). On conserve les deux interfaces Observateur et Observable.

Listing 1. L'interface Observable

```
// Interface qu'une classe doit implémenter pour que ses objets soient  
observables.  
interface Observable  
{  
    // Ajoute un observateur à l'objet courant.  
    public function ajouterObservateur(Observateur $observateur);  
    // Supprime un observateur à l'objet courant.  
    public function supprimerObservateur(Observateur $observateur);  
    // Notifie l'ensemble des observateurs du changement d'état de  
l'objet courant.  
    public function notifierObservateurs();  
}
```

Listing 2. L'interface Observateur

```
// Interface qu'une classe doit implémenter pour observer un objet.
interface Observateur
{
    // Méthode appelée automatiquement lors du changement d'état de
    l'objet observé.
    public function actualiser(Observable $observable);
}
```

Listing 3. La classe station de ski

```

// Station de ski qui souhaite pouvoir être observée afin de communiquer sur ses ouvertures/fermetures.
class StationSki implements Observable
{
    // Vrai si la station est ouverte, faux sinon.
    private $stationOuverte;
    // Tableau des observateurs de l'objet.
    private $observateurs;

    // Constructeur d'une station de ski qui permet d'initialiser les attributs.
    public function __construct()
    {
        $this->stationOuverte = false;
        $this->observateurs = array();
    }

    // Permet d'ajouter un observateur à la station de ski.
    public function ajouterObservateur(Observateur $observateur)
    {
        // Si on le souhaite on peut vérifier que l'observateur à ajouter n'observe pas déjà la station de
        // ski.
        if(false !== array_search($observateur, $this->observateurs, true))
        {
            throw new Exception("Cet observateur observe déjà la station de ski.");
        }
        $this->observateurs[] = $observateur;
    }

    // Permet de supprimer un observateur de la station.
    public function supprimerObservateur(Observateur $observateur)
    {
        $clef = array_search($observateur, $this->observateurs, true);
        if(!is_int($clef))
        {
            throw new Exception("Cet observateur n'observe pas la station de ski.");
        }
        unset($this->observateurs[$clef]);
    }

    // Notification des observateurs du changement d'état de la station (ouverte/fermée).
    public function notifierObservateurs()
    {
        foreach($this->observateurs as $observateur)
        {
            $observateur->actualiser($this);
        }
    }

    // Accesseur en lecture afin de savoir si la station est ouverte.
    public function getStationOuverte()
    {
        return $this->stationOuverte;
    }

    // Accesseur en écriture qui permet de mettre à jour l'état de la station puis de notifier les observateurs.
    public function setStationOuverte($ouverture)
    {
        if($this->stationOuverte != $ouverture)
        {
            $this->stationOuverte = $ouverture;
            $this->notifierObservateurs();
        }
    }
}

```

La classe `StationSki` fera office d'observable concret et les classes `Skieurs` et `SiteStation` d'observateurs concrets.

Débutons la présentation du code de notre extranet par les deux interfaces Listing 1 et 2. Pas de surprise, il y a seulement la signature des méthodes présentées dans le diagramme UML. A noter tout de même que le type des paramètres des méthodes est indiqué. Cela permet de déléguer à PHP la vérification du type des paramètres et le cas échéant de générer une erreur. Ce mécanis-

me appelé typage d'objet implicite (en anglais *Type Hinting*) est disponible depuis la version 5 de PHP.

Le Listing 3 présente la classe `StationSki` qui implémente l'interface `Observable`. L'attribut `$stationOuverte` est un booléen (initialisé par défaut à faux par le constructeur) qui correspond à l'état présenté dans le pattern Observateur. Puis l'attribut `$observateurs` est un tableau qui permet de stocker l'ensemble des objets souhaitant observer l'état de la station de ski.

La méthode `ajouterObservateur(Observateur)` est utilisée pour ajouter un objet comme observateur, ce qui se traduit par son insertion dans le tableau `$observateurs`. A noter que si cet objet observe déjà la station on générera une exception. En effet, dans notre cas, on ne souhaite pas qu'un objet observateur puisse être informé plusieurs fois du même changement d'état (par exemple on risquerait d'envoyer les mails en double aux skieurs). La méthode `supprimerObservateur(Observateur)` est bâtie sur le même principe et générera

une exception si l'on essaye de désabonner un objet qui n'est pas abonné.

Pour changer l'état de la station on utilise l'accesseur en écriture `setStationOuverte(booléen)` qui va appeler la méthode `notifierObservateurs()` qui elle même va déclencher l'appel aux méthodes `actualiser(Observable)` pour chaque observateur abonné. Pour rappel, on est certain que cette méthode existe car tous les observateurs doivent implémenter l'interface Observateur qui décrit la signature de cette méthode. A noter que la méthode `setStationOuverte(booléen)` vérifie que le nouvel état passé en paramètre est bien différent de l'état actuel de la station afin d'informer les observateurs seulement si la valeur de l'état a changé.

La classe `Skieurs`, qui observe la station de ski et réagit à un changement d'état, est présentée dans le Listing 4. L'attribut `$mails` est un tableau qui contient l'ensemble des mails des skieurs qui souhaitent être informés de l'état de la station. Pour notre exemple, les mails sont indiqués dans le constructeur mais pour une application réelle, il serait plus pertinent de rechercher ces mails dans une base de données. Toujours à titre d'exemple, suivant l'état de la station on affiche seulement un message mais en commentaire est présenté le code utilisant la fonction `mail()` de PHP.

Sur le même principe, le code du Listing 5 détaille la classe `SiteStation`. Ici aussi, on se contente d'afficher un message en fonction de l'état de la station, mais on pourrait très bien utiliser des services web si le site internet de la station utilise un CMS compatible. Mais cela sort du cadre de notre exemple et je vous invite à lire les différents articles parus dans PHP Solutions qui traitent de SOAP et REST.

Le fichier `index.php` (voir Listing 6) permet de tester l'extranet en créant une station de ski et en faisant observer son état par les skieurs et le site de la station. La fonction magique `__autoload($nom_classe)` permet d'inclure dynamiquement les classes nécessaires à l'extranet. Pour que l'extranet fonctionne, il faut conserver tous les fichiers dans un même dossier et le nom de chaque fichier doit correspondre aux classes (ou interfaces) en minuscules (par exemple la classe `StationSki` est enregistrée dans le fichier `stationski.php`).

Le résultat est présenté dans le Listing 7 et permet de valider que notre extranet fonctionne correctement. On constate que lorsque l'on déclenche l'ouverture de la station seuls les skieurs en sont informés (aucune publication sur le site internet). En effet, l'objet `$siteStation` n'a pas encore été ajouté comme observateur de la station de ski via

Listing 4. La classe `skieurs`

```
// Classe représentant les skieurs qui veulent rester informés de l'état de la station.
class Skieurs implements Observateur
{
    // Tableau contenant les mails des skieurs.
    private $mails;

    // Constructeur qui pour l'exemple initialise les mails des skieurs.
    public function __construct()
    {
        $this->mails = array('jean@example.com', 'pierre@example.com');
    }

    // Méthode appelée automatiquement lors d'un changement d'état (ouverture/fermeture) de la station.
    public function actualiser(Observable $observable)
    {
        // On teste l'état de la station afin de générer le message.
        $message = "Désolé mais la station de ski vient de fermer.";
        if(true == $observable->getStationOuverte())
        {
            $message = "La station de ski est maintenant ouverte.";
        }
        // On parcourt l'ensemble des skieurs pour les informer.
        foreach($this->mails as $mail)
        {
            // Pour visualiser le résultat on affiche l'action effectuée mais on pourrait facilement envoyer les mails.
            echo "Mail envoyé à $mail avec le message suivant : $message <br/>";
            //mail($mail, "Station de ski.", $message, "From: stationski@example.com \r\n");
        }
    }
}
```

Listing 5. La classe du site de la station de ski

```
// Classe représentant le site Internet de la station.
class SiteStation implements Observateur
{
    // Méthode appelée automatiquement lors d'un changement d'état (ouverture/fermeture) de la station.
    public function actualiser(Observable $observable)
    {
        // Vérification de l'état de la station.
        if(true == $observable->getStationOuverte())
        {
            echo "Publication d'un article sur le site Internet de la station au sujet de l'ouverture de la station.<br/>";
        }
        else
        {
            echo "Publication d'un article sur le site Internet de la station annonçant la fermeture de la station.<br/>";
        }
    }
}
```

Listing 6. Le fichier `index.php` pour tester le fonctionnement de la station de ski

```
// Fonction magique qui permet d'inclure automatiquement les classes.
function __autoload($nom_classe)
{
    require __once(strtolower($nom_classe).'.php');
}

// Création de l'objet représentant la station de ski.
$stationSki = new StationSki();
// On crée les observateurs potentiels que sont les skieurs et le site internet de la station.
$skieurs = new Skieurs();
$siteStation = new SiteStation();
// On ajoute les skieurs comme observateur de la station.
$stationSki->ajouterObservateur($skieurs);
// On simule l'ouverture de la station.
$stationSki->setStationOuverte(true);
// On ajoute le site internet de la station comme observateur de la station.
$stationSki->ajouterObservateur($siteStation);
// On simule la fermeture de la station.
$stationSki->setStationOuverte(false);
```

Listing 7. Le résultat affiché lors de l'exécution de l'application

Mail envoyé à jean@example.com avec le message suivant : La station de ski est maintenant ouverte.
 Mail envoyé à pierre@example.com avec le message suivant : La station de ski est maintenant ouverte.
 Mail envoyé à jean@example.com avec le message suivant : Désolé mais la station de ski vient de fermer.
 Mail envoyé à pierre@example.com avec le message suivant : Désolé mais la station de ski vient de fermer.
 Publication d'un article sur le site Internet de la station annonçant la fermeture de la station.

Listing 8. La classe station de ski en utilisant la SPL

```
// Station de ski qui souhaite pouvoir être observée afin de communiquer
// sur ses ouvertures/fermetures.
class StationSki implements SplSubject
{
    // Vrai si la station est ouverte, faux sinon.
    private $stationOuverte;
    // Tableau des observateurs de l'objet.
    private $observateurs;

    // Constructeur d'une station de ski qui permet d'initialiser les
    // attributs.
    public function __construct()
    {
        $this->stationOuverte = false;
        $this->observateurs = array();
    }

    // Permet d'ajouter un observateur à la station de ski.
    public function attach(SplObserver $observateur)
    {
        // Si on le souhaite on peut vérifier que l'observateur
        // à ajouter n'observe pas déjà la station de ski.
        if(false !== array_search($observateur, $this->
observateurs, true))
        {
            throw new Exception("Cet observateur observe
déjà la station de ski.");
        }
        $this->observateurs[] = $observateur;
    }

    // Permet de supprimer un observateur de la station.
    public function detach(SplObserver $observateur)
    {
        $clef = array_search($observateur, $this->observateurs,
true);
        if(!is_int($clef))
        {
            throw new Exception("Cet observateur n'observe
pas la station de ski.");
        }
        unset($this->observateurs[$clef]);
    }

    // Notification des observateurs du changement d'état de la
    // station (ouverte/fermée).
    public function notify()
    {
        foreach($this->observateurs as $observateur)
        {
            $observateur->update($this);
        }
    }

    // Accesseur en lecture afin de savoir si la station est ouverte.
    public function getStationOuverte()
    {
        return $this->stationOuverte;
    }

    // Accesseur en écriture qui permet de mettre à jour l'état de la
    // station puis de notifier les observateurs.
    public function setStationOuverte($ouverture)
    {
        if($this->stationOuverte != $ouverture)
        {
            $this->stationOuverte = $ouverture;
            $this->notify();
        }
    }
}
```

la méthode `ajouterObservateur(Observateur)`. Une fois cet ajout effectué, lorsqu'on ferme la station, l'information est propagée à la fois aux skieurs et sur le site web de la station.

Approfondissement du pattern Observateur

Maintenant que nous savons utiliser le pattern Observateur, voyons un peu les différentes variantes que l'on pourra rencontrer et leurs avantages et inconvénients. Tout d'abord il est tout à fait possible d'ajouter plusieurs attributs dans un observable concret. Par exemple, en plus de l'attribut `$stationOuverte` de la classe `StationSki`, on pourrait rajouter un attribut `$dateOuverture` afin de connaître la date de la première ouverture de cette saison. On informera alors les observateurs lors de l'ouverture ou de la fermeture de la station mais s'ils le souhaitent, ils pourront également accéder à ce nouvel attribut grâce à un accesseur en lecture. Par exemple, on pourra publier sur le site internet de la station la date d'ouverture de la station cette saison.

Dans certaines applications et dans certains exemples sur internet, les deux interfaces `Observable` et `Observateur` sont remplacées par des classes abstraites. Cette implémentation a pour but de déplacer une partie du code au niveau des classes abstraites. En effet, on peut alors déplacer l'attribut `$observateurs` dans la classe abstraite `observable` ainsi que le corps des méthodes `ajouterObservateur(Observateur)`, `supprimerObservateur(Observateur)` et `notifierObservateur()`. Le principal inconvénient est que l'on restreint la possibilité d'utilisations de ces classes. En effet, PHP ne supporte pas l'héritage multiple et donc si l'on souhaite utiliser le pattern Observateur sur une classe qui hérite déjà d'une autre classe cela s'avère impossible.

On trouve également des cas où le pattern Observateur est appliqué mais sans interface ni classe abstraite (uniquement avec les deux classes concrètes). Il est fortement déconseillé d'utiliser cette implémentation car on est alors dépendant des types concrets des classes utilisées et toute évolution est rendue plus difficile. En effet, pour s'assurer que les observateurs implémentent bien la méthode `actualiser()` on va devoir lister leurs types concrets dans la classe `ObservableConcret` (car ils n'implémentent pas une interface commune). Il ne sera alors plus possible d'ajouter des observateurs sauf à modifier l'implémentation de la classe `ObservableConcret`. Dans la mesure du possible, il faut donc privilégier l'utilisation des interfaces qui couplent faiblement les observateurs à l'observable.

Une autre variation de ce pattern existe. Dans l'exemple ci-dessous un objet *Observable* est mis à disposition des observateurs qui sont chargés d'aller lire l'état grâce à un accesseur en lecture. Cette solution est nommée *tirer* car c'est aux observateurs de récupérer (tirer) l'information. Il existe la solution inverse qui est de passer directement l'état à la place de l'objet *Observable*. La méthode actualiser dans notre exemple aurait donc comme signature : *actualiser(boolean)*. Cette solution est appelée *pousser* mais elle dispose d'un inconvénient. Si l'implémentation de l'observable change, par exemple si on ajoute un deuxième attribut qui peut intéresser certains observateurs, alors la signature des méthodes *actualiser()* des observateurs doit évo-

Sur Internet

- <http://www.design-patterns.fr/Observateur.html> – Une présentation du pattern Observateur,
- <http://julien-pauli.developpez.com/tutoriels/php/observer-spl/> – Explication du pattern Observateur en utilisant la SPL,
- http://www.phppatterns.com/docs/design/observer_pattern – Le pattern Observateur avec un exemple de forum en PHP (en anglais),
- <http://www.gautheret.com> – Le site de l'auteur de l'article si vous souhaitez le contacter.

luer (ce qui peut être dommageable suivant le nombre d'observateurs potentiels).

Où se cache ce pattern ?

Il est légitime de se demander mais où est utilisé ce pattern ? Dans de nombreuses applications et pour cause il fait partie des patterns

utilisés par l'architecture MVC (*Modèle Vue Contrôleur*). En effet, c'est ce pattern qui va permettre de tenir informé les vues des changements d'état dans les modèles et cela en utilisant un couplage faible. De ce fait on retrouve ce pattern dans les principaux CMS (Joomla...), mais également dans les boutiques

Listing 9. La classe skieurs en utilisant la SPL

```
// Classe représentant les skieurs qui veulent rester informés de l'état de la station.
class Skieurs implements SplObserver
{
    // Tableau contenant les mails des skieurs.
    private $mails;

    // Constructeur qui pour l'exemple initialise les mails des skieurs.
    public function __construct()
    {
        $this->mails = array('jean@example.com', 'pierre@example.com');
    }

    // Méthode appelée automatiquement lors d'un changement d'état (ouverture/fermeture) de la station.
    public function update(SplSubject $observable)
    {
        // On teste l'état de la station afin de générer le message.
        $message = "Désolé mais la station de ski vient de fermer.";
        if(true == $observable->getStationOuverte())
        {
            $message = "La station de ski est maintenant ouverte.";
        }

        // On parcourt l'ensemble des skieurs pour les informer.
        foreach($this->mails as $mail)
        {
            // Pour visualiser le résultat on affiche l'action effectuée mais on pourrait facilement
            envoyer les mails.
            echo "Mail envoyé à $mail avec le message suivant : $message <br/>";
            //mail($mail, "Station de ski.", $message, "From: stationski@example.com \r\n");
        }
    }
}
```

Listing 10. La classe du site de la station de ski en utilisant la SPL

```
// Classe représentant le site Internet de la station.
class SiteStation implements SplObserver
{
    // Méthode appelée automatiquement lors d'un changement d'état (ouverture/fermeture) de la station.
    public function update(SplSubject $observable)
    {
        // Vérification de l'état de la station.
        if(true == $observable->getStationOuverte())
        {
            echo "Publication d'un article sur le site Internet de la station au sujet de l'ouverture
de la station.<br/>";
        }
        else
        {
            echo "Publication d'un article sur le site Internet de la station annonçant la fermeture
de la station.<br/>";
        }
    }
}
```

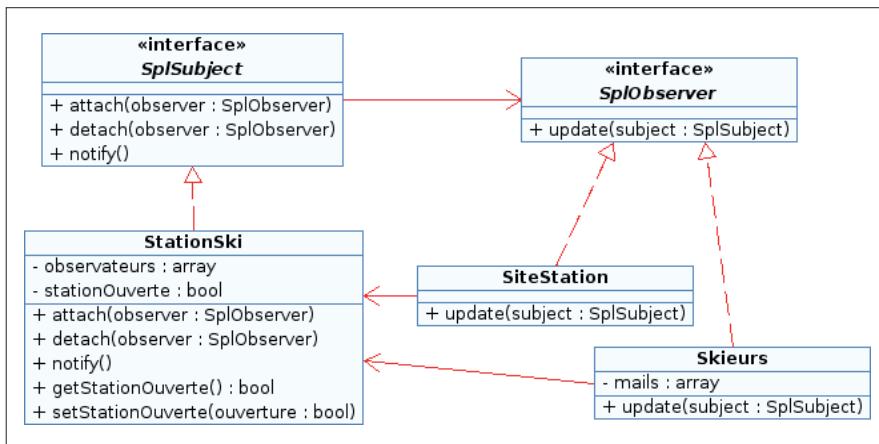


Figure 3. Le diagramme UML de la station de ski en utilisant la SPL

e-commerce (Magento...) voir même dans des extensions de ces applications.

Par exemple dans la version 1.5 du gestionnaire de contenu Joomla si on parcourt le répertoire `/libraries/joomla/base/` on trouve notamment deux fichiers :

- observable.php,
- observer.php.

Si on visualise le code de ces deux fichiers, les commentaires ne laissent aucun doute : il s'agit bien de l'implémentation du pattern Observateur. A noter que dans ce cas l'implémentation est un peu différente afin d'être compatible avec PHP4.

Standardiser l'utilisation du pattern Observateur

D'ailleurs ce pattern est tellement utilisé que les concepteurs de PHP ont mis en place

deux interfaces pour faciliter et standardiser son utilisation. Elles sont présentes dans la SPL (*Standard PHP Library*) qui a fait son apparition dans la version 5 de PHP. Cette extension est compilée par défaut avec PHP ce qui assure la portativité des applications l'utilisant. Voyons comment modifier notre extranet de *station de ski* pour utiliser le pattern Observateur tel que définit dans la SPL.

La SPL définit deux interfaces pour le pattern Observateur. L'interface `SplSubject` remplace l'interface `Observable` et possède trois méthodes similaires dont seul le nom diffère. L'autre interface qui va remplacer Observateur est nommée `SplObserver` toujours avec la même méthode mais nommée différemment. La Figure 3 présente le diagramme UML de l'extranet de la station de ski avec l'utilisation de la SPL. On constate alors que mis à part le nom des interfaces et des

méthodes le fonctionnement reste strictement identique.

Le Listing 8 présente la classe `StationSki` qui implémente maintenant `SplSubject`. Le code des classes `Skieurs` et `SiteStation` est détaillé dans les Listings 9 et 10. A noter que pour utiliser la SPL nous n'avons pas eu besoin de changer le corps des méthodes. Quand au Listing 11 qui présente le fichier `index.php`, on a simplement remplacé le nom de la méthode `ajouterObservateur()` par `attach()`. Sans surprise le résultat obtenu est strictement identique à celui obtenu précédemment (voir Listing 7).

Mais quels sont les avantages d'utiliser les interfaces de la SPL plutôt que nos propres interfaces ? Tout d'abord cela facilite la reconnaissance de pattern car ces interfaces sont connues des développeurs PHP. Mais en plus dans le cas où un objet souhaite observer plusieurs objets distincts si la SPL a été utilisée alors l'observateur doit seulement implémenter l'interface `SplSubject`. Alors que si pour chaque observable, on a utilisé une interface différente (car programmer par différents développeur) on est obligé d'implémenter les deux interfaces et donc de multiplier les méthodes ayant la même fonction.

Conclusion

Le pattern Observateur doit être utilisé à chaque fois qu'un objet souhaite pouvoir tenir informé d'autres objets du changement de son état. La solution mise en place par ce pattern permet de coupler faiblement les classes et donc rend l'application plus évolutive. De manière générale, lors de la phase d'analyse il est important de se demander quelle partie du code est susceptible de changer ou d'évoluer dans le futur de l'application. Une fois les problématiques identifiées, on peut facilement rechercher dans la littérature informatique si un pattern ne résout pas ce problème de façon optimale.

Il est également intéressant de bien connaître les patterns pour s'en servir au mieux lorsqu'on les rencontre dans les applications ou les frameworks que l'on utilise. Et c'est dans cette optique que PHP Solutions présente les différents patterns qui peuvent vous faciliter le développement.

MATHIEU GAUTHERET

L'auteur est ingénieur en informatique et travaille chez 100% Net, une SSII Montpelliéraise spécialisée dans le Web et les logiciels libres. Ce membre de l'APRIL est un passionné du Web et plus particulièrement de l'architecture des applications PHP.

Listing 11. Le fichier index.php en utilisant la SPL

```

// Fonction magique qui permet d'inclure automatiquement les classes.
function __autoload($nom_classe)
{
    require_once(strtolower($nom_classe).'.php');
}

// Créeation de l'objet représentant la station de ski.
$stationSki = new stationSki();

// On crée les observateurs potentiels que sont les skieurs et le site
// internet de la station.
$skieurs = new Skieurs();
$siteStation = new SiteStation();

// On ajoute les skieurs comme observateurs de la station.
$stationSki->attach($skieurs);

//On simule l'ouverture de la station.
$stationSki->setStationOuverte(true);

// On ajoute le site internet de la station comme observateur de la
// station.
$stationSki->attach($siteStation);

// On simule la fermeture de la station.
$stationSki->setStationOuverte(false);
  
```



DÉVELOPPEMENT D'APPLICATIONS WEB **SUR MESURE**

Nous sommes convaincus que la réussite de nos clients et de leurs projets web repose sur des solutions spécifiquement adaptées à leurs besoins. Kernix vous propose donc un service sur mesure : de la conception à l'hébergement.



Retrouvez-nous les 29, 30 septembre & 1^{er} octobre 2009 sur le stand n°147 au SALON E-COMMERCE 2009

En savoir plus sur www.kernix.com

PHP et les expressions régulières

Les expressions régulières ont été grâce à leur puissance adoptées par de nombreux langages de programmation et ont vues leur champ d'utilisation s'étendre des fichiers htaccess aux feuilles de style CSS. Voyons comment tirer profit des nombreux avantages qu'offrent les expressions régulières.

Cet article explique :

- L'utilisation des expressions régulières en PHP.

Niveau de difficulté



Les expressions régulières sont une suite d'expressions logiques qui permettent de vérifier le format d'une chaîne de caractères devant correspondre à un modèle précis (pattern en anglais). La manipulation des expressions régulières est assurée par deux bibliothèques : POSIX et PCRE. Ces fonctions seront présentées dans cet article.

Structure d'une expression régulière

Une *Regex* est composée de deux parties séparées par des délimiteurs (séparateurs) :

- la première constitue le modèle qui sera appliquée à la chaîne de caractères.
- la seconde correspond aux options possibles qui peuvent suivre le modèle.

Le caractère # est le plus utilisé comme délimiteur mais il est possible de choisir ce qu'on veut. Une *Regex* a donc la structure suivante : #*Regex*#Options.

Tableau des métacaractères

Les métacaractères ou caractères spéciaux ont une signification particulière dans la construction des motifs de recherche. Ils permettent de créer des noms génériques.

Ce qu'il faut savoir :

- Avoir une connaissance minimum de la syntaxe de base des expressions régulières et des fonctions PHP qui les utilisent.

Règles générales :

- il faut le faire précéder par un \ (*antislash*) pour utiliser un métacaractère dans une recherche,
- Les métacaractères n'ont pas besoin d'être échappés dans une classe, sauf pour # (symbole de fin de la *regex*) et] (symbole de la fin de la classe) que l'on doit faire précéder d'un antislash.
- Le crochet fermant et le tiret garde leur statut de métacaractère dans une classe. Par contre en dehors de la classe, chacun d'eux est un simple littéral.

Tableau des classes prédéfinies

Les classes prédéfinies sont contenues entre des doubles crochets, elle permettent de simplifier la syntaxe des *regex*.

Les classes abrégées

Les classes abrégées permettent de réduire la longueur des expressions régulières.

Les fonctions PCRE

PCRE est l'acronyme de Perl Compatible Regular Expressions. Les fonctions PCRE commencent toutes par `preg_`. Ces fonctions PCRE sont :

- `preg_grep` - Retourne un tableau contenant les éléments recherchés.
- `preg_last_error` - Retourne le code erreur de la dernière expression PCRE exécutée.

- `preg_match_all` - Expression rationnelle globale.
- `preg_match` - Expression rationnelle standard (Listing 1).
- `preg_quote` - Protection des caractères spéciaux des expressions rationnelles.
- `preg_replace_callback` - Recherche et remplace par expression rationnelle en utilisant une fonction de callback.
- `preg_replace` - Recherche et remplace par expression rationnelle standard (Listing 2).
- `preg_split` - Éclate une chaîne par expression rationnelle (Listing 3).
- `preg_filter` - Recherche et remplace avec une expression rationnelle.

Ces fonctions utilisent des délimiteurs qui servent à indiquer le début du masque et sa fin. Ces délimiteurs peuvent être suivis d'options.

Un masque classique se présente de la forme suivante : délimiteur | motif | délimiteur, options.

N.B : il faut tout de même préciser qu'un délimiteur ne doit en aucun cas être un alpha-numérique ou un antislash \ qui est réservé car il sert de caractère d'échappement au délimiteur.

Bien qu'autorisés depuis la version 4.0.4 de PHP, les délimiteurs symétriques {}, {}, {}, et <>, il serait plus prudent d'éviter de les utiliser, de même que le point d'exclamation !. Le jeu de caractères installé sur le serveur peut également poser des problèmes.

Les options de PCRE

Les options de PCRE sont listées ci-dessous (cette liste est tirée de la documentation PHP). Les noms entre parenthèses sont les noms internes à PCRE. Les espaces et les caractères de nouvelles lignes sont ignorés dans les modificateurs, les autres caractères causent des erreurs.

Tableau 1. Tableau des métacaractères

Caractères	Descriptions
\ (Antislash)	Caractère de protection ou caractère d'échappement à usage multiple
^ (Accent circonflexe)	Le début d'une chaîne de caractères ou de ligne, en mode multilignes (option m).
\$ (Dollar)	La fin d'une chaîne de caractères ou de ligne, en mode multilignes (option m).
. (Point)	Remplace n'importe quel caractère, hormis un saut de ligne (sauf si l'option s est active) .
* (Astérisque)	Quantificateur de 0 ou plus.
+ (Plus)	Quantificateur de 1 ou plus.
(Barre verticale)	Caractère de début d'alternative
? (Point d'interrogation)	Nombre d'occurrence de 0 ou 1 du caractère ou de la séquence de caractère qui le précède.
() (Parenthèses)	Caractère de début et de fin de sous-masque.
[] (Crochets)	Caractère de début et de fin de définition de classe
{ } (Accolades)	Caractère de début et de fin de quantificateur minimum/maximum.
! (Point d'exclamation)	Sert de délimiteur.

Tableau 2. Tableau des classes prédéfinies.

Noms de l'intervalle	Description	Classe équivalente
:alnum:]	Tout caractère alphanumérique.	a-zA-Z0-9
:alpha:]	Tout caractère alphabétique.	a-zA-z
:blank:]	Tout caractère blanc (espace, tabulation...).	
:digit:]	Tout caractère numérique.	0-9
:punct:]	Tout caractère de ponctuation.	
:space:]	Caractère d'espacement.	
:xdigit:]	Tout caractère hexadécimal	a-fA-F0-9
:ctrl:]	Tout caractère de contrôle.	
:print:]	Tout caractère imprimable.	
:lower:]	Tout caractère minuscule de a à z.	[a-z]
:upper:]	Tout caractère majuscule de A à Z.	[A-Z]
:graph:]	Tout caractère affichable et imprimable.	

Tableau 3. Tableau des classes abrégées

Classes abrégées	Descriptions
\d	Tout caractère décimal.
\D	Tout caractère qui n'est pas un caractère décimal.
\h	N'importe quel espace horizontal (depuis PHP 5.2.4).
\H	N'importe quel caractère qui n'est pas un espace horizontal (depuis PHP 5.2.4).
\s	Tout caractère blanc.
\S	Tout caractère qui n'est pas blanc.
\v	N'importe quel espace vertical (depuis PHP 5.2.4).
\w	Tout caractère de mot.
\t	Correspond à une tabulation.
\n	Correspond à un saut de ligne.
\r	Correspond à un retour chariot .
. (Point)	N'importe quel caractère.

i (PCRE_CASELESS) : Effectue une recherche insensible à la casse.

m (PCRE_MULTILINE) : Par défaut, PCRE traite la chaîne sujet comme une seule ligne (même si cette chaîne contient des retours chariot). Le métacaractère *début de ligne* (^) ne sera valable qu'une seule fois, au début de

la ligne, et le métacaractère *fin de ligne* (\$) ne sera valable qu'à la fin de la chaîne, ou avant le retour chariot final (à moins que l'option D ne soit activée). C'est le même fonctionnement qu'en Perl. Lorsque cette option est activée, *début de ligne* et *fin de ligne* correspondront alors aux caractères suivant et

précédent immédiatement un caractère de nouvelle ligne, en plus du début et de la fin de la chaîne. C'est le même fonctionnement que l'option Perl/m. S'il n'y a pas de caractère de nouvelle ligne \n dans la chaîne sujet, ou s'il n'y a aucune occurrence de ^ ou \$ dans le masque, cette option ne sert à rien.

Listing 1. Utilisation de la fonction preg_match().

```
<?php
//isset() permet de vérifier si le champ carte bancaire est renseigné
if (isset($_POST['carteBancaire']))
{
// htmlspecialchars() permet d'éviter que des données fournies par les utilisateurs contiennent des //balises HTML.
$_POST['carteBancaire'] = htmlspecialchars($_POST['carteBancaire']);
/*
Notre expression régulière est : #^([0-9]{4}){4}$#
#" (dièse) : il sert à indiquer le début et la fin de la Regex.
^ (Accent circonflexe) : Représente le début d'une chaîne de caractères.
( ) (Parenthèses) : Indique respectivement l'ouverture et la fermeture d'un sous-masque ou séquence de caractères.
[0-9]{4} pour indiquer que l'on veut 4 chiffres pouvant aller de 0 à 9.
{4} pour indiquer que tout ça doit se répéter 4 fois.
$ (Dollar) : Représente la fin d'une chaîne de caractères
*/
if (preg_match("#^([0-9]{4}){4}$#", $_POST['carteBancaire']))
{
    echo $_POST['carteBancaire'] . ' est un numéro de carte bancaire valide!';
}
else
{
    echo $_POST['carteBancaire'] . ' n\'est pas un numéro de carte bancaire valide, recommencez !';
}
?>
</p>
<form method="post">
<p>
    <label for="carteBancaire">Votre numéro de carte bancaire ?</label> <input id="carteBancaire" name="carteBancaire" /><br />
    <input type="submit" value="Vérifier le numéro" />
</p>
</form>
```

Listing 2. Utilisation de tableaux indexés avec preg_replace().

```
<?php
$string = 'Je kiffe Homer Simpson.';
$patterns[0] = '/kiffe/';
$patterns[1] = '/Homer/';
$patterns[2] = '/Simpson/';
$replacements[2] = 'préfère';
$replacements[1] = 'Peter';
$replacements[0] = 'Griffin';
echo preg_replace($patterns, $replacements, $string);
// "Je préfère Peter Griffin" au lieu de "Je kiffe Homer Simpson". Ça se passe de commentaires.
?>
```

Listing 3. Utilisation de tableaux indexés avec preg_split().

```
<?php
$string = 'je code donc je suis';
/*
PREG_SPLIT_OFFSET_CAPTURE retournera l'offset de début de résultat, en plus de la chaîne résultat. Cela change la nature du résultat retourné en un tableau contenant une chaîne à l'offset 0 et une chaîne contenant un offset à 1.
*/
$chars = preg_split('/ /', $string, -1, PREG_SPLIT_OFFSET_CAPTURE);
print_r($chars);
?>
```

Listing 4. Extraction des données d'une table: cas des variables numériques.

```
<?php
try{
    //chaine de connexion
    $cnx=new PDO('mysql:host=localhost; dbname=catalogue', 'root', '');
    /*
        Imaginons que nous ayons une table des catégories et que nous souhaitons récupérer toutes les catégories qui contiennent 4 chiffres mais uniquement de 2 à 9 MySQL utilise les expressions rationnelles étendues avec les opérateurs REGEXP et NOT REGEXP
    */
    $sql="SELECT * FROM categories where ID_CAT REGEXP '([2-9]{4})'";
    foreach($cnx->query($sql) as $row){
        echo $row['ID_CAT'] . '-' . $row['NOM_CAT'] . '-' . $row['DESCRIPTION'] . '<br />';
    }
    $cnx=null;
}
//capture et affichage des exceptions
catch(PDOException $error){die($error->getMessage());}
?>
```

s (PCRE_DOTALL) : Avec cette option, le métacaractère point(.) remplace n'importe quel caractère, y compris les nouvelles lignes. Sans cette option, le caractère point ne remplace pas les nouvelles lignes. Cette option est équivalente à l'option Perl /s. Une classe de caractères négative telle que [^a] acceptera toujours les caractères de nouvelles lignes, indépendamment de cette option.

x (PCRE_EXTENDED) : Avec cette option, les caractères d'espacement sont ignorés, sauf lorsqu'ils sont échappés, ou à l'intérieur d'une classe de caractères, et tous les caractères entre # non échappés et en dehors d'une classe de caractères, et le prochain caractère de nouvelle ligne sont ignorés. C'est l'équivalent Perl de l'option /x : elle permet l'ajout de commentaires dans les masques compliqués. Notez bien, cependant, que cela ne s'applique qu'aux caractères de données. Les caractères d'espacement ne doivent jamais apparaître dans les séquences spéciales d'un masque, par exemple dans la séquence (? qui introduit une parenthèse conditionnelle.

e (PREG_REPLACE_EVAL) : Avec cette option, preg_replace() effectue la substitution normale des références arrières dans la chaîne de remplacement, puis l'évalue comme un code PHP, et utilise le résultat pour remplacer la chaîne de recherche. Les guillemets simples, les guillemets doubles, les antislashs et les caractères NULL sont protégés avec des antislashs (\) dans les références arrières substituées. Seule preg_replace() utilise cette option. Elle est ignorée par les autres.

A (PCRE_ANCHORED) : Avec cette option, le masque est ancré de force, c'est-à-dire que le masque doit s'appliquer juste au début de la chaîne sujet pour être considéré comme trouvé. Il est possible de réaliser le même effet en ajoutant les métacaractères adéquats, ce qui est la seule manière de le faire en Perl.

D (PCRE_DOLLAR_ENDONLY) : Avec cette option, le métacaractère \$ ne sera valable qu'à la fin de la chaîne sujet. Sans cette option, \$ est aussi valable avant une nouvelle ligne, si cette dernière est le dernier caractère de la chaîne. Cette option est ignorée si l'option m est activée. Il n'y a pas d'équivalent en Perl.

S : Lorsqu'un masque est utilisé plusieurs fois, cela vaut la peine de passer quelques instants de plus pour l'analyser et optimiser le code pour accélérer les traitements ultérieurs. Cette option force cette analyse plus poussée. Actuellement, cette analyse n'est utile que pour les masques non ancrés, qui ne commencent pas par un caractère fixe.

U (PCRE_UNGREEDY) : Cette option inverse la tendance à la gourmandise des expressions rationnelles. Vous pouvez aussi inverser cette

Listing 5. Extraction des données d'une table: cas des chaînes de caractère.

```
<?php

try{
    //chaine de connexion
    $cnx=new PDO('mysql:host=localhost; dbname=catalogue', 'root', '');
    //nous souhaitons récupérer toutes les catégories dont le nom
    //est 'DVD' ou 'livres'
    $sql="SELECT * FROM categories where NOM_CAT REGEXP
        '(DVD|livres)';
    foreach($cnx->query($sql) as $row){
        echo $row['ID_CAT'] . '-'. $row['NOM_CAT'] . '-';
        $row['DESCRIPTION'] . '<br />';
    }
    $cnx=null;
}
//capture et affichage des exceptions
catch(PDOException $error){die($error->getMessage());}
?>
```

Listing 6. Validation d'une URL

```
<?php

/*
Construisons d'abord un masque permettant de vérifier l'adresse web d'un site.l'adresse d'un site commence par http:// ou par https:// pour les sites sécurisés. Notre masque commencera donc par ^http(s)?://. Il faut maintenant ajouter le 'www' et le nom de domaine. Le nom de domaine peut être constitué de lettres, de chiffres ou d'un tiret (signe moins). Cela ressemblera à :
^http(s)?://[-[:alnum:]]+\.[-[:alnum:]]+
Les crochets [ ] définissent une liste de caractères autorisés (ou interdits). Le signe - permet quand à lui de définir un intervalle. [:alnum:] équivaut à a-zA-Z0-9 c'est à dire indique tout caractère alphanumérique;
+ (Signe plus) : spécifie le nombre d'occurrence de un ou plus du caractère ou de la séquence de caractère qui le précède.
\ (Antislashe) : indique le caractère d'échappement.
Il faut ajouter maintenant l'extension du domaine (.fr, .com, .net, etc...) qui peut être constitué de deux à trois caractères.
^http(s)?://[-[:alnum:]]+\.[-[:alnum:]]+\.[a-zA-Z]{2,3}
[a-zA-Z]{2,3} : Désigne une chaîne de caractères suivie de deux à trois caractères.
L'adresse peut aussi également contenir un numéro de port :
^http(s)?://[-[:alnum:]]+\.[-[:alnum:]]+\.[a-zA-Z]{2,4}(:[0-9]+)?$
Les ( ) (Parenthèses) indique respectivement ouverture et fermeture d'un sous-masque ou séquence de caractères.
[0-9] indique que l'on souhaite un chiffre.
? Point d'interrogation Nombre d'occurrences de zéro ou un du caractère ou de la séquence de caractères qui le précède.
$ (Dollar) représente la fin d'une chaîne de caractères.
*/
//déterminer si une variable est bien définie
if (isset($_POST['webAdresse']))
{
// htmlspecialchars() permet d'éviter que des données fournies par les utilisateurs contiennent des balises HTML
$_POST['webAdresse'] = htmlspecialchars($_POST['webAdresse']);

if( ereg('^(http(s)?://[-[:alnum:]]+\.[-[:alnum:]]+\.[a-zA-Z]{2,4}(:[0-9]+)?$', $_POST['webAdresse']) )
{
    echo 'Adresse web valide';
}
else
{
    echo 'Adresse web non valide';
}
}
?>
</p>

<form method="post">
<p>
    <label for="webAdresse">L'adresse de votre site web ?</label> <input id="webAdresse" name="webAdresse" /><br />
    <input type="submit" value="Vérifier l'adresse" />
</p>
</form>
```

Listing 7. Utilisation de la fonction eregi()

```

<?

//lecture d'une url
/*
fopen - ouvre un fichier ou une URL
l'option "r" ouvre notre url en lecture seule, et place le pointeur de
fichier au début du fichier.
*/

$furl = fopen("http://www.phpmag.org", "r");

//on parcourt toutes les lignes
while (!feof($furl)) {

// lecture du contenu de la ligne
$page .= fgets($fp, 4096);

}

//on isole le titre grâce à la balise <title></title>
$titre = eregi("<title>(.*)</title>", $page, $rep);

// on retourne la première occurrence trouvée
echo $rep[1];

// Les occurrences se trouvent entre parenthèses
// $rep[0] renvoie toute la chaîne
fclose($furl);

?>

```

tendance au coup par coup avec un ? mais cela rendra la séquence gourmande . Cette option n'est pas compatible avec Perl. Elle peut aussi être mise dans le masque avec l'option ? U dans le pattern ou par un point d'interrogation avant le quantificateur .

X (PCRE_EXTRA) : Cette option ajoute d'autres fonctionnalités incompatibles avec le PCRE de Perl. Tous les antislashs suivis d'une lettre qui n'aurait pas de signification particulière provoquent une erreur, permettant la réservation de ces combinaisons pour des ajouts fonctionnels ultérieurs. Par défaut, comme en Perl, les antislashs suivis d'une lettre sans signification particulière sont traités comme des valeurs littérales. Actuellement, cette option ne déclenche pas d'autres fonctions.

J (PCRE_INFO_JCHANGED) : L'option (?) interne de configuration modifie l'option locale PCRE_DUPNAMES. Permet la duplication de noms pour les sous-masques.

u (PCRE8) : Cette option désactive les fonctionnalités additionnelles de PCRE qui ne sont pas compatibles avec Perl. Les chaînes sont traitées comme des chaînes UTF-8. Cette option est disponible en PHP 4.1.0 et plus récent sur plate-forme Unix et en PHP 4.2.3 et plus récent sur plate-forme Windows.

Les fonctions POSIX

POSIX est l'acronyme de Portable Operating System Interface dont le X est emprunté à UNIX. Fonctions POSIX Regex :

- ereg_replace - Remplacement par expression rationnelle.
- ereg - Recherche par expression rationnelle standard (Listing 6).
- eregi_replace - Remplacement par expression rationnelle insensible à la casse.
- eregi - Recherche par expression rationnelle insensible à la casse (Listing 7).
- split - Scinde une chaîne en un tableau, grâce à une expression rationnelle.
- spliti - Cette fonction est identique à split(), hormis le fait qu'elle ignore la casse pour les caractères alphabétiques.
- sql_regcase - Prépare une expression rationnelle pour effectuer une recherche insensible à la casse.

PCRE ou POSIX ?

La rapidité d'exécution, cherchons le mot *enfer* dans le texte suivant :

- Un atome de bon sens : Je ne ferai pas de skate-board dans les couloirs.
- Simpsonothérapie : Je ne roterai pas en classe.

- St Lisa Blues : Je ne provoquerai pas la révolution.
- Une soirée d'enfer : Je ne crierai pas au feu en pleine classe.
- L'odyssée d'Homère : Je ne traiterai pas ma maîtresse d' .
- Bart a perdu la tête : Je n'ai pas rencontré Elvis.

La fonction PCRE preg_match('# enfer #' , \$txt, \$out) trouvera la première occurrence du motif approximativement 4 fois plus rapidement que son équivalent POSIX ereg(' enfer ', \$txt, \$out).

Conclusion 1 : Les fonctions PCRE sont plus rapides.

Quantificateur *non gourmand (non greedy)*, essayons de capturer l'url contenue dans la balise suivante :

```

<a href=>http://phpmag.org>PHP
solutions</a>
la fonction POSIX
ereg('<a href=(.*>', $txt, $out), le
point (tout caractère sauf retour ligne) re-
tournera tous les caractères jusqu'au dernier
> rencontré (celui de </a>) c'est-à-dire toute
cette partie «http://phpmag.org>PHP
solutions</a>. Alors que dans la fonction
PCRE on peut demander au point de s'arrêter
au premier > rencontré grâce au point d'inter-
rogation. Ainsi la fonction preg_match('#<a
href=(.*?>#', $txt, $out) retournera
http://phpmag.org.

```

Conclusion 2 : PCRE l'emporte sur POSIX une fois de plus.

Les fonctions PCRE offrent d'autres fonctionnalités bien plus avantageuses que les POSIX telles que :

- Fonctions *Callback*.Capture intégrale des occurrences.
- Utilisation d'assertions, masques conditionnels.
- Etc...

Les PCRE reprennent la norme POSIX en ajoutant de nouvelles fonctionnalités. Avec PHP 6 , les fonctions POSIX vont être abandonnées au profit des PCRE.

DONY CHIQUEL

Dony Chiquel est ingénieur développeur spécialisé en technologies .NET et en conception objet avec UML. Il travaille depuis près de six ans dans le développement d'outils de gestion en tant que prestataire.

Sur Internet

- <http://www.expreg.com/mysql.php> – Le site des expressions régulières en PHP,
- <http://www.php.net/manual/fr/book.regex.php> – Documentation officielle PHP,
- http://lumadis.be/regex/tuto_pcres.php – Les expressions régulières PCRE.

A NE PAS
MANQUER !



Java EE

Guide de développement d'applications web en Java

Auteur : Jérôme LAFOSSE

OFFRE PROMOTIONNELLE sur Editions-eni.fr

-10% sur la version numérique du livre

Code Promotionnel : JAV0510

Offre valable jusqu'au 31/07/2010, non cumulable avec d'autres remises et promotions en cours sur www.editions-eni.fr



En vente sur

www.editions-eni.fr

Livraison Gratuite en France métropolitaine, Corse et Monaco à partir de 25€.



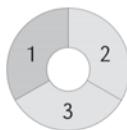
Les nouveautés de PHP 5.3

Dans cet article, nous verrons les nouveautés de PHP 5.3, notamment celles liées à la programmation orientée objet. Nous verrons aussi les précautions à prendre dans le cadre de migration de PHP 5.2 vers PHP 5.3.

Cet article explique :

- Les différences entre PHP 5.2 et PHP 5.3.
- L'utilisation de certaines des nouvelles structures de données basée sur des listes doublément chaînées.
- L'utilisation des fonctions anonymes.
- L'utilisation des espaces de noms.

Niveau de difficulté



Ce qu'il faut savoir :

- Les principes de base de la programmation orientée objet.

Migration de PHP 5.2 vers PHP 5.3

Tout d'abord, voyons quels sont les éléments auxquels il faudra tenir compte lors de la migration de PHP 5.2 vers PHP 5.3 :

- certaines fonctions de tri de tableau qui pouvaient prendre indifféremment des tableaux ou des objets en arguments n'acceptent désormais plus que des tableaux. C'est le cas des fonctions de type `sort()`, `natsort()`, `natcasesort()`, `usort()`, `uasort()`, `uksort()`, ainsi que les fonctions `array_flip()` et `array_unique()`,
- les méthodes magiques introduites dans les versions précédentes doivent désormais être déclarées obligatoirement en tant que méthode publique, et ne peuvent plus être déclarées en tant que méthode `private`, `protected` ou `statique`. Pour rappel, cela concerne les méthodes `__get()`, `__set()`, `__call()`, `__isSet()`, `__unset()`. La gestion des méthodes statiques se fait avec la nouvelle méthode magique introduite dans PHP 5.3 : `__callStatic()`,
- la méthode magique `__toString()` qui permet de convertir un objet en chaîne de caractères n'accepte désormais plus d'arguments,
- PHP 5.3 comporte aussi son lot de fonctions marquées comme étant dépréciées, et qui seront supprimées dans le

futur. C'est le cas des fonctions d'expressions régulières POSIX (`ereg()`, `ereg_*`), mais aussi des fonctions `set_magic_quotes_runtime()`, `magic_quotes_runtime()`, `session_register()`/`session_unregister()`/`session_is_registered()`, `call_user_method()`, et `split()`/`spliti()`, certaines options dans `php.ini` sont aussi dépréciées comme par exemple `magic_quotes_*`, `register_globals`, et `safe_mode`, de nouveaux mots clés sont désormais réservés pour supporter les nouveautés de PHP 5.3 : `goto` et `namespace`.

Pour plus d'informations sur les incompatibilités entre PHP 5.2 et PHP 5.3, vous pouvez consulter la documentation officielle de PHP.

Améliorations et nouveautés de la SPL et de la gestion des dates

Avec PHP 5.3, la SPL (*Standard PHP Library*), qui est la base du support de la POO sous PHP, a connu quelques améliorations. Désormais avec PHP 5.3, la SPL est toujours activée et il est impossible de la désactiver : le but est de favoriser son usage par les développeurs d'applications PHP.

PHP 5.3 ajoute quelques classes supplémentaires à la SPL. Ainsi on retrouve des itérateurs au niveau du système de fichiers : `FilesystemIterator` qui permet de parcourir le système de fichier, et `BlobIterator` qui permet de parcourir des fichiers dont les noms correspondent à un masque. De nouvelles structures de données font aussi leur apparition telles que `SplDoublyLinkedList` qui permet de gérer des listes doublément chaînées. Elle est utilisée par deux autres nouvelles classes de la SPL : `SplStack`, qui implémente

Dans cet article nous allons avant tout nous concentrer sur les nouveautés concernant la programmation orientée objet de PHP 5.3.

une pile de type LIFO (*dernier élément entré, premier élément sorti*), et *SplQueue*, qui implémente une file de type FIFO (*premier élément entré, premier élément sorti*).

Un autre ajout est la classe *SplFixedArray* qui, comme le nom le suggère, est une implémentation de tableau à taille fixe. Elle est très rapide ; en fait, elle est tellement rapide qu'elle a été mesurée pour un gain de 10 à 30% par rapport à l'implémentation *Array* native de PHP. Ce gain de vitesse est dû au fait que le tableau est de taille fixe, contrairement au type *Array* natif de PHP, et que les indexées non numériques ne sont pas permis. Cependant dans le cas où il y a souvent la création de nouvelles listes ne contenant que peu d'éléments, utiliser une instance de *SplFixedArray* se révélera pénalisant car l'instanciation d'un objet est coûteux. Il faut bien noter que la taille d'un *SplFixedArray* doit être définie lors de sa création, il est possible de modifier sa taille avec la méthode *SplFixedArray ::setSize()*.

Il devient aussi beaucoup plus facile de gérer des dates et de faire des calculs avec des dates grâce aux nouveautés ajoutées à la classe *DateTime*. Ainsi on peut désormais ajouter ou soustraire des jours, mois ou années à un objet *DateTime*. Pour cela il convient de passer un objet de type *DateInterval* aux méthodes *DateTime ::add()* et *DateTime ::sub()*. D'ailleurs les classes *DateInterval*, mais aussi *DatePeriod* sont des nouveautés de PHP 5.3. La classe *DateInterval* permet de représenter un intervalle de temps entre deux dates, tandis que la classe *DatePeriod* permet de récupérer la liste des dates d'une période donnée (Listing 1).

On peut aussi calculer l'intervalle de temps entre deux dates avec la méthode *DateTime ::diff()*. Enfin, on peut définir ou récupérer le timestamp UNIX d'une date directement via les méthodes *DateTime ::setTimestamp()* et *DateTime ::getTimestamp()*.

Gestion des archives Phar

Phar(*PHP Archive*) est un format d'archive qui peut être utilisé avec PHP. Avant PHP 5.3, le support du format Phar se trouvait dans une extension PECL, désormais avec PHP 5.3, le support du format Phar devient natif. Le concept des archives Phar provient de la technologie d'archives JAR de Java™, qui permet d'empaqueter une application dans un fichier unique contenant tout le nécessaire pour l'exécuter. À l'inverse des archives JAR, les archives Phar peuvent être utilisées par PHP lui-même et elles ne nécessitent pas d'outil externe pour les créer ou pour les lire. Ainsi une archive *Phar* peut être créée ou éditée directement via la classe *Phar*. Phar offre aussi la possibilité de compresser les fichiers stockés au sein d'une archive, en utilisant

Listing 1. Exemple d'utilisation des nouvelles fonctions de date et de *SplStack*/*SplQueue*

```
<?php
$date_debut = new DateTime('2010-01-01');
$date_fin = new DateTime('2010-03-31');

// Calcul le nombre de jours total entre les 2 dates
$diff = $date_debut->diff($date_fin);

print $date_debut->format('d/m/Y') . ' au ';
print $date_fin->format('d/m/Y') . " il y a ";
print $diff->format("%a jours\n");

var_dump($diff);

// on définit un interval de 2 jours afin d'afficher un jour sur 2
$interval = new DateInterval('P2D');

// on récupère 1 jour sur 2 entre les 2 dates
$period = new DatePeriod($date_debut, $interval, $date_fin);

$pile = new SplStack();
$file = new SplQueue();

// On ajoute les dates dans des SplStack et SplQueue
foreach ($period as $dt)
{
    $pile->push($dt->format("l d/m/Y"));
    $file->push($dt->format("l d/m/Y"));
}

// Affiche le nombre d'éléments dans la pile
print 'Nombre de dates enregistrées : ' . count($pile) . "\n";

// On affiche les dates par ordre ascendant / chronologique
print "\n++++ File FIFO : \n";
foreach ($file as $item)
    echo $item . "\n";

// On affiche les dates par ordre anti-chronologique
print "\n++++ Pile LIFO : \n";
foreach ($pile as $item)
    echo $item . "\n";

// On retire 2 mois
$date_fin = $date_fin->sub(new DateInterval('P2M')) ;

print $date_debut->format('d/m/Y') . ' au ';
print $date_fin->format('d/m/Y') . " il y a ";
print $date_debut->diff($date_fin)->format("%a jours\n");
?>
```

Listing 2. Exemple d'utilisation d'une archive Phar

```
<?php
// On inclut l'archive
include 'monarchive.phar';

// On charge seulement un fichier particulier de l'archive
include 'phar://monarchive.phar/fichier.php';
?>
```

soit la compression *gzip*, soit la compression *bzip2*. Dans un cas comme dans l'autre, c'est au moment de la création de l'archive que nous spécifions qu'il convient de compresser le contenu de celle-ci, à l'aide de la méthode *compressFiles* de la classe *Phar*.

Autre point intéressant, il est possible d'inclure directement une archive Phar au sein du code PHP afin de pouvoir utiliser les ressources contenues au sein de l'archive. Grâce au format *Phar*, le déploiement d'une application PHP s'en trouve grandement simplifié : on peut regrouper plu-

sieurs fichiers PHP au sein d'une archive Phar, copier l'archive sur le serveur, et directement inclure l'archive. Grâce au gestionnaire de flux *phar://*, on peut ensuite inclure un fichier précis contenu dans l'archive Phar (Listing 2).

Gestion des espaces de noms ou Namespaces

Autre nouveauté de PHP 5.3 : la gestion des espaces de noms. Le concept d'espace de noms donne les moyens d'aider à la prévention de problèmes avec plusieurs fonctions, classes

Listing 3. Exemple d'utilisation des espaces de noms

```
<?php
namespace Foo; // définition de l'espace de nom Foo
class Test
{
    public function callMe()
    {
        print "Appel depuis espace de noms Foo\n";
    }
}
namespace Bar; // Définition de l'espace de nom Bar
class Test
{
    public function callMe()
    {
        print "Appel depuis espace de noms Bar\n";
    }
}

use \Bar as PHPsolutions; // On définit un alias pour l'espace de nom Bar
$a = new \Foo\Test();
$b = new PHPsolutions\Test(); // On utilise l'alias sans le préfixer \
\$a->callMe\(\); // affiche "Appel depuis espace de noms Foo"
\$b->callMe\(\); // affiche "Appel depuis espace de noms Bar"
?>
```

Listing 4. Comparaison des closures et des fonctions lambdas

```
<?php
$chaine = "PHP Solutions" ;

// définition d'une fonction lambda
$f_lambda = function() { print 'Hello ' . $chaine . "\n"; };

// définition d'une closure
$f_closure = function() use ($chaine) { print 'Hello ' . $chaine. "\n"; };

$f_lambda(); // Affiche 'Hello '
$f_closure(); // Affiche 'Hello PHP Solutions'
?>
```

Listing 5. Exemple d'utilisation de la méthode __callStatic

```
<?php
class Foo
{
    public static function __callStatic($nom, $args)
    {
        print "La méthode $nom est appelée de manière statique\n";
    }

    public function __call($nom, $args)
    {
        print "La méthode $nom est appelée de manière normale\n";
    }
}

Foo::test(); // affiche "La méthode test est appelée de manière statique"

$foo = new Foo;
$foo->test(); // affiche "La méthode test est appelée de manière normale"
?>
```

Listing 6. Exemple d'utilisation des appels statiques dynamiques

```
<?php
class Foo
{
    public static function test()
    {
        print "ceci est un test\n";
    }
}

$var = 'Foo';
$fonction = 'test';
$var ::$fonction(); // affiche "ceci est un test"
?>
```

et constantes ayant le même nom et définies plusieurs fois. Ceci sera notamment très utile si vous faites cohabiter plusieurs frameworks, et permettra d'être sûr de ne pas avoir des conflits au niveau des noms de fonctions ou de classes entre les frameworks externes, et votre propre application.

Pour définir un nouvel espace de nom, il suffit tout simplement d'ajouter le mot clé *namespace* suivi du nom de l'espace de nom en tout début de votre script PHP ou avant que tout autre caractère ne soit affiché. Il est aussi possible de définir plusieurs espaces de noms au sein d'un même fichier PHP (Listing 3), mais aussi de définir plusieurs niveaux d'espaces de nom.

Une fois l'espace de nom défini, il ne vous reste plus qu'à définir vos fonctions ou classes qui feront parties de votre espace de nom. L'appel des fonctions ou classes d'un espace de nom se fait en utilisant le nom de l'espace de nom suivi d'un anti-slash, puis du nom de la fonction ou de la classe. Par exemple pour appeler la fonction *test()* issue de l'espace de nom PSolutions, il suffit d'utiliser *Psolutions\test()*. Il est possible de définir un alias pour un espace de nom en utilisant le mot clé *use*.

Fonctions anonymes : closures et lambdas

PHP 5.3 introduit un vrai support des fonctions anonymes. Les fonctions anonymes peuvent être stockées dans une variable et appelées en appliquant l'opérateur *()* (Listing 4). De plus il est tout à fait possible de transmettre des arguments en paramètres. Lors de la création d'une fonction anonyme, il conviendra de ne pas oublier d'ajouter un point-virgule à la fin de la définition de la fonction. Bien qu'il était possible de définir des fonctions anonymes avec *create_function()*, cette méthode avait l'inconvénient d'offrir de faibles performances car la compilation avait lieu lors de l'exécution au lieu de lors de la compilation, ce qui empêchait les caches d'opcode de mettre la fonction en cache. De plus la fonction devait être écrite au sein d'une chaîne de caractères ce qui empêchait à la coloration de code de la plupart des IDE de fonctionner.

Pour rappel, une fonction *lambda* est une fonction anonyme : aucun nom ne lui est donné au moment de sa déclaration. Ceci peut notamment se révéler utile lorsque l'on veut appliquer une fonction aux éléments d'un tableau comme par exemple avec la fonction *array_map*.

Une closure est aussi une fonction anonyme mais qui est évaluée dans son propre environnement, qui a une ou plusieurs variables

liées, disponibles au moment où la fonction est appelée. Les closures sont comme des fonctions `lambda` mais elles sont plus intelligentes dans la mesure où elles ont la capacité d'interagir avec des variables hors de l'environnement dans lequel la closure est définie. Pour définir une closure, il suffit de rajouter le mot clé `use` suivi du nom de la variable ou des variables qui devront être disponibles au sein de la closure lors de la définition de la fonction anonyme. Par défaut, ces variables sont transmises par valeur, ce qui signifie que modifier la valeur passée dans la définition de la fermeture ne modifie pas la valeur externe. En préfixant la variable par l'opérateur `&`, celle-ci sera passée par référence et permettra à la closure de modifier la variable externe.

Nouveautés pour la POO

PHP 5.3 contient quelques nouveautés lorsqu'il s'agit de la programmation orientée objet pour PHP. Parmi ces nouveautés on retrouve les appels statiques dynamiques, la liaison statique différée, et enfin la méthode magique `__callStatic()`.

La méthode magique `__callStatic()` fonctionne de manière similaire à la méthode magique `__call()`, qui est prévue pour gérer les appels de méthodes qui ne sont pas définies ou pas visibles dans une classe. Cependant, `__callStatic()` est conçue pour gérer les appels statiques, ce qui nous donne la possibilité de mieux prévoir la surcharge de méthodes. C'est aussi la raison qui explique que la méthode magique `__call()` ne peut plus être définie en tant

Sur Internet

- <http://g-rossolini.developpez.com/tutoriels/php/5.3/> – Introduction aux nouveautés PHP 5.3,
- http://www.ibm.com/developerworks/views/opensource/libraryview.jsp?search_by=new+PHP+V5.3 – Articles d'IBM developerWorks sur les nouveautés de PHP 5.3 (en anglais),
- <http://blog.pascal-martin.fr/post/php-5.3-introduction-sommaire> – Présentation détaillée avec exemples à l'appui des nouveautés de PHP 5.3,
- <http://techportal.ibuildings.com/2010/01/11/learning-php-5.3-by-writing-your-own-orm/> – Apprenez PHP 5.3 en écrivant votre propre ORM,
- <http://fr.php.net/manual/fr/migration53.new-features.php> – Documentation officielle sur les nouveautés de PHP 5.3.

que méthode statique. La déclaration de la méthode `__callStatic()` doit être `public` et `static` (Listing 5).

Les variables de variables sont une fonctionnalité sympathique de PHP. Cela signifie que vous pouvez utiliser la valeur `chaîne` d'une variable pour spécifier le nom d'une autre variable, d'une fonction ou d'une méthode de classe. Cette fonctionnalité est souvent utilisée notamment lorsque l'on fait appel aux méthodes magiques. Une nouveauté de PHP 5.3 est la possibilité d'avoir le nom de la classe en tant que variable lors d'un appel statique ou d'appeler via une variable une méthode statique d'un objet, ainsi on peut appeler une classe et une de ses méthodes statiques via une variable qui contient le nom de la classe (Listing 6) : cela s'appelle l'appel statique dynamique.

Jusqu'à présent, les références statiques, comme celles faites avec `self` ou `__CLASS__`,

étaient résolues dans la même portée que celle dans laquelle la fonction était définie. Le problème est que la référence est incorrecte si la classe est étendue et si l'appel est fait depuis la nouvelle classe `fille`. Late static binding (résolution statique à la volée) a été ajouté à PHP 5.3 pour résoudre ce problème. Les éléments `static::` ne seront plus résolus en utilisant la classe où la méthode a été définie, mais celle qui est active durant l'exécution (Listing 7). L'adjectif statique a été ajouté car ce problème s'applique aux méthodes statiques, mais pas seulement.

Conclusion

Comme vous pouvez le voir, PHP 5.3 apporte de nombreuses nouveautés directement visibles pour le développeur. La programmation orientée objet devient partie intégrante du langage. Nous n'avons pas abordé toutes les nouveautés de PHP 5.3, comme par exemple les chaînes NOWDOC ou la nouvelle constante magique `__DIR__`. Cependant ne serait-ce qu'avec les points que nous avons abordés, il est déjà possible de faire beaucoup de choses. Le déploiement des applications PHP se trouve grandement simplifié avec le support des archives Phar et les espaces de noms. La gestion de la mémoire, notamment pour les scripts ayant un temps d'exécution important a aussi été améliorée avec notamment le ramasse miette circulaire. Pour plus d'informations, n'hésitez pas à consulter la documentation officielle de PHP.

Listing 7. Exemple d'utilisation de la résolution statique à la volée

```
<?php

class Foo

{
    protected static $name = 'Foo';

    public static function test()
    {
        return static::$name . "\n";
    }

    public static function test2()
    {
        return self::$name . "\n";
    }
}

class Bar extends Foo

{
    protected static $name = 'Bar';
}

echo Bar::test();           // affiche 'Foo' car self fait référence à la
                           // classe parente qui l'a définie
echo Bar::test();           // affiche 'Bar'

?>
```

FABRICE FACORAT

L'auteur est administrateur réseaux et intervient occasionnellement pour le CESI afin de donner des formations à des futurs informaticiens. Contributeur Mandriva et spécialiste du monde Linux, il développe depuis une dizaine d'années en PHP après être passé par d'autres langages tels que le C, Java ou encore Visual Basic.

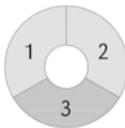
Coupler la puissance de Java EE et PHP grâce à GlassFish

La tendance actuelle est au développement d'applications Internet avec la plate-forme Java EE 6 ou PHP. Mais pourquoi ne pas coupler la puissance de ces deux langages de programmation ?

Cet article explique :

- Le couplage des langages Java EE et PHP.
- L'installation et configuration du serveur GlassFish responsable du couplage.
- Le développement d'une application Java EE 6 / PHP.

Niveau de difficulté



Ce qu'il faut savoir :

- Les bases du langage de programmation PHP.
- Les bases du langage de programmation Java.
- La maîtrise de projets Java EE.
- Des notions d'infrastructures Web pour la mise en place du serveur.

La principale architecture Open Source actuelle pour le développement d'applications Internet de grande envergure est Java EE 6 avec ses API comme : *Servlets*, *JavaServer Pages*, *JavaServer Faces*, *Entreprise JavaBean*, *Java Persistence API* et son serveur standardisé : GlassFish. Le concurrent largement utilisé sur le Web est le langage PHP et le serveur Apache.

Les entreprises développent des projets de toutes tailles à l'aide de ces deux architectures et acquiert de facto des connaissances approfondies dans l'une ou l'autre des technologies. Les développeurs travaillent avec des IDE évolués et utilisent en production le serveur Apache pour l'architecture LAMP et GlassFish pour l'architecture Java EE. Cependant, le serveur GlassFish permet, par l'intermédiaire d'une librairie, de coupler la puissance de Java EE et PHP pour le développement d'applications Internet.

GlassFish, le serveur Java EE de référence

GlassFish (<http://glassfish.org>) est le nom du nouveau serveur Java EE 6 d'applications *Open Source Java EE 6* développé par la société *Sun MicroSystems*. Le serveur GlassFish

est gratuit, libre et distribué sous licence CD-DL et GPL v2 avec *Classpath Exception*. Le mot d'ordre de ce serveur est rapidité et performance. La version 3.0 date de décembre 2009 et propose la gestion de Web Services, des EJB3.1, l'implémentation en standard du moteur de persistance *Java Persistence API* et plusieurs autres outils. Actuellement, GlassFish est le seul serveur totalement compatible Java EE 6, il est d'ailleurs utilisé comme serveur de référence par plusieurs sociétés comme *PSA Peugeot Citroën* et RTL.

Depuis quelques années, les serveurs d'applications sont devenus de plus en plus complexes et offrent un nombre croissant de services. Les fonctionnalités et services proposés ont augmenté en même temps que le coût des serveurs. Le serveur *Sun GlassFish* propose un logiciel serveur pour les applications Web et un développement rapide pour un coût moindre. La dernière version de GlassFish v3 implémente en standard la dernière plate-forme Java EE 6, mais en plus, le serveur GlassFish fournit une plate-forme complète de type LAMP/SAMP (*Sun Apache MySQL PHP*) avec le support multi-langages (PHP, Java, JRuby ou Groovy/Grails). Le serveur GlassFish est également compatible et permet un fonctionnement avec le serveur Apache en frontal (*mod_jk*).

L'administration de GlassFish est basée sur le concept de domaines gérés par un *Domain Administration Server* (DAS). Le DAS permet de gérer le système central contenant les ap-

plications déployées. L'API *Java Management Extensions* (JMX) est disponible pour gérer le serveur et changer sa configuration. À la suite de l'installation du serveur, nous utiliserons le domaine installé par défaut (*domain1*).

Le serveur est livré en standard avec trois outils de gestion et d'administration. L'interface graphique disponible par défaut sur le

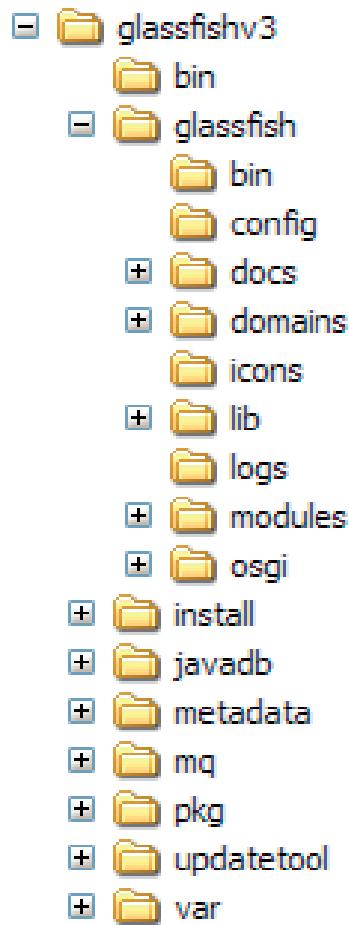


Figure 1. Arborescence du serveur GlassFish

Listing 1. Test du fonctionnement de Quercus

```
Fichier : /WebContent/php.php
<html>
<head>
<title>Premier projet Java EE avec PHP</title>
<?php
    function quercus _ test()
    {
        return function_
exists("quercus _ version");
    }
?>
</head>
<body>
<p>Test de Quercus
</p>
<?php
if (quercus _ test())
{
    echo "Quercus fonctionne
parfaitement";
}
?>
</body>
</html>
```

port 4848 (<http://localhost:4848>) permet de gérer, à partir d'un navigateur Web, le serveur et ses applications. L'interface en ligne de commande nommée *asadmin* (ITALIQUE) permet de réaliser les mêmes opérations. Enfin, l'admin RESTful accessible à l'adresse <http://localhost:4848/management/domain/>.

Installation du serveur GlassFish

Le couplage Java EE 6 / PHP utilise le serveur GlassFish v3 (*sges-v3.zip* ou *glassfish-v3.zip* suivant le dépôt) qui peut être installé à partir d'une archive ZIP, d'un script pour les systèmes Linux, MacOs X et Solaris ou d'un exécutable Windows. Le serveur est téléchargé sur le site officiel du projet GlassFish : <https://glassfish.dev.java.net/public/downloadsindex.html>. Remarque : Le serveur GlassFish Java EE est développé en Java et nécessite l'installation d'un kit de développement Java JDK 1.6 ou supérieur.

L'installation du serveur est simple et peut être réalisée à partir de l'archive multi plates-formes au format ZIP.

- Copier l'archive du serveur GlassFish précédemment téléchargée dans le répertoire des sources.

```
#cp sges-v3.zip /usr/local/src
```

- Se déplacer dans le répertoire des sources.

```
#cd /usr/local/src
```

- Décompresser l'archive du serveur GlassFish.

```
#unzip sges-v3.zip
```

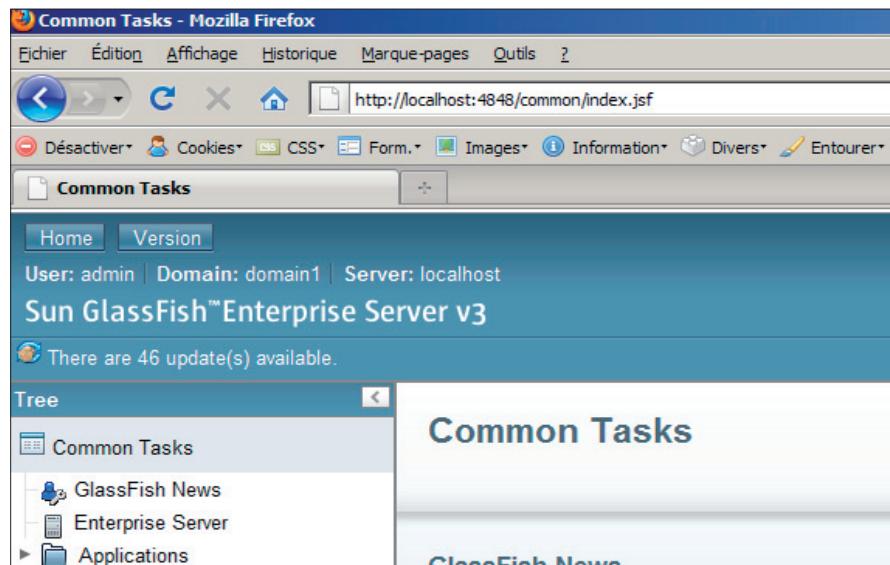


Figure 2. Interface d'administration GlassFish

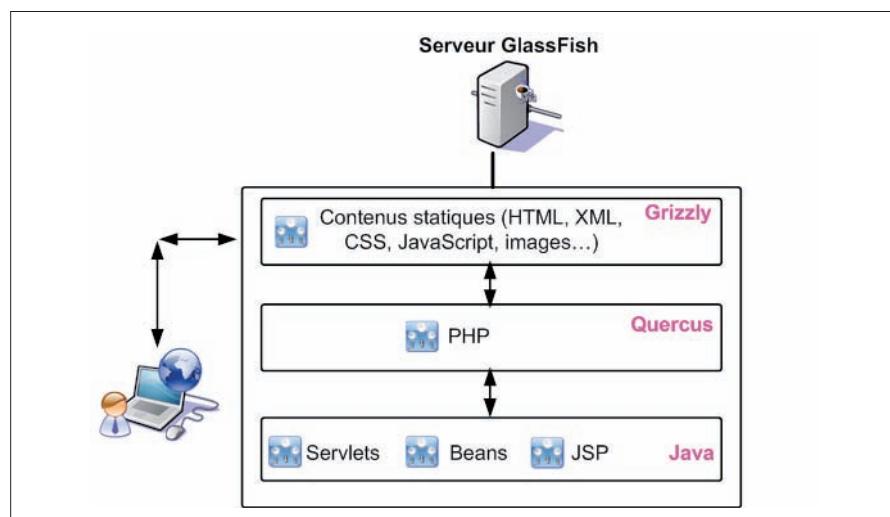


Figure 3. Dialogue Java / PHP avec Quercus



Figure 4. Arborescence du projet

- Déplacer l'archive décompressée du serveur GlassFish dans le répertoire des logiciels.

```
#mv glassfishv3 /usr/local
```

Sous Windows l'installation peut être réalisée également à l'aide de l'archive au format ZIP ou directement avec l'exécutable. Le serveur GlassFish est maintenant correctement installé. Nous devons terminer la configuration en ajoutant

Listing 2. Paramétrage du fichier de configuration de l'application Java EE

```
Fichier : /WEB-INF/web.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/
  xml/ns/javaee/web-app_2_5.xsd" xsi:schemaLocation="http://java.sun.
  com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp _ ID" version="2.5">
  <display-name>projetjavaeephp</display-name>
  <servlet>
    <servlet-name>Quercus Servlet</servlet-name>
    <servlet-class>com.caucho.quercus.servlet.QuercusServlet</servlet-
    class>
    <init-param>
      <param-name>script-encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
      <param-name>license-directory</param-name>
      <param-value>WEB-INF/licenses</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>Quercus Servlet</servlet-name>
    <url-pattern>*.php</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Listing 3. Classe JavaBean Personne

```
Fichier : com.gdawj.Personne
package com.gdawj;
public class Personne {
    private int id;
    private String nom;
    public Personne() {
        System.out.println("Appel du constructeur");
    }
    public String afficherPersonne() {
        return "Bonjour "+this.id+" "+this.nom;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
}
```

Listing 4. Utilisation d'une classe Java en PHP

```
Fichier : /WebContent/personne.php
<?php
    //importation du paquetage Java
    import com.gdawj.Personne;
    //instanciation de la classe
    $pojo=new Personne();
    //declenchement de methodes
    $pojo->setId(1);
    $pojo->setNom("Lafosse");
    echo $pojo->afficherPersonne();

    //importation
    import java.util.Calendar;
    //singleton
    $calendar=Calendar::getInstance();
    //afficher la valeur
    echo("<br>".$calendar);
?>
```

la variable d'environnement `GLASSFISH_HOME` dans le fichier caché `/root/.bashrc` sous Linux en précisant le chemin absolu vers le répertoire d'installation du serveur.

```
export GLASSFISH_HOME=/usr/local/glassfishv3
export PATH=$JAVA_HOME/bin:$GLASSFISH_HOME/bin:$PATH
```

Sous Windows le principe reste le même et il est conseillé d'ajouter la variable d'environnement `GLASSFISH_HOME` avec le panneau de configuration. Cette variable d'environnement permet de référencer directement les exécutables utilisés par le serveur. Remarque : Avec l'installation par défaut du serveur GlassFish, nous utilisons le port 8080 pour le serveur Java EE, le port 4848 pour l'accès à l'interface d'administration.

L'arborescence de GlassFish est présentée dans la Figure 1 et montre les différents dossiers nécessaires pour l'exécution du serveur, la configuration ou la gestion des domaines. L'installation du serveur étant terminée, nous pouvons tester son fonctionnement en lançant directement GlassFish à l'aide de la commande suivante dans un terminal ou une console Ms-DOS : `#asadmin start-domain`

Le serveur est alors accessible à l'adresse suivante `http://localhost:8080/` et l'interface d'administration à l'adresse `http://localhost:4848` (Figure 2).

GlassFish et PHP avec Quercus

Il existe plusieurs approches pour exécuter du code PHP sur le serveur Java EE GlassFish. La première consiste à installer un exécutable comme Quercus, capable d'interpréter du PHP avec la machine virtuelle Java. La seconde consiste à installer un pont entre l'implémentation Java EE et PHP comme JavaBridge. Enfin, la dernière solution consiste à utiliser le serveur Web Apache en frontal et à exécuter les pages PHP avec le module d'Apache. Cette technique est utilisée pour le load balancing mais ne permet pas de coupler/mélanger le code Java avec PHP.

La première solution concerne l'utilisation d'un interpréteur comme Quercus permettant d'utiliser un seul serveur, et surtout de bénéficier du mélange Java / PHP dans les applications comme nous allons le voir ci-après.

Quercus est une technologie *Caucho* sous licence GPL utilisant *Resin* et développée en Java pour l'interprétation du langage PHP. Quercus permet l'intégration de Java dans des services PHP ou scripts mais également une interaction entre le code Java et PHP. Avec Quercus, les applications PHP peuvent ainsi manipuler les technologies Java comme les Entreprise JavaBeans, les frameworks, Java Message-Service et les JavaBeans. Ce concept est rendu possible grâce aux techniques suivantes (Figure 3) :

- Le code PHP est interprété et compilé dans le code Java.
- Quercus est entièrement écrit en Java et donc interprétable par un serveur Java.
- Quercus propose une API de communication Java vers PHP.

Quercus (version 4.0.3 pour cet ouvrage) est livré sous forme de fichier *.war* (Web ARchive ou archive de type *.tar*) contenant l'interpréteur Quercus et les librairies PHP. Ce fichier *.war* peut être déployé sur n'importe quel serveur compatible Java comme GlassFish, Tomcat ou encore JBoss. La mise en place de Quercus commence par le téléchargement de la dernière version du fichier *.war* (<http://quercus.caucho.com>).

La suite consiste à installer la librairie Quercus. L'installation d'une librairie sous GlassFish peut être réalisée de deux façons :

- Pour l'ensemble des applications du domaine en copiant les fichiers *.jar* dans le répertoire */glassfishv3/glassfish/lib/*.
- Pour un domaine particulier en copiant les archives *.jar* dans le répertoire */glassfishv3/glassfish/domains/domain1/lib/ext*.

Nous utilisons la première solution et nous copions les archives téléchargées *resin.jar* et *inject-16.jar* dans le répertoire */glassfishv3/glassfish/lib*. Remarque : L'installation d'une nouvelle librairie nécessite un redémarrage du serveur GlassFish.

Mise en place d'un premier projet Java EE / PHP

Nous allons créer un nouveau projet Java EE de type *Dynamic Web Project* nommé *projetjavaeephp* avec l'IDE Eclipse ou NetBeans. Nous allons ensuite créer un simple fichier PHP (Listing 1) dans le répertoire Web de notre projet avec le contenu suivant permettant de tester le fonctionnement de l'interpréteur.

L'installation est presque terminée, il reste à préciser dans le fichier de déploiement (Listing 2) de l'application Java EE */WEB-INF/web.xml*, l'exécution des fichiers portant l'extension *.php* par Quercus.

Nous pouvons tester notre application déployée avec GlassFish et déclencher le fichier PHP à l'URL suivante : <http://localhost:8080/projetjavaeephp/php.php> (Figure 5). L'arborescence du projet à cette étape est présentée dans la Figure 4.

Utilisation de classes Java en PHP

Afin de présenter la puissance du couplage Java EE / PHP, nous allons créer une première classe simple JavaBean *POJO* nommée *Personne* avec des attributs et méthodes.

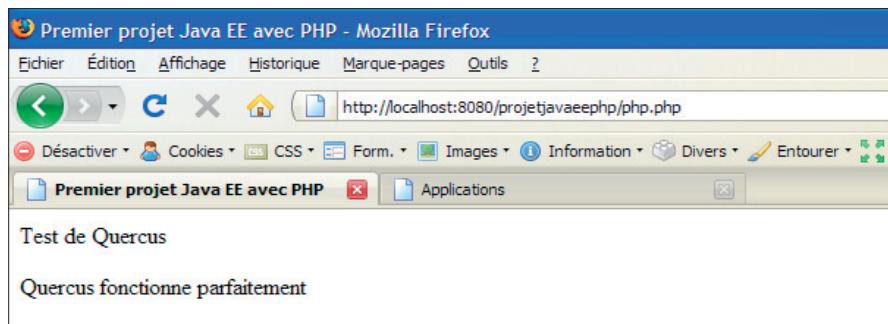


Figure 5. Exécution de code PHP dans un projet Java EE

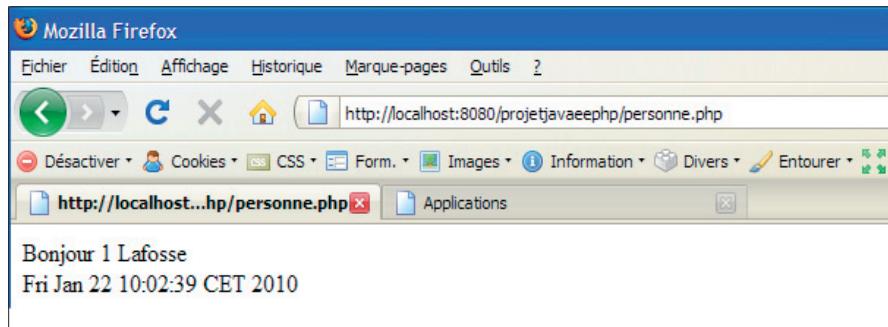


Figure 6. Utilisation d'une classe Java en PHP

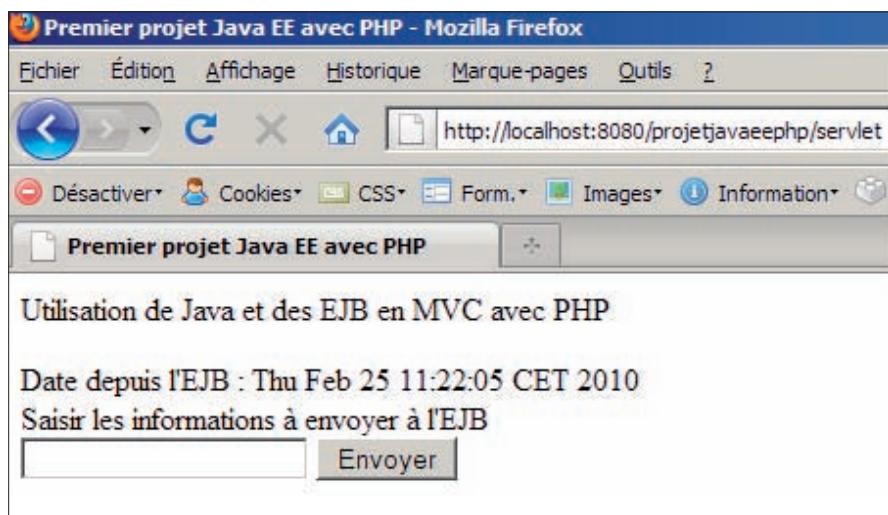


Figure 7. Utilisation des EJB Java avec PHP

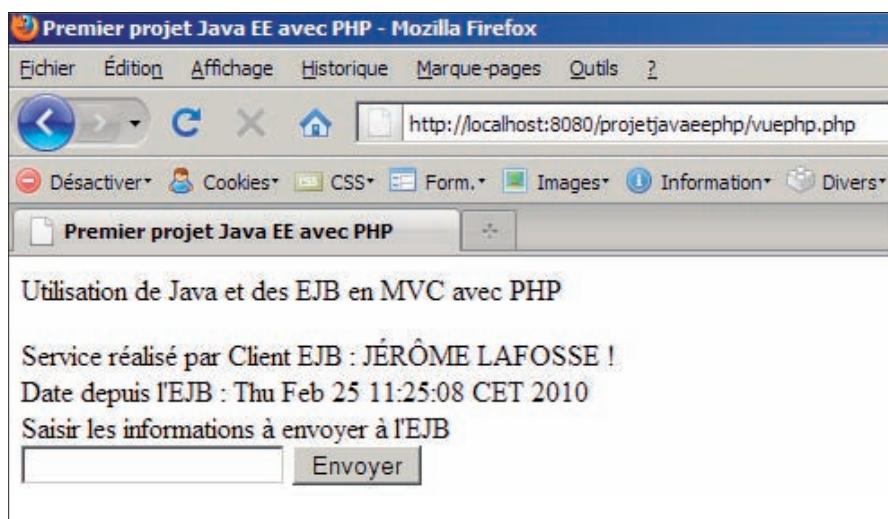


Figure 8. Résultat de l'exécution de l'EJB

Listing 5. Classe statique boîte d'outils

```
Fichier : com.gdawj.BoiteOutils.java

package com.gdawj;

import java.util.Calendar;

public class BoiteOutils {

    public static String afficherDate(){
        Calendar calendar=Calendar.getInstance();
        return calendar.getTime().toString();
    }
}
```

Listing 6. EJB fournissant deux services

```
Fichier : com.gdawj.ejb.ClientEJBBean.java

package com.gdawj.ejb;

import java.util.Calendar;
import javax.ejb.Stateless;

@Stateless
public class ClientEJBBean implements ClientEJB{
    Calendar calendar=Calendar.getInstance();

    public String infoClient(String param){
        return "Service réalisé par Client EJB : "+param.toUpperCase()+" !";
    }

    public String infoDateClient(){
        return "Date depuis l'EJB : "+calendar.getInstance().getTime().toString();
    }
}
```

Listing 7. Servlet contrôleur de gestion du client

```
Fichier : com.gdawj.servlets.ServletClientEJB.java

package com.gdawj.servlets;

import java.io.IOException;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.gdawj.ejb.ClientEJB;

public class ServletClientEJB extends HttpServlet
{
    private static final long serialVersionUID = 1L;

    // utiliser l'EJB
    @EJB
    private ClientEJB clientEJB;

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException
    {
        // chargement de l'EJB Java et envoi à la vue PHP
        if(request.getParameter("saisie")!=null && !request.getParameter("saisie").equals(""))
        {
            String infoEJB=clientEJB.infoClient(request.getParameter("saisie").toString());
            request.setAttribute("infoEJB", infoEJB);
            response.sendRedirect("vuephp.php?infoEJB="+infoEJB);
            return;
        }

        // redirection vers la vue PHP
        getServletContext().getRequestDispatcher("/vuephp.php").forward(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException
    {
        doGet(request, response);
    }
}
```

L'importation de classes Java dans un fichier PHP est réalisée avec l'opérateur *import*. L'appel de méthode Java en PHP respecte la notation pointée. Nous allons utiliser notre classe Java `Personne` (Listing 3) dans un fichier PHP `personne.php` (Listing 4). L'appel de membres statiques et de Singleton est également possible avec PHP. Le résultat de l'exécution est présenté avec la Figure 6.

GlassFish et Quercus permettent une intégration simple et performante de PHP avec Java. Les possibilités de PHP et Java sont multiples et permettent également d'utiliser les pools de connexions JDBC et accès JNDI avec PHP pour les connexions aux bases de données, de réaliser des conversions de type PHP vers Java, de gérer les sessions ou autres.

Ajouter de nouvelles fonctionnalités à PHP

Quercus permet également de déclencher des méthodes Java de classes statiques sans instancier d'objet. La classe Java statique `BoiteOutils.java` présentée dans le Listing 5, permet d'afficher la date courante du système.

Cette classe est déclenchée en PHP avec la méthode `java_class()` permettant de charger des informations statiques.

```
<?php
    $class=java _ class("com.gdawj.
BoiteOutils");
    echo $class->afficherDate();
?>
```

Cette technique est très largement utilisée pour proposer de nouvelles fonctionnalités à PHP en profitant de la puissance du langage Java. Par exemple, la manipulation de flux d'octets est largement détaillée et documentée en Java. Elle pourra ainsi être utilisée à l'aide de PHP. Remarque : Pour les exemples qui suivent nous utilisons les librairies Quercus installées dans le répertoire `WEB-INF/lib` de notre projet Java EE afin de pouvoir y faire référence et importer les classes.

Utiliser des composants Java EE avec PHP

Les *Entreprises JavaBeans* (EJB) sont largement utilisés pour la gestion de la couche métier en Java EE et permettent de séparer la partie modèle du design pattern MVC de façon optimisée. Les EJB sont par nature transactionnels et donc parfaitement adaptés pour réaliser les accès aux SGBDs. Ce type de technologie n'existe pas en PHP mais il est tout à fait possible de profiter de la puissance des EJB Java en PHP. Les développeurs peuvent ainsi coder en Java EE et les intégrateurs en PHP.

Listing 8. Vue PHP utilisant les services d'un EJB Java

Fichier : `/vuephp.php`

```
<html>
<head>
<title>Premier projet Java EE avec PHP</title>
</head>
<body>
<p>Utilisation de Java et des EJB en MVC avec PHP</p>
<?php
// afficher les informations de l'EJB Java EE dans du PHP
$resultatClientEJB=$ _ REQUEST["infoEJB"];
if(!empty($resultatClientEJB))
{
echo $resultatClientEJB."<br/>";
}

// utiliser le client EJB directement
$clientEJB=jndi _ lookup("java:global/projetjavaepphpEAR/
projetjavaepphpEJB/ClientEJBBean");

echo $clientEJB->infoDateClient()."<br/>";

?>

<form name="formulaire" action="servlet" method="POST">
Saisir les informations &agrave; envoyer &agrave; l'EJB<br/>
<input type="text" name="saisie" id="saisie"/>
<input type="submit" value="Envoyer"/>
</form>
</body>
</html>
?>
```

Listing 9. Classe JavaBean ImageMiniature

Fichier : `com.gdawj.ImageMiniature.java`

```
package com.gdawj;

public class ImageMiniature {

    private String chemin;
    private int largeur;
    private int hauteur;

    public ImageMiniature()
    {
        System.out.println("Appel du constructeur
ImageMiniature");
    }

    public String getChemin() {
        return chemin;
    }

    public void setChemin(String chemin) {
        this.chemin = chemin;
    }

    public int getLargeur() {
        return largeur;
    }

    public void setLargeur(int largeur) {
        this.largeur = largeur;
    }

    public int getHauteur() {
        return hauteur;
    }

    public void setHauteur(int hauteur) {
        this.hauteur = hauteur;
    }
}
```

Listing 10. Page PHP responsable de la génération de l'image

```
Fichier : /image.php
<?php
// importation du paquetage Java
import com.gdawj.ImageMiniature;

// instantiation de la classe Java
$pojo=new ImageMiniature();

// déclenchement de méthodes
$pojo->setChemin("images/image.jpg");

// afficher l'image d'origine
echo "Image d'origine<br/><img src=\"".$pojo->
getChemin()."\"/>";

// retailler l'image avec PHP GD
$pojo->setHauteur(150);
$pojo->setLargeur(150);

// recuperer la source avec Java
$source=imagecreatefromjpeg($pojo->getChemin());
// créer la vignette avec les infos du pojo
$destination=imagecreatetruecolor($pojo->getHauteur(),
$pojo->getLargeur());

// les fonctions imagesx et imagesy renvoient la
largeur et la hauteur d'une image
$largeur_source=imagesx($source);
$hauteur_source=imagesy($source);
$largeur_destination=imagesx($destination);
$hauteur_destination=imagesy($destination);

// on cree la miniature
imagecopyresampled($destination, $source, 0, 0, 0,
$largeur_destination, $hauteur_destination,
$largeur_source, $hauteur_source);

// on enregistre la miniature sous le nom "miniature.jpg"
imagejpeg($destination, "miniature.jpg");

// afficher l'image miniature
echo "<br/>Image miniature<br/><img src=\"miniature.jpg\"/>";
?>
```

Pour cet exemple, nous utilisons une classe EJB Java EE nommée ClientEJBBean (Listing 6) fournit deux services simples. Dans un cas professionnel, ces services pourraient accéder à un SGBD ou un ERP. Ce code représente la couche *Modèle* du design pattern MVC.

Nous allons maintenant dans le Listing 7, présenter la couche *Contrôleur* avec une *Servlet* nommée *ServletClientEJB*. Cette *Servlet* invoque l'EJB à partir du paramètre entré dans la vue PHP. Enfin, la dernière partie (Listing 8) est codée en PHP et représente la couche *Vue* du modèle MVC. Cette page déclenche la *Servlet* et affiche le résultat de l'exécution de l'EJB Java en PHP. Nous utilisons également JNDI pour appeler direc-

tement l'EJB depuis le code PHP. Le résultat est présenté dans les Figures 7 et 8.

Remarque : L'*API Java Naming and Directory Interface* (JNDI) fournit un mécanisme de nommage de type annuaire pour l'accès aux ressources. Ces ressources peuvent être de différents types mais le but étant d'associer les objets à un nom (*bind*) et de retrouver ces objets (*lookup*) dans un annuaire de nommage semblable à LDAP, DNS ou NIS.

Profiter de la souplesse de PHP en Java

Le paragraphe précédent montre le principe d'utilisation de Java avec PHP mais il est également possible d'utiliser des techniques PHP avec Java. Par exemple, la librairie de graphisme GD est très simple à utiliser en PHP et peut être ajoutée à Java.

Nous allons dans cet exemple utiliser PHP et la librairie GD à l'aide d'une classe Java afin de générer une image à la volée. Pour cela, nous allons créer une nouvelle classe Pojo Java nommée *ImageMiniature.java* (Listing 9) composée de ses attributs et accesseurs.

La page PHP nommée *image.php* utilise le POJO Java pour les informations et génère une miniature de l'image source à l'aide de PHP/GD.

Conclusion

La plate-forme Java EE 6 et le langage PHP ont actuellement le monopole pour les développements Internet Open Source, et il est courant de trouver des codeurs pro Java EE ou PHP. Java EE est utilisé pour les grosses infrastructures et permet de gérer des pools de connexion, *Enterprise JavaBean*, Java Persistence API ou encore des *WebServices* évolués quand PHP permet un développement rapide de vues ou l'utilisation simplifiée de librairies pour la manipulation d'images par

exemple. Le serveur GlassFish Open Source totalement compatible Java EE 6, permet de coupler la puissance de ces deux langages.

De même, l'exécution des applications PHP avec un serveur d'applications comme GlassFish permet d'augmenter les performances, la sécurité et de rationaliser l'utilisation des serveurs.

Un projet existant pourra par exemple migrer de PHP vers Java EE, ou conserver les meilleurs services de chaque langage afin d'optimiser le produit. Ainsi dans le cas d'une migration totale de PHP vers Java EE, le passage pourra être progressif et réalisé service par service. Enfin, dans le meilleur des cas, les services optimisés pour Java seront développés avec Java EE et les autres resteront en PHP. Plus rien ne nous empêche désormais de développer des projets en MVC I ou II avec Java EE pour les parties *Contrôleur* et *Modèle* et de conserver la couche *Vue* en PHP/XHTML.

JÉRÔME LAFOSSE

Ingénieur en informatique et diplômé du CNAM, Jérôme Lafosse intervient comme consultant, concepteur et formateur sur les technologies Java. Spécialiste des technologies web, il travaille à promouvoir les outils et solutions Open Source pour le développement de projets Internet. Il enseigne également la plate-forme Java Entreprise Edition et la conception de projets web en Licence et Master. Son expérience pédagogique s'allie à ses compétences techniques et offrent au lecteur un guide de réellement opérationnel sur le développement d'applications web en Java. Auteur des ouvrages Java EE – Guide de développement d'applications web en Java et Struts 2 Le framework de développement d'applications Java EE, Jérôme Lafosse propose son site dédié aux langages du web à cette adresse : <http://www.gdawj.com> et travaille sur un nouvel ouvrage consacré à Java EE.

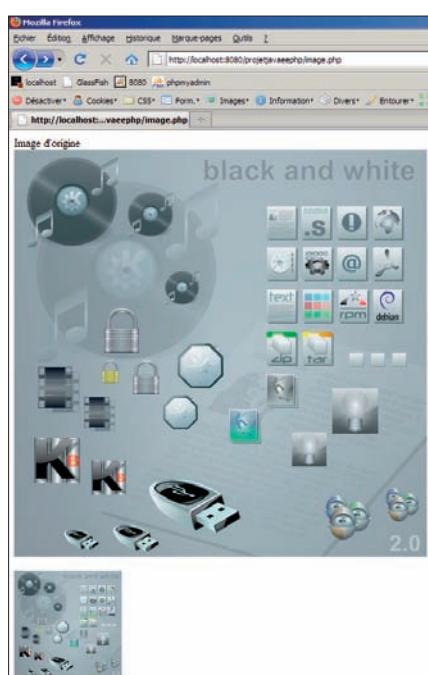


Figure 9. Génération d'images avec Java, PHP et GD

Framework PHP Vulnérabilités Bibliothèques Flash Flex

Sites Web Gestion de fichiers Configuration **HTML**

Bases de données MySQL ASP Application XML CSS Javascript

Margin **Programmation** Forums Classes Animation Flash

[Attributs](#) [JSP](#) [Feuille de Style](#) [Compilation](#) [Tutoriaux](#)

Assets

Connaissez-vous aussi les meilleurs logiciels du Web ?

Résumé

Top-logicie®

Pro www.top-logiciel.net

Fairness is a central value in our society, and it's important to ensure that everyone has equal opportunities and rights.

Languages de programmation

Test Automatisme Propriétés Programmation objets

Paradigme Infographie | Actualité du web | Actualité informatique, dossiers & téléchargements sur Top-logiciel

Les meilleurs logiciels du web : [Avalon](#) | [Google](#)

http://www.top-logiciel.net/ | Actualité informatique, dossiers & téléchargements

www.top-logiciel.net Les meilleurs logiciels du web : Actualité informatique

[Accueil](#) [Articles](#) [Tutoriaux](#) [Liens Web](#) [Telechargements](#) [Forums](#) [Emploi](#) [Partenaires](#) [Contact](#)

[Accueil](#) | [Forum](#)

[Service informatique](#) | [Accès à la personne](#), dépannage à domicile pour

Rechercher sur le site... OK Logiciel CAO 3D Diminez Vos Coûts De Conception ! Adoptez Notre Logiciel CAO 3D service à la personne, dépannage 30€ www.lasci-micro.com

Pseudo Logiciel de CAO 3D
www.cadware.fr/CAO Annonces Google

Mot de passe : Inscrivez-vous maintenant ! Connexion Années Google

[Logiciel Planning](#)
[Logiciel Informatique](#)

Annonces Google Logiciels Gratuits
Telecharger Des Films Accéder à Internet

Logiciel HAD Telesante Plateforme de Coordination de soins

Coordination de serveur
Mode Hébergé, Full Web,
SaaS

www.arcan.fr

Télécharger un fichier depuis un script PHP

Cet article présente trois méthodes alternatives de téléchargement de fichiers à partir d'un script PHP. La première méthode utilise des fonctions PHP natives, la seconde emploie la bibliothèque gratuite et open source CURL, la dernière établit une communication par sockets.

Cet article explique :

- Comment télécharger un fichier sur le serveur.

Ce qu'il faut savoir :

- Vous devez connaître les bases du langage PHP.

Niveau de difficulté



Vous avez appris, dans les précédents articles de cette série sur les fichiers, comment lire et écrire dans des fichiers situés sur le disque dur du serveur web et comment manipuler les fichiers reçus par le biais d'un formulaire HTML. PHP permet également de télécharger des fichiers situés sur des serveurs distants, afin de les stocker, les analyser et incorporer leurs données dans le résultat envoyé au navigateur. Il est ainsi possible de récupérer automatiquement des pages web externes, des images, etc. Cet article présente trois méthodes alternatives de téléchargement de fichiers distants, depuis un script PHP.

Afin de mieux comprendre les mécanismes de téléchargement utilisés, l'article présente dans un premier temps l'architecture client/serveur et le protocole HTTP. Chaque méthode de téléchargement est ensuite présentée avec ses avantages et ses inconvénients. Enfin l'article montre comment implémenter un téléchargement de fichier distant fonctionnant dans la majorité des configurations.

Architecture client/serveur

Un client est une application qui émet une requête vers une autre application atteignable par le réseau et qui attend sa réponse. Un

serveur est une application qui attend que des requêtes soient émises vers un port dédié, qui traite la requête et envoie une réponse au client (Figure 1). Dans le cadre du web, le client est généralement un navigateur.

Un serveur peut fournir plusieurs services (web, base de données, ftp, ssh, mail,...). Lorsqu'un client veut utiliser un service, il doit envoyer une requête vers le nom d'hôte ou l'adresse IP du serveur (identifiant unique sur Internet) en précisant un numéro de port (identifiant unique de l'application sur le serveur demandé, à laquelle les données sont destinées). Le numéro de port est un entier codé sur 16 bits, tout ordinateur a donc 65536 ports disponibles. Les services disponi-

nables sur un serveur ont tous un numéro de port, celui-ci peut être modifié par l'administrateur système dans le fichier de configuration de chaque service. Les ports des services les plus courants ont été fixés par le IANA (*Internet Assigned Numbers Authority*) : 80 pour HTTP, 21 pour FTP, ... Par exemple, lorsque le navigateur émet une requête vers un serveur web, il communique par défaut sur le port 80. Le serveur reçoit des données destinées au port 80, il les transmet à l'application serveur Web (Apache). La communication entre le serveur web et le client utilise le protocole de transport HTTP.

Protocole HTTP

Le protocole de transport HTTP (*HyperText Transfer Protocol*) est utilisé pour l'échange de données entre un client et un serveur web. Le client émet une requête HTTP vers le serveur web, celui-ci la traite et envoie une réponse. Le client peut être un navigateur ou bien un autre serveur. Les requêtes et les réponses comportent deux parties toujours séparées

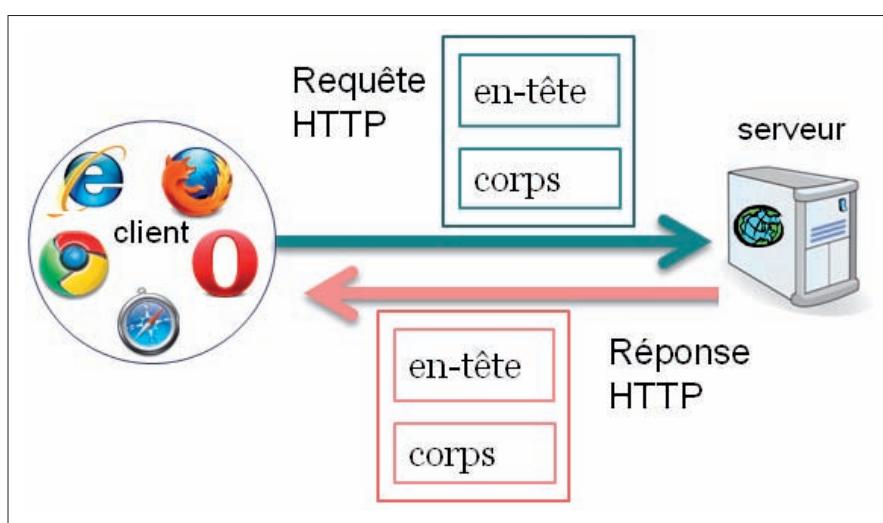


Figure 1. Architecture client/serveur web

par deux sauts de lignes (afin d'obtenir une ligne vide) : l'en-tête et le corps.

Requête HTTP

La requête est composée d'une ou plusieurs lignes d'en-tête et éventuellement d'un corps, en fonction de la méthode HTTP utilisée (GET ou POST). A l'exception de la première ligne, chaque ligne d'en-tête comporte un champ HTTP, le caractère deux points et la valeur associée au champ. Il ne peut y avoir qu'un seul champ HTTP par ligne d'en-tête.

La première ligne de l'en-tête de la requête précise la méthode HTTP à appliquer, l'adresse de la ressource et la version du protocole HTTP. Dans la version 1.0, la connexion est fermée après la réponse du serveur. Avec la version 1.1, la connexion reste établie. En fonction du choix de la version du protocole HTTP, la partie adresse de la première ligne de l'en-tête diffère. En 1.0, l'adresse correspond à l'URL de la ressource. En 1.1 il faut indiquer le chemin de la ressource à partir de la racine du serveur web (*document root*). Une seconde ligne d'en-tête, préfixée par `Host`, est alors nécessaire pour indiquer le nom du serveur web.

Les deux requêtes suivantes sont identiques, elles demandent au serveur factice *monEphemeride.com* la ressource *info_soleil.php*, par la méthode GET. La première requête utilise la version 1.0 du protocole :

```
GET http://monEphemeride.com/info_soleil.php HTTP/1.0
```

La seconde requête utilise la version 1.1 :

```
GET /info_soleil.php HTTP/1.1
Host:monEphemeride.com
```

Ces deux requêtes retournent la page web générée par le script PHP *info_soleil.php*. Par défaut ce script, s'il ne reçoit pas de données en paramètres, retourne l'heure du coucher et du lever du soleil à Paris pour le jour courant.

L'en-tête de la requête HTTP peut contenir d'autres informations optionnelles, sur le client (préférences de langue, jeux de caractères, ...) et sur le corps (encodage, ...).

Envoyer d'arguments à la ressource

Le script *info_soleil.php* peut prendre en arguments deux données, la zone géographique dans le monde (*Europe, Asia, America, Africa, ...*) et un nom de ville en anglais située dans la zone demandée. Pour envoyer des données à une ressource, il faut les formater sous la forme de paires `nom=valeur`, séparées par des caractères &. Par exemple pour obtenir l'éphéméride de la ville de Rome (Europe), la chaîne d'arguments est :

```
ville=Rome&zone=Europe
```



Figure 2. Méthode HTTP GET

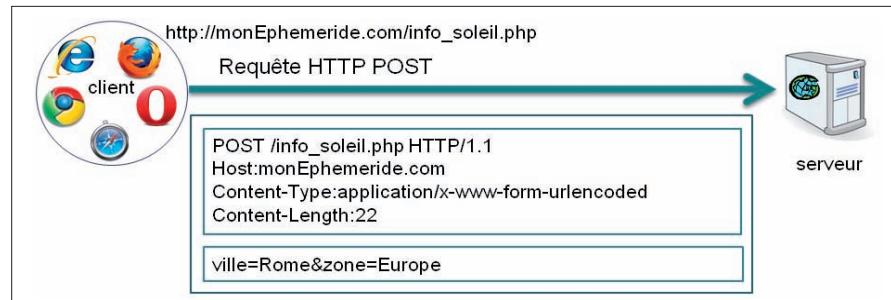


Figure 3. Méthode HTTP POST

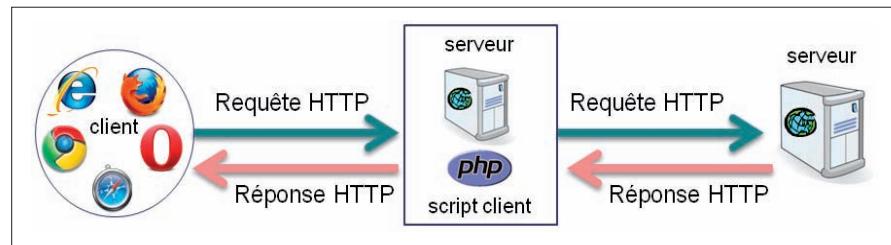


Figure 4. Architecture utilisée dans l'article

Dans le cas de la méthode GET, cette chaîne de données est envoyée dans la partie argument (*query string*) de l'URL. Les données sont placées après le point d'interrogation et le corps de la requête est vide (Figure 2). La ligne suivante utilise la méthode GET et la version 1.1 du protocole HTTP, pour demander les informations sur les heures de lever et coucher de soleil à Rome en Europe :

```
GET /info_soleil.php?ville=Rome&zone=Europe HTTP/1.1
Host:monEphemeride.com
```

Avec la méthode POST (Figure 3), les données doivent être placées dans le corps de la requête et des informations sur la taille et le type des données doivent être précisées dans l'en-tête. Le champ `Content-Length` de l'en-tête HTTP

Listing 1. *fonctions_url_open.php*

```
<?php
/**
 * Telecharge une ressource a partir d'un serveur distant en utilisant une
 * fonction PHP native
 * @param $url (string) URL de la ressource sur le serveur distant
 * @param $donnees (array) tableau de donnees a passer en parametres au
 * serveur distant
 * @param $resultat (string) contenu du fichier (var affectee par
 * reference dans la fonction)
 * @param $erreur (string) message d'erreur eventuel (var affectee par
 * reference dans la fonction)
 * @return boolean true si la ressource a ete telechargee, false sinon
 */

function utiliserURLopen($url, $donnees, &$resultat, &$erreur){
    // generer la query string
    $arguments = @http_build_query($donnees);
    // recuperer le fichier sur le serveur distant
    $resultat = @file_get_contents($url.'?'.$arguments);
    if (!$resultat) {
        $erreur = 'Impossible de recuperer le fichier';
    }
    return empty($erreur);
}
?>
```

Listing 2. methode_1.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Heures de coucher/lever du soleil</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Premiere methode : fonction native PHP</h1>
    <?php
      // inclusion de fonctions
      require_once 'fonctions_url_fopen.php';
      // initialisation de variables
      $resultat = '';
      $erreur = '';
      $url = 'http://monEphemeride.com/info_soleil.php';
      // chercher et afficher les horaires pour Rome
      $donnees = array('ville' => 'Rome', 'zone' => 'Europe');
      if (utiliserURLFopen($url, $donnees, $resultat, $erreur)) {
        echo "<div class='resultat'>$resultat</div>";
      } else {
        echo "<div class='erreur'>$erreur</div>";
      }
    ?>
  </body>
</html>
```

permet de préciser la taille du corps en octets. Le champ Content-Type définit le type MIME des données contenues dans le corps de la requête. Dans le cas d'envoi de données par la méthode POST, encodées sous la forme de paires nom=valeur, ce champ doit prendre la valeur application/x-www-form-urlencoded.

Le code suivant envoie les mêmes données que précédemment, mais en utilisant la méthode POST. Les arguments soumis au script sont dans le corps de la requête (partie après le saut de ligne, suivant le dernier champ de l'en-tête Content-Length). La taille des données dans le corps est de 22 octets :

```
POST /info_soleil.php HTTP/1.1
Host:monEphemeride.com
Content-Type:application/x-www-form-urlencoded
Content-Length:22

ville=Rome&zone=Europe
```

GET versus POST

Les méthodes GET et POST n'envoient pas les données au script dans la même partie de la requête HTTP. Pour des raisons d'ergonomie et de sécurité, il est important de savoir quand utiliser ces deux méthodes.

La méthode GET contient dans l'URL les données envoyées au script. Ces données, qui apparaissent dans la barre de navigation, peuvent être stockées dans le navigateur (historique de navigation, signets), sur un serveur proxy et dans les fichiers de log du serveur web. Les données peuvent donc être facilement interceptées par un tiers sur le client, dans le cas d'un ordinateur partagé par plusieurs utilisateurs, ou sur les serveurs. La méthode GET doit être utilisée uniquement pour envoyer des données non confidentielles. Cette méthode est réservée à l'interrogation d'un site et à des opérations de lecture. Elle ne doit jamais être utilisée pour des opérations de mise à jour de données situées sur

le site web. En effet, l'utilisation de GET rend vulnérable aux failles de type CSRF (*Cross Site Request Forgeries*). Ces failles consistent à faire envoyer une requête malicieuse à un site web (suppression d'article dans la base, modification de données, ...), par un utilisateur légitime de ce site.

La méthode POST envoie les données dans le corps de la requête HTTP. Les données ne sont donc pas visibles dans la barre de navigation et ne sont pas stockées dans les logs des serveurs. Cette méthode doit être utilisée lors de l'envoi de données confidentielles, il faut bien entendu dans ce cas utiliser le protocole HTTPS (données cryptées) afin de protéger les données d'une écoute réseau (interception de la requête HTTP). Lorsque les données sont utilisées pour effectuer une opération de mise à jour sur le site web, il faut impérativement utiliser la méthode POST.

En résumé, la méthode GET est utile pour envoyer des données destinées à interroger un site (moteur de recherche, article sur un site marchand, ...). Cette méthode permet de placer des signets qui comporteront les données recherchées, ce qui représentera un gain de temps lors des recherches futures. La méthode POST quant à elle doit être utilisée pour toute opération qui met à jour des données sur le site web, ou pour assurer la confidentialité des données.

Réponse HTTP

L'en-tête de la réponse du serveur contient des informations sur le corps et le serveur. La première ligne de la réponse précise la version du protocole HTTP utilisée par le serveur, ainsi qu'un code à trois chiffres indiquant le statut de la réponse accompagné d'une phrase en anglais expliquant le code.

Par exemple la ligne d'en-tête suivante indique que la ressource a été trouvée, et que le serveur qui l'héberge utilise la version 1.1 du protocole HTTP :

```
HTTP/1.1 200 OK
```

Les codes commençant par 3 indiquent une redirection. Par exemple lorsque la ressource a été déplacée, le code 301 est retourné. Les erreurs commises par le client commencent par le chiffre 4, par exemple le code 404 quand la page demandée n'existe pas. Les erreurs côté serveur commencent par le chiffre 5, par exemple le code 500 est retourné lorsque le serveur rencontre une erreur interne.

Les lignes suivantes de l'en-tête donnent des informations sur le corps de la réponse HTTP (type, encodage, langue, taille,...) et sur le serveur.

Le corps de la réponse est optionnel, en fonction de la méthode HTTP utilisée. Cer-



Figure 5. Résultat du téléchargement avec file_get_contents

taines méthodes qui ne sont pas présentées dans cet article (HEAD,...) ne retournent pas de corps. Tous les exemples retournent une réponse HTTP composée de l'en-tête et du corps.

Script PHP client d'un serveur Web

Dans cet article un script PHP situé sur un serveur web se comporte comme un client d'un autre serveur Web (Figure 4). Le navigateur envoie une requête au script PHP situé sur le serveur client. Celui-ci demande une ressource (image, son, document PDF,...) à un autre serveur Web. La ressource renvoyée par ce dernier devient disponible dans le script PHP client. Le script client peut alors traiter ou enregistrer la ressource sur le disque du serveur Web client. Ceci est l'équivalent d'un téléchargement de fichier.

PHP fournit trois méthodes pour réaliser le téléchargement d'un fichier situé sur un autre serveur Web. La première utilise des fonctions PHP natives pour les fichiers. La seconde permet d'obtenir une ressource avec la bibliothèque CURL. Enfin la troisième méthode établit une connexion par socket.

Dans cet article tous les scripts PHP donnés en exemple sont des clients qui téléchargent la ressource *info_soleil.php* sur le serveur factice *monEphemeride.com*. Chacun des trois exemples présente une des méthodes offerte par PHP. Enfin un dernier exemple complet effectue un téléchargement en utilisant la première méthode disponible, en fonction des réglages du serveur Web hébergeant le script PHP client.

Utilisation de fonctions PHP natives

Il est possible de télécharger un fichier situé sur un serveur Web distant en utilisant des fonctions PHP natives. Les fonctions *file_get_contents* et *file* permettent de récupérer tout le contenu d'un fichier et de le stocker dans une variable. La première retourne une chaîne de caractères, la seconde un tableau avec une case par ligne de fichier. Ces deux fonctions prennent en paramètre le chemin sur le disque du fichier à charger et s'occupent intégralement de l'ouverture du fichier, de l'extraction des données et de la fermeture du fichier. Il est possible de passer en paramètre une URL pointant sur une ressource située sur un serveur Web externe. Dans l'exemple ci-après, le code HTML retourné par le fichier *info_soleil.php* est stocké dans la variable *\$resultat*:

```
$url = 'http://monEphemeride.com/
info_soleil.php';
$resultat = @file_get_contents($url);
```

Listing 3. *fonctions_curl.php*

```
<?php

/**
 * Definit les options d'envoi et l'url
 * @param $curl_ref (ressource) session curl
 * @param $donnees (array) tableau de donnees a passer en parametres au serveur
distant
 * @param $url (string) URL de la ressource sur le serveur distant
 * @param $methode (string) methode d'envoi de la requete HTTP
 */

function setOptions($curl_ref, $donnees, $url, $methode){
    // ne pas envoyer la reponse au navigateur, la retourner dans une chaine
    @curl_setopt($curl_ref, CURLOPT_RETURNTRANSFER, true);
    if ($methode == 'POST') {
        // tableau des options de transfert
        $options = array(CURLOPT_URL => $url,
                         CURLOPT_POST => true,
                         CURLOPT_POSTFIELDS => $donnees );
        // definir les options curl pour la session courante
        @curl_setopt_array($curl_ref, $options);
    } else { // methode GET par defaut
        // initialiser les arguments envoyees avec la methode GET
        $arguments = http_build_query($donnees);
        // fixer l'URL de la ressource
        @curl_setopt($curl_ref, CURLOPT_URL, $url.'?'.$arguments);
    }
}

/**
 * Telecharge une ressource a partir d'un serveur distant avec la bibliotheque
curl
 * @param $url (string) URL de la ressource sur le serveur distant
 * @param $donnees (array) tableau de donnees a passer en parametres au
serveur distant
 * @param $resultat (string) contenu du fichier (var affectee par reference
dans la fonction)
 * @param $erreur (string) message d'erreur eventuel (var affectee par
reference dans la fonction)
 * @param $methode (string) methode d'envoi de la requete HTTP (methode GET
par defaut)
 * @return boolean
 */

function utiliserCurl($url, $donnees, &$resultat, &$erreur, $methode = 'GET') {
    // initialiser la session curl
    if ($curl_ref = curl_init()) {
        // fixer les options (URL, interception de la reponse)
        setOptions($curl_ref, $donnees, $url, $methode);
        // envoyer la requete et recuperer le corps de la reponse HTTP
        $resultat = @curl_exec($curl_ref);
        // recuperer le code de la reponse dans l'entete HTTP
        $code_http = @curl_getinfo($curl_ref, CURLINFO_HTTP_CODE);
        // si erreur
        if ($code_http != '200') {
            $erreur = 'Fichier non trouve ('.$code_http.')';
        }
        // liberer les ressources
        @curl_close($curl_ref);
    } else {
        $erreur = 'Echec initialisation curl';
    }
    return empty($erreur);
}
?>
```

Cette méthode de téléchargement est la plus simple des trois méthodes proposées dans cet article mais est cependant limitée. En effet si des arguments doivent être envoyés au serveur externe hébergeant la ressource, seule la méthode d'envoi par GET est disponible. Ainsi il est possible d'envoyer des données au serveur pour obtenir la ressource voulue, mais uniquement en passant les données dans la partie argument (*query string*) de l'URL don-

née en paramètre à la fonction *file_get_contents* ou *file*.

De plus cette méthode de téléchargement est disponible uniquement si la directive du *php.ini* *allow_url_fopen* a la valeur *on*. Cette directive permet aux deux fonctions *file* et *file_get_contents* de lire un fichier situé sur un serveur distant. Lorsqu'elle a la valeur *off*, seuls des fichiers hébergés sur le même serveur que le script

Listing 4. methode_2.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Heures de coucher/lever du soleil</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Deuxieme methode : utilisation de la bibliotheque <em>CURL</em></h1>
        <?php
        // inclusion de fonctions
        require_once 'fonctions_curl.php';
        // initialisation de variables
        $resultat = '';
        $erreur = '';
        $url = 'http://monEphemeride.com/info_soleil.php';
        // recuperer les heures du lever et du coucher de soleil pour
Londres (methode GET)
        $donnees = array('ville' => 'London', 'zone' => 'Europe');
        if (utiliserCurl($url, $donnees, $resultat, $erreur)) {
            echo $resultat;
        } else {
            echo "<div class='erreur'>$erreur</div>";
        }
        // reinitialiser les variables
        $resultat = '';
        $erreur = '';
        // et pour Oslo (methode POST)
        $donnees = array('ville' => 'Oslo', 'zone' => 'Europe');
        if (utiliserCurl($url, $donnees, $resultat, $erreur, 'POST')) {
            echo $resultat;
        } else {
            echo "<div class='erreur'>$erreur</div>";
        }
    ?>
</body>
</html>
```

Listing 5. fonctions_sockets.php

```
<?php

/**
 * Prepare la requete HTTP a envoyer au serveur distant
 * @param $host (string) nom du serveur distant hebergeant la ressource
 * @param $chemin_fichier (string) chemin de la ressource sur le serveur
distant
 * @param $donnees (array) tableau de donnees a passer en parametres au
serveur distant
 * @param $methode (string) methode d'envoi de la requete HTTP
 * @return (string) requete
 */
function creeRequeteHTTP($host, $chemin_fichier, $donnees, $methode){
    $header = '';
    $corps = '';
    $arguments = http_build_query($donnees);
    // methode POST
    if ($methode == 'POST') {
        // preparer l'entete HTTP avec les arguments
        $header = "POST /$chemin_fichier HTTP/1.1\n";
        $header .= "Host:$host\n";
        $header .= "Content-type: application/x-www-form-urlencoded\n";
        $header .= "Content-length:".strlen($arguments)."\n";
        // preparer le corps
        $corps = $arguments."\n";
    }
    // methode GET
} else {
    // preparer l'entete HTTP avec les arguments
    $header = "GET /".$chemin_fichier.'?' . $arguments . " HTTP/1.1\n";
    $header .= "Host:$host\n";
    // pas de corps a envoyer avec la methode GET
}
// fermer la connexion
$header .= "Connection:close\n\n";
return $header.$corps;
}
```

utilisant ces fonctions peuvent être lus. Pour des raisons de sécurité, la directive `allow_url_fopen` a le plus souvent la valeur `off`. Dans ce cas, l'appel à un fichier sur un serveur distant provoquera le message d'alerte suivant :

warning: wrapper is disabled in the server configuration by allow_url_fopen=0

Les Listings 1 et 2 présentent un exemple de téléchargement de fichier avec la fonction PHP `file_get_contents`. Le script du Listing 2 demande une ressource sur le serveur distant `monEphemeride.com` et affiche le contenu du fichier téléchargé dans le navigateur, ou un message d'erreur si l'ouverture du fichier distant a échoué (Figure 5).

La méthode présentée dans cette section est implémentée dans le Listing 1. La fonction `utiliserURLopen` récupère la ressource dont l'URL est passée en premier argument. Elle stocke les données renvoyées par le serveur distant dans la variable `$resultat`. Cette variable étant passée par adresse à la fonction (`symbole & avant le nom de la variable`), son contenu sera accessible en dehors de la fonction. En cas d'erreur de téléchargement du fichier, la fonction `file_get_contents` retourne le booléen `false` ainsi qu'un message d'alerte PHP (`warning`). Afin de bloquer l'affichage du message d'alerte, l'opérateur de contrôle d'erreur `@` est utilisé. Un message d'erreur est stocké dans la variable `$erreur`, passée elle aussi par référence. La fonction `utiliserURLopen` retourne un booléen : la valeur `true` si le message d'erreur est vide, la valeur `false` sinon.

Le deuxième argument passé à la fonction `utiliserURLopen` est un tableau associatif contenant les données à envoyer à la ressource, dans la partie arguments (*query string*) de l'URL. La fonction `http_build_query`, disponible depuis PHP 5, génère une chaîne de requête sous la forme de paires nom=valeur, à partir du tableau de données. Il suffit de préfixer cette chaîne par un point d'interrogation et de l'ajouter à l'URL pour demander la ressource en lui passant des arguments. En cas d'erreur cette fonction déclenche un message d'alerte PHP, l'opérateur de contrôle d'erreur `@` est donc employé pour bloquer le message.

Le Listing 2 inclut le Listing 1. Il initialise les deux variables `$resultat` et `$erreur`, passées par référence à la fonction `utiliserURLopen`, ainsi que l'URL pour télécharger la ressource `info_soleil.php`. Par défaut le fichier `info_soleil.php` retourne les heures du lever et coucher de soleil à Paris, dans la zone Europe. Dans l'exemple présent, l'éphéméride de la ville de Rome est demandé. Le

tableau de données \$données est donc initialisé avec les arguments à envoyer au serveur distant, afin d'obtenir les informations pour la ville de Rome.

En cas de réussite, le contenu généré par le fichier *info_soleil.php* est affiché dans le navigateur. Il s'agit d'un titre correspondant à la ville et à la zone géographique demandées, ainsi que deux lignes de texte affichant les heures de lever et de coucher du soleil. Le résultat est visible en Figure 5. En cas d'échec, le message d'erreur contenu dans la variable \$erreur est affiché à l'écran.

Utilisation de la bibliothèque CURL

L'extension PHP/CURL permet à un script PHP de se comporter comme un client d'un autre serveur. Il devient possible de construire une requête HTTP, de la soumettre au serveur avec la méthode GET ou POST et de traiter directement la réponse du serveur dans le script. L'extension repose sur la bibliothèque open-source et gratuite *libcurl*, disponible pour de nombreux systèmes d'exploitation sur le site officiel de CURL (voir cadre *Sur Internet*).

Pour communiquer avec un serveur web avec CURL il faut :

- initialiser une session CURL avec la fonction PHP curl_init,
- définir les options de transfert avec les fonctions curl_setopt ou curl_setopt_array,
- envoyer la requête HTTP avec la fonction curl_exec,
- récupérer le corps de la réponse HTTP (option de transfert CURLOPT_RETURNTRANSFER),
- fermer la session CURL avec la fonction curl_close.

Cette section de l'article est illustrée par les Listings 3 et 4, qui montrent deux exemples de téléchargement avec CURL de la ressource *info_soleil.php* en utilisant les méthodes GET et POST. Le Listing 3 est une bibliothèque de fonctions et est inclus dans le Listing 4.

Initialiser une session CURL

La fonction curl_init initialise une session CURL et retourne une référence vers une ressource externe CURL :

```
$curl_ref = curl_init();
```

Cette ressource sera utilisée par les autres fonctions de la bibliothèque CURL. La fonction curl_init retourne le booléen false en cas d'erreur. Si la bibliothèque n'est pas chargée, les fonctions préfixées par curl ne sont pas disponibles et tout appel à l'une de ces

Listing 5. *fonctions_sockets.php* – suite

```
/*
 * Extrait l'entete et le corps de la reponse HTTP
 * @param $resultat (string) contenu du fichier (var affectee par
reference dans la fonction)
 * @param $erreur (string) message d'erreur eventuel (var affectee par
reference dans la fonction)
 */
function analyserReponseHTTP(&$resultat, &$erreur){
    // separer l'entete HTTP du corps du resultat
    $tab_resultat = preg_split('#\n\n|\r\n\r\n#', $resultat);
    // $tab_resultat doit comporter une case pour l'entete et au moins une
case pour le corps, sinon erreur
    if (is_array($tab_resultat) && count($tab_resultat) >= 2) {
        // recuperer les donnees dans 2 variables
        $rep_header = array_shift($tab_resultat);
        $resultat = implode("\n", $tab_resultat);
        // recuperer le code de la reponse dans l'entete de la reponse
HTTP
        if (preg_match('#^HTTP/1.\d\s+(\d{3})#', $rep_header, $matches))
{
            if ($matches[1] != '200') {
                $erreur = "Fichier non trouve";
            }
            } else {
                $erreur = "Erreur entete HTTP";
            }
        } else {
            $erreur = "Erreur reponse HTTP";
        }
    }
/**
 * Telecharge une ressource a partir d'un serveur distant en utilisant les
sockets
 * @param $host (string) nom du serveur distant hebergeant la ressource
 * @param $chemin_fichier (string) chemin de la ressource sur le serveur
distant
 * @param $donnees (array) tableau de donnees a passer en parametres au
serveur distant
 * @param $resultat (string) contenu du fichier (var affectee par
reference dans la fonction)
 * @param $erreur (string) message d'erreur eventuel (var affectee par
reference dans la fonction)
 * @param $methode (string) methode d'envoi de la requete HTTP (methode
GET par defaut)
 * @return boolean
 */
function utiliserSockets($host, $chemin_fichier, $donnees, &$resultat,
&$erreur, $methode = 'GET') {
    // ouvrir la connexion
    if ($socket = @fsockopen($host, '80')) {
        // preparer la requete HTTP pour demander la ressource au serveur
distant
        $requete = creeRequeteHTTP($host, $chemin_fichier, $donnees,
$methode);
        // envoyer les donnees (requete) au serveur
        $nb = fwrite($socket, $requete);
        // verifier que les donnees ont ete envoyees correctement
        if ($nb == strlen($requete)) {
            // recuperer la reponse du serveur
            $resultat = stream_get_contents($socket);
            // si erreur ne pas continuer
            if ($resultat == false) {
                $erreur = 'Echec de lecture des donnees';
            } else {
                analyserReponseHTTP($resultat, $erreur);
            }
        } else {
            $erreur = "Echec de l'envoi des donnees au serveur";
        }
        // liberer les ressources
        fclose($socket);
    } else {
        $erreur = 'Erreur connexion socket';
    }
    return empty($erreur);
}
?>
```

Listing 6. methode_3.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Heures de coucher/lever du soleil</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Troisième méthode : utilisation des <em>sockets</em></h1>
        <?php
        // inclusion de fonctions
        require_once 'fonctions_sockets.php';
        // initialisation de variables
        $resultat = '';
        $erreur = '';
        // recuperer les heures du lever et du coucher de soleil pour Los Angeles (methode GET)
        $donnees = array('ville' => 'Los_Angeles', 'zone' => 'America');
        if (utiliserSockets('monEphemeride.com', 'info_soleil.php', $donnees, $resultat, $erreur)) {
            echo $resultat;
        } else {
            echo "<div class='erreur'>$erreur</div>";
        }
        // reinitialiser les variables
        $resultat = '';
        $erreur = '';
        // et pour Tokyo (methode POST)
        $donnees = array('ville' => 'Tokyo', 'zone' => 'Asia');
        if (utiliserSockets('monEphemeride.com', 'info_soleil.php', $donnees, $resultat, $erreur, 'POST')) {
            echo $resultat;
        } else {
            echo "<div class='erreur'>$erreur</div>";
        }
    ?>
    </body>
</html>
```

Listing 7. utils.php

```
<?php

require_once 'fonctions_url_fopen.php';
require_once 'fonctions_curl.php';
require_once 'fonctions_sockets.php';
/**
 * Telecharge une ressource a partir d'un serveur distant
 * @param $host (string) nom du serveur distant hébergeant la ressource
 * @param $chemin_fichier (string) chemin de la ressource sur le serveur distant
 * @param $donnees (array) tableau de données à passer en paramètres au serveur distant
 * @param $resultat (string) contenu du fichier (var affectée par référence dans la fonction)
 * @param $erreur (string) message d'erreur éventuel (var affectée par référence dans la fonction)
 * @param $methode (string) méthode d'envoi de la requête HTTP
 * @return boolean
 */
function recupererRessource($host, $chemin_fichier, $donnees, &$resultat, &$erreur, $methode='GET') {
    $url = "http://$host/$chemin_fichier";
    // essayer de récupérer les données par la première méthode
    if (ini_get('allow_url_fopen')){
        utiliserURLFopen($url, $donnees, $resultat, $erreur);
    } else if (function_exists('curl_init')){
        // si la première méthode n'est pas disponible, tenter la seconde
        utiliserCurl($url, $donnees, $resultat, $erreur, $methode);
    } else {
        // si les deux premières ont échoué, essayer la troisième méthode
        utiliserSockets($host, $chemin_fichier, $donnees, $resultat, $erreur, $methode);
    }
    return empty($erreur);
}
?>
```

fonctions génère une erreur fatale PHP avec le message suivant :

Call to undefined function curl_init()

La fonction `utiliserCurl` du Listing 3 initialise une session CURL ou affecte une variable d'erreur en cas d'échec.

Définir les options de transfert

Les options de transfert sont définies avec les fonctions `curl_setopt` ou `curl_setopt_array` qui prennent en premier argument la ressource CURL. La fonction `curl_setopt` définit une seule option de transfert, elle accepte en second argument l'option à fixer (constante prédefinie) et en troisième argument la valeur à lui affecter. La fonction `curl_setopt_array` prend en second argument un tableau d'options.

Par défaut la réponse du serveur est redirigée vers la sortie standard, c'est à dire vers le navigateur. Pour obtenir le résultat de la requête HTTP dans une chaîne de caractères, il faut donner la valeur `true` à l'option `CURLOPT_RETURNTRANSFER`, comme dans la ligne de code suivante :

```
curl_setopt($curl_ref, CURLOPT_RETURNTRANSFER, true);
```

Dans l'exemple des Listings 3 et 4, le but est de télécharger les données renvoyées par le serveur `monEphemeride.com` et non pas de les retourner directement vers le navigateur. Cette option est donc utilisée dans la fonction `setoptions` du Listing 3, qui définit les options d'envoi et l'URL.

Les méthodes GET et POST peuvent être utilisées pour envoyer des données au script `info_soleil.php` situé sur le serveur `monEphemeride.com`. La méthode de transfert CURL par défaut est la méthode GET. Les données étant placées dans l'URL, il suffit alors de concaténer la variable `$arguments` au chemin de la ressource, en les séparant par un caractère point d'interrogation. L'URL est fixée avec l'option `CURLOPT_URL`, comme dans l'exemple ci-dessous :

```
curl_setopt($curl_ref, CURLOPT_URL, $url.'?'.$arguments);
```

Dans le Listing 4, la première opération demande les horaires du lever et coucher du soleil pour la ville de Londres, en utilisant la méthode GET.

Il est possible de préciser le numéro de port de l'application sur le serveur distant hébergeant la ressource. Il suffit de rajouter le caractère deux points suivi du numéro de port après le nom du serveur, ou bien de paramétrier l'option `CURLOPT_PORT` en lui affectant

Listing 8. exemple.php

```

<?php

// inclusion de fonctions

require_once 'utils.php';

/***
 * Telecharge un fichier sur un serveur distant et extrait l'heure du coucher
 * du soleil a partir du contenu du fichier telecharge
 * @param $host (string) nom du serveur distant hébergeant la ressource
 * @param $chemin_fichier (string) chemin de la ressource sur le serveur distant
 * @param $donnees (array) tableau de donnees a passer en parametres au serveur distant
 * @param $methode (string) methode d'envoi de la requete HTTP (methode GET par defaut)
 */

function recupererCoucherSoleil($host, $chemin_fichier, $donnees, $methode = "GET") {
    $resultat = '';
    $erreur = '';
    // telecharger le fichier et recuperer son contenu
    if (recupererRessource($host, $chemin_fichier, $donnees, $resultat, $erreur, $methode)) {
        // expression reguliere pour recuperer uniquement les horaires de lever/coucher de soleil
        $exp = '/(\d{2}:\d{2}:\d{2})/';
        // decouper la reponse pour n'extraire que le coucher du soleil
        if (preg_match_all($exp, $resultat, $correspondances)) {
            // ne conserver que la deuxieme correspondance dans la chaine (le coucher du soleil)
            $coucher_soleil = $correspondances[0][1];
            echo "<div class='resultat'>Aujour d'hui a {$donnees['ville']}, le soleil se couchera a : $coucher_soleil</div>";
        } else {
            echo "<div class='erreur'>Impossible d'extraire le resultat pour la ville de {$donnees['ville']}</div>";
        }
    } else {
        echo "<div class='erreur'>$erreur</div>";
    }
}
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
        <title>Heures de coucher/lever du soleil</title>
        <link rel="stylesheet" type="text/css" href="style2.css" >
    </head>
    <body>
        <h1>Coucher du soleil</h1>
        <?php
        // recuperer les heures du coucher de soleil pour Shanghai (methode GET) et pour Honolulu (methode POST)
        $donnees = array('ville' => 'Shanghai', 'zone' => 'Asia');
        recupererCoucherSoleil('monEphemeride.com', 'info_soleil.php', $donnees);
        $donnees = array('ville' => 'Honolulu', 'zone' => 'Pacific');
        recupererCoucherSoleil('monEphemeride.com', 'info_soleil.php', $donnees, 'POST');
        ?>
    </body>
</html>

```

comme valeur le numéro de port voulu. Par exemple pour communiquer avec le serveur Web *monEphemeride.com* sur le port 80, afin de récupérer les informations pour la ville de Londres, l'option `CURLOPT_URL` est paramétrée ci-dessous :

```
curl_setopt($curl_ref, CURLOPT_URL, 'http://monEphemeride.com:80/info_soleil.php?ville=London&zone=Europe');
```

Par défaut les serveurs web écoutent sur le port 80 donc il est rarement nécessaire de préciser cette information.

Dans le cas d'une requête effectuée avec la méthode `POST`, les données doivent être placées dans le corps de la requête. Ceci est effectué en donnant la valeur `true` à l'option `CURLOPT_POST`. Les données elles-mêmes sont placées dans un tableau associatif ou dans une chaîne de caractères sous la forme de paires `nom=valeur`. Ces données sont affectées à l'option `CURLOPT_POSTFIELDS`.

Dans le Listing 4, la seconde opération effectuée dans le script demande l'éphéméride pour la ville d'Oslo, en utilisant la méthode `POST`. La fonction `setOptions` va être de nouveau appelée pour définir les options de transfert CURL. Lorsque la méthode `POST`

est utilisée, les options sont définies grâce à la fonction `curl_setopt_array`, comme dans l'exemple suivant :

```
$options = array(CURLOPT_URL => $url,
    CURLOPT_POST => true,
    CURLOPT_POSTFIELDS => $donnees);
curl_setopt_array($curl_ref, $options);
```

Envoyer la requête HTTP et récupérer la réponse du serveur

La fonction `curl_exec` envoie la requête HTTP au serveur distant. Elle prend en paramètre une référence à la session CURL.

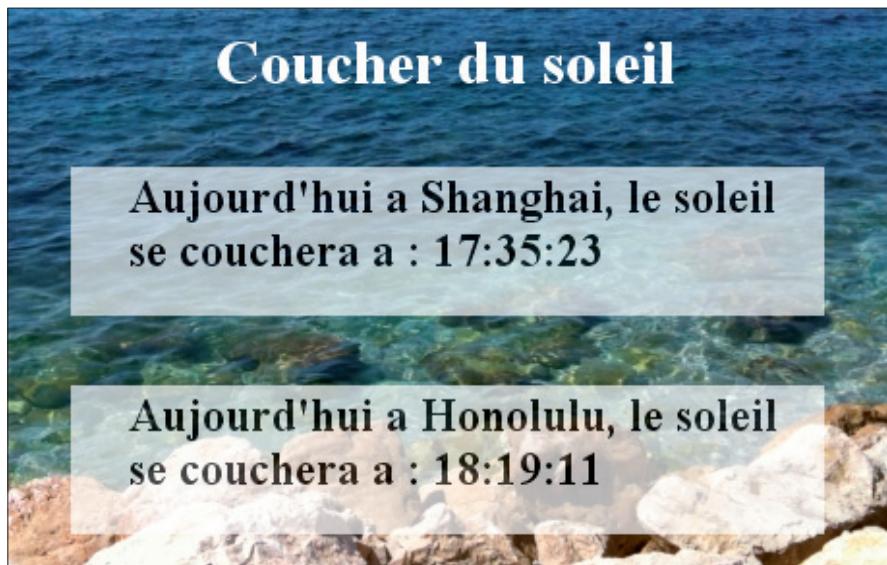


Figure 6. Téléchargement en fonction du réglage du serveur

Lorsque l'option `CURLOPT_RETURNTRANSFER` a la valeur `true`, la fonction retourne le corps de la réponse HTTP dans une chaîne de caractères. En cas d'échec la fonction retourne le booléen `false`.

La réponse obtenue par défaut ne contient que le corps de la réponse HTTP. Des informations sur l'en-tête peuvent être obtenues avec la fonction `curl_getinfo` qui prend en argument la ressource CURL et une option. Dans le Listing 3, cette fonction est utilisée dans la fonction `utiliserCurl` pour vérifier que le code de la réponse HTTP est bien 200 (succès).

Fermer la session CURL

La fonction `curl_close` prend en argument la ressource retournée par la fonction `curl_init`. Elle libère les ressources et ne retourne aucune valeur.

Pour en savoir plus sur CURL, vous pouvez vous référer à deux articles parmi les anciens numéros de ce magazine, le premier étant une introduction à CURL (Novembre 2008), le second décrivant des fonctionnalités plus avancées de cette bibliothèque (Janvier 2009).

Utilisation de socket

La troisième méthode de téléchargement de fichier expliquée dans cet article consiste à établir une communication par `socket`, entre le serveur sur lequel le script PHP est hébergé et un serveur web externe hébergeant la ressource voulue (fichier à télécharger). La communication par `socket` est effectuée directement à l'aide de fonctions natives PHP, comme dans l'exemple des Listings 5 et 6.

La combinaison de l'adresse IP d'un serveur et du numéro de port du service demandé sur le serveur est un `socket`. Lorsqu'un navigateur

web ou un serveur établit une connexion avec un serveur web sur le port 80, il établit un `socket`. Des données sont envoyées par le client à l'application (requête), et celle-ci retourne des données au client (réponse). Une fois une connexion par `socket` établie, il est possible d'écrire (envoyer) ou de lire (télécharger) des données.

Pour communiquer avec un serveur web par `socket` il faut :

- établir une connexion avec le serveur grâce à la fonction PHP `fsockopen`,
- envoyer des données (requête HTTP) au serveur avec la fonction `fwrite`,
- récupérer la réponse HTTP avec la fonction `stream_get_contents`,
- fermer le socket avec la fonction `fclose`.

Etablir une connexion

La fonction `fsockopen` prend en premier paramètre le nom du serveur (ou son adresse IP) avec lequel établir une connexion. Elle peut recevoir d'autres arguments optionnels, comme le numéro de port du service demandé sur le serveur (80 pour le service Web), deux variables (passées par adresse) qui recevront les numéros et messages d'erreur éventuels, ou encore un dernier argument correspondant à un délai d'attente maximal pour établir la connexion. La fonction retourne un identifiant de ressource qui sera utilisé par les fonctions de lecture, d'écriture et de fermeture de `socket`. L'exemple suivant ouvre une connexion sur le port 80 du serveur `monEphemeride.com` :

```
$socket = fsockopen('monEphemeride.com', '80');
```

En cas d'erreur, la fonction retourne le booléen `false` et un message d'alerte PHP est gé-

néré. Ce message est bloqué dans le code du Listing 5 grâce à l'utilisation de l'opérateur de contrôle d'erreur `@`.

Envoyer des données

Le serveur Web avec lequel la connexion est établie attend une requête HTTP. Il faut écrire cette requête dans une chaîne de caractères et l'envoyer au serveur en utilisant la fonction `fwrite`. Cette dernière reçoit en paramètres l'identifiant de la ressource de la connexion et la chaîne de caractères à envoyer au serveur. Vous avez vu précédemment dans cet article comment écrire une requête HTTP vers un serveur. La fonction `creerRequeteHTTP` du Listing 5 effectue cette opération, à partir des données qu'elle reçoit en paramètre.

Dans le Listing 6 les deux méthodes d'envoi `GET` et `POST` sont utilisées pour récupérer les heures de lever et de coucher du soleil à Los Angeles (méthode `GET`) et à Tokyo (méthode `POST`).

Dans le cas de l'envoi par la méthode `GET` les données à envoyer sont placées directement dans l'URL, dans la partie *query string*. Par exemple pour demander les informations au serveur pour la ville de Los Angeles, la requête à envoyer est :

```
GET /info_soleil.php?ville=Los_Angeles&zone=America HTTP/1.1\n
Host:monEphemeride.com\n
Connection:close\n
\n
```

Le caractère `\n` permet de passer à la ligne. Comme la connexion est effectuée en utilisant la version 1.1 du protocole HTTP, il est nécessaire d'ajouter une seconde ligne pour le champ `Host`, correspondant au nom du serveur Web hébergeant la ressource.

Dans le cas d'une connexion effectuée par la méthode `POST`, les données sont envoyées dans le corps de la requête HTTP. Il faut donc préciser dans l'en-tête le type MIME des données et la taille du corps de la requête. Par exemple :

```
POST /info_soleil.php HTTP/1.1\n
Host:monEphemeride.com\n
Content-Type:application/x-www-form-urlencoded\n
Content-Length:21\n
Connection:close\n
\n
ville=Tokyo&zone=Asia
```

Pour les deux méthodes il est nécessaire de préciser le champ `connection` en lui affectant la valeur `close` pour signaler au serveur Web que la communication est terminée. Si

ce champ n'est pas précisé, le serveur mettra un délai avant de renvoyer une réponse. Il faut ensuite ajouter deux sauts de lignes (les deux caractères \n suivant la ligne Connection:close\n). Ces deux sauts de ligne permettent d'obtenir une ligne blanche entre l'en-tête et le corps de la requête. Cette ligne est obligatoire, même pour l'envoi d'une requête avec GET qui contient un corps vide.

Dans la fonction `utiliserSockets` du Listing 5, la requête HTTP est envoyée au serveur distant en utilisant la fonction PHP `fwrite`. Cette dernière prend en premier argument la ressource et en second argument la chaîne contenant la requête HTTP. Elle retourne le nombre d'octets transmis au serveur, ou `false` en cas d'erreur. Afin de s'assurer que la requête a été intégralement transmise au serveur distant, un test sur le nombre de caractères envoyés est réalisé dans la fonction `utiliserSockets`.

Récupérer la réponse

La fonction `stream_get_contents` retourne la réponse HTTP envoyée par le serveur distant dans une chaîne de caractères. Elle prend en argument la ressource de la connexion renvoyée par `fsockopen`. En cas d'échec la fonction retourne le booléen `false`. La réponse HTTP renvoyée par le serveur comporte deux parties : l'en-tête et le corps. Ce dernier correspond au contenu du fichier téléchargé.

Dans l'exemple du Listing 5, la réponse du serveur est passée en paramètre à la fonction `analyserReponseHTTP`, qui se charge d'extraire le corps de la réponse.

La fonction `preg_split` sépare la réponse en fonction du séparateur saut de ligne. Dans le cas d'un serveur Windows, celui-ci est représenté par la suite de symboles \r\n\r\n. Dans le cas d'un serveur Linux et Mac OS, le saut de ligne est représenté par les symboles \n\n. La fonction `preg_split` retourne un tableau contenant les parties séparées de la réponse. La première case contient l'en-tête, la ou les autres cases contiennent le corps.

La fonction `array_shift` permet de supprimer la première case dans le tableau et retourne son contenu dans une chaîne. La fonction `implode` regroupe le contenu du tableau restant, soit le corps de la requête, dans une chaîne de caractères.

Afin de s'assurer que la transaction s'est déroulée correctement, le code à trois chiffres de la réponse HTTP est extrait de l'en-tête de la réponse, stockée dans la variable `$rep_header`. Seul le code 200 est accepté, tout autre code affecte un message d'erreur à la variable `$erreur`.

Sur Internet

- <http://php.net/manual/fr/ref.filesystem.php> – Manuel des fonctions de manipulation de fichiers sur le site officiel de PHP,
- <http://curl.haxx.se/> – Site officiel de la bibliothèque CURL,
- <http://fr.php.net/manual/fr/book.curl.php> – Manuel des fonctions de l'extension CURL sur le site officiel de PHP,
- <http://fr2.php.net/manual/fr/ref.network.php> – Manuel des fonctions réseaux sur le site officiel de PHP.

Fermer la connexion

La fonction `fclose` ferme le *socket* dont le pointeur est passé en paramètre. Elle retourne `false` en cas d'erreur.

Télécharger un fichier en fonction des réglages du serveur

Vous venez d'apprendre à utiliser trois méthodes différentes de téléchargement de fichier depuis un script PHP. Toutes ont leurs avantages et inconvénients. La première méthode qui permet d'ouvrir un fichier à distance est la plus simple à utiliser, mais elle ne permet pas d'envoyer des données par la méthode `POST` et est dépendante de la valeur d'une directive dans le `php.ini`.

La seconde méthode est un peu plus complexe à utiliser, elle permet d'envoyer des requêtes HTTP par les méthodes `GET` et `POST` et de récupérer séparément l'en-tête et le corps de la réponse HTTP. Elle est cependant dépendante de la présence de la bibliothèque CURL sur le serveur hébergeant le script qui effectue le téléchargement. Elle dépend également de l'activation de l'extension PHP/CURL.

Enfin la troisième méthode est la plus complexe à utiliser. Il faut écrire intégralement la requête HTTP GET ou POST à envoyer au serveur distant et extraire les informations de la réponse de ce dernier. Elle présente malgré tout l'avantage de ne nécessiter aucune installation ni réglage du serveur. Les *sockets* sont activés par défaut dans PHP, les fonctions présentées dans cet article font partie du cœur de PHP.

En combinant ces trois méthodes il est possible d'implémenter un téléchargement de fichier distant (image, page web,...) fonctionnant dans la majorité des configurations. Le Listing 7 choisit une méthode de téléchargement en fonction des réglages du serveur. Dans la fonction `recupRessource`, si la directive `allow_url_fopen` est activée, la première méthode est utilisée. Sinon dans le cas où la fonction `curl_init` est disponible, le téléchargement est effectué en utilisant CURL. Enfin si les deux premières méthodes ne sont pas disponibles, la fonction établit une connexion par *socket*. Si des erreurs interviennent dans l'une de ces trois méthodes (fichier

inexistant, URL incorrecte, ..), alors la variable `$erreur` reçoit le message d'erreur correspondant et la fonction `recupRessource` retourne le booléen `false`.

Le Listing 8 fait appel au Listing 7 afin de récupérer des informations qu'il formate avant de les afficher à l'écran. Il envoie les paramètres nécessaires à la fonction `recupRessource` du Listing 7 pour obtenir les éphémérides des villes de Shanghai et de Honolulu et ce, quels que soient les réglages du serveur.

Les données du corps de la réponse peuvent être traitées par des fonctions sur les chaînes de caractères ou des expressions régulières afin d'extraire les informations souhaitées. Dans l'exemple présenté ici, une expression régulière est utilisée dans la fonction `recupCoucherSoleil` afin de ne récupérer que l'information sur l'heure du coucher de soleil dans la réponse HTTP (Figure 6).

Conclusion

Vous avez appris à télécharger un fichier situé sur un serveur distant, depuis un script PHP, par l'intermédiaire de trois méthodes différentes. Une fois le fichier récupéré, vous pouvez le stocker sur le disque du serveur hébergeant le script client. Vous pouvez également le manipuler directement pour en extraire des informations, en utilisant les connaissances acquises dans les précédents articles de ce magazine, décrivant la manipulation de fichiers.

CÉCILE ODERO MAGALI CONTENSIN

Cécile Odero est spécialisée dans la conception et le développement d'applications web en PHP. Elle travaille au CNRS comme ingénieur en développement et déploiement d'applications.

Contact : cecile.oder@gmail.com

Magali Contensin, auteure du livre *Bases de données et Internet avec PHP et MySQL*, est chef de projet en développement d'applications au CNRS. Elle enseigne depuis plus de dix ans le développement d'applications web à l'Université. Contact : <http://magali.contensin.online.fr>

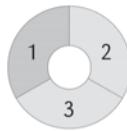
PHP et Delphi

Cet article présente les différentes possibilités pour faire interagir votre application locale en Delphi avec PHP et ainsi permettre la communication entre logiciels et serveurs simplement.

Cet article explique :

Les différentes utilisations de PHP dans Delphi, j'aborderais donc ce domaine assez vaste pour vous montrer les différentes possibilités entre ces deux langages, le but étant de pouvoir faire interagir une application locale avec un serveur web.

Niveau de difficulté



Delphi est un langage de programmation orientée objet (POO) créé et édité par Borland en 1995 pour les premières versions de Windows, en 2001, une version linux du nom de *Kylix* fit son apparition. Borland créa ensuite *CodeGear* possédant ses propres moyens pour créer des outils destinés aux développeurs, après deux ans de recherches, *CodeGear* fut racheté par la société *Embarcadero* pour 23 millions de dollars.

Delphi possède un EDI (*Environnement de Développement Intégré*) ou encore IDE, il existe plusieurs versions de cet IDE. Dans cet exemple, nous utiliserons *Embarcadero Delphi 2010* fonctionnant sous Windows, sachez tout de même que plusieurs versions linux existent comme *Kylix* ou encore *Lazarus*.

Prise en main de l'IDE Embarcadero Delphi 2010

La prise en main de l'IDE est assez simple, l'IDE possède deux grandes parties,

Tout d'abord, nous pouvons apercevoir la vue arborescente des objets (*Structuration des composants de l'application*) qui permettent de

Ce qu'il faut savoir :

- Avoir une base dans le développement en PHP.
- Avoir quelques bases en Delphi (Environnement de l'IDE Borland ou Embarcadero).

voir tous les objets placés sur l'application visible en direct.

Le code source de l'application en question peut être modifié avant la compilation, deux choix s'offrent à vous, soit vous codez manuellement les événements des boutons ou autres objets de l'application en passant du *Design au Code* grâce aux onglets en bas, soit vous pouvez utiliser l'onglet *Events* visible dans l'inspecteur d'objets (voir à gauche de la Figure 1).

Je vous conseille de suivre quelques tutoriaux sur Delphi avant de poursuivre si vous n'avez jamais utilisé l'IDE, vous pouvez pour cela, consulter des sites comme *phidels.com*.

Le composant TwebBrowser

Le composant *WebBrowser* est un composant de Delphi permettant de visualiser des pages web depuis une application, ce composant se trouve dans l'onglet *Internet*. Grâce à ce composant, nous pouvons facilement faire un navigateur fonctionnant avec le moteur d'Internet Explorer (voir Figure 2).

L'application est composée comme ceci :

- Deux composants *Tpanel* sans captions (un en *Align Center* et un autre en *Align Top*).
- Un composant *TwebBrowser* en *Align Center* par dessus le *Tpanel* en *Align Center* aussi.

- Un *Tedit* sur le *Tpanel* en *Align Top*.
- Trois boutons sur le *Tpanel* en *Align Top*.

Voici les procédures de l'application (*Events*) :

- // Dès que l'application se lance, le *WebBrowser1* va sur Google.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebBrowser1.Navigate('http://www.google.fr/');
end;
```

- // Dès qu'on clique sur l'*Edit1*, le texte s'efface pour pouvoir marquer une url.

```
procedure TForm1.Edit1Click(Sender: TObject);
begin
  Edit1.Clear;
end;
```

- // Dès qu'on clique le *bouton1* (*Ok*), le *WebBrowser1* va sur l'url indiquée dans l'*Edit1*.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  WebBrowser1.Navigate(Edit1.Text);
end;
```

- // Dès qu'on clique sur le *bouton2* (*Back*), le *WebBrowser1* va à la page précédente.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  WebBrowser1.GoBack;
end;
```

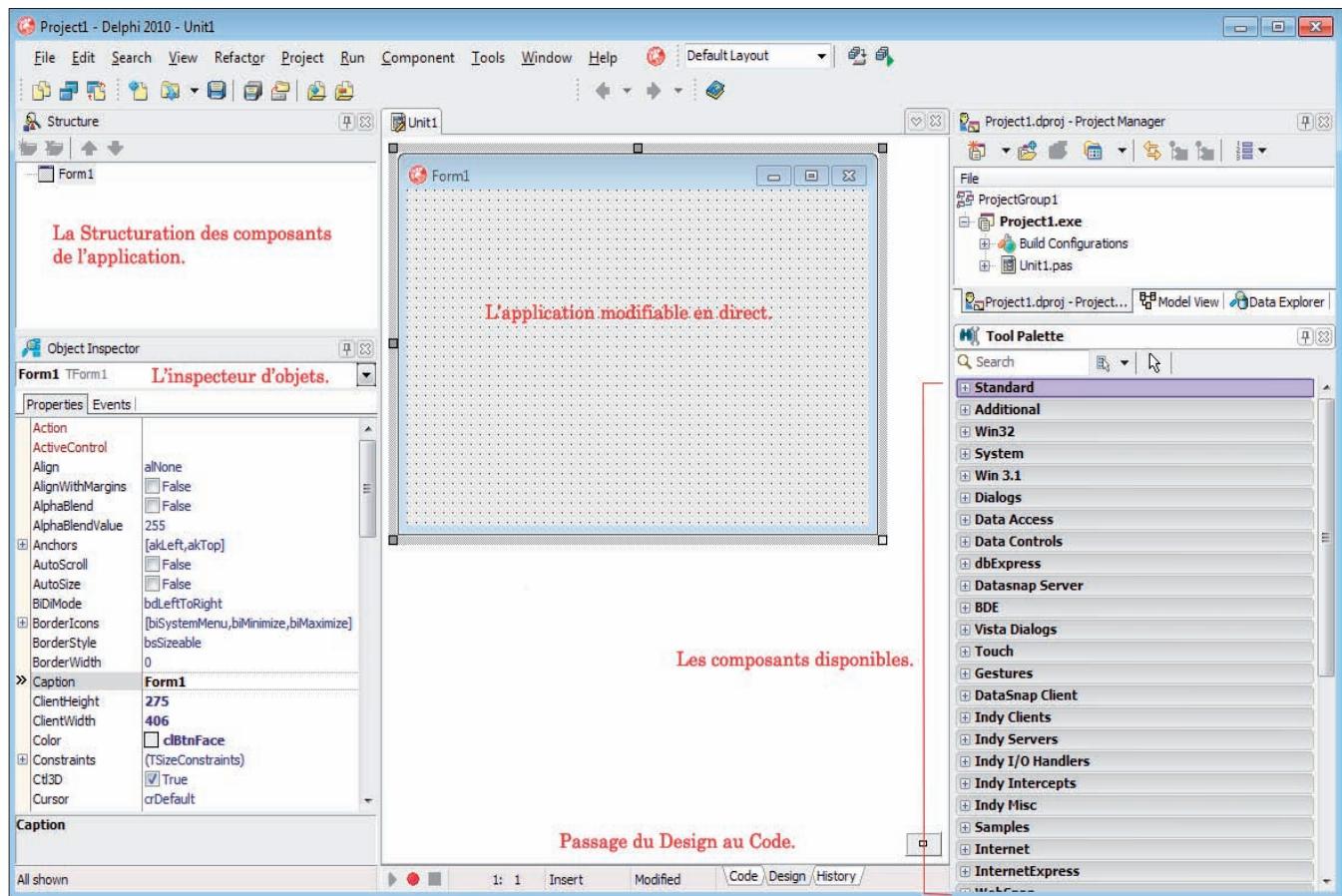


Figure 1. IDE Embarcadero Delphi 2010

- // Dès qu'on clique sur le bouton3 (Forward), le WebBrowser1 va à la page suivante.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  WebBrowser1.GoForward;
end;
```

Avant de poursuivre, assurez vous de savoir faire un navigateur basique comme le mien. Nous pouvons maintenant aussi récupérer le code source d'une page grâce au *WebBrowser*.

- // Au lancement de l'application, le *WebBrowser1* va sur Google.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WebBrowser1.Navigate('http://www.google.fr/');
end;
```

- // Dès que le *WebBrowser* est allé sur une page et qu'il a fini son chargement...

```
procedure TForm1.WebBrowser1DocumentComplete(ASender: TObject);
```

```
const pDisp: IDispatch; var URL: OleVariant;
begin
```

```
// On efface le texte du Memo1 et
// on affiche la source de la page
// à l'intérieur.
Memo1.Clear;
Memo1.Text := WebBrowser1.OleObject.
  Document.body.innerHTML;
end;
```

Le résultat est montré dans la Figure 3.

Il est aussi possible d'utiliser ce composant sans être connecté à Internet soit avec un serveur *xampp* local afin d'utiliser PHP, soit en mettant la propriété *Offline* du composant à *True* pour charger des pages HTML se trouvant sur un disque dur.

La librairie Indy

La librairie *Indy* (*Internet Direct*) est une librairie Open Source pour les développeurs Delphi qui permet d'établir des connexions sockets entre clients et serveurs. Dans cet article, j'aborderais seulement quelques fonctionnalités d'*Indy* en rapport avec PHP et non toutes les possibilités d'*Indy*.

Avec *Indy*, il est possible par exemple de transmettre un \$_GET grâce au composant

TidHTTP de la librairie. Voici un exemple basique de code avec Google :

```
uses
  IdHTTP; // À rajouter dans votre code si
  n'est pas présent.
type
  IdHTTP1: TIdHTTP; // À rajouter dans votre
  code aussi si n'est pas présent.
```

- // Dès que vous cliquerez sur le bouton1, une requête GET sera envoyée sur la recherche google en fonction de ce que vous avez tapé dans l'*Edit1*.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```
// http://www.google.fr/#hl=fr&source=
hp&q=%s&btnG=Recherche+Google
&meta=&aq=f&oq=' + le mot tapé (Exemple :
Sofia Rezgui Villaume clonée) +'&fp=
f8ad601b167bde66
IdHttp1.Get('http://www.google.fr/
#hl=fr&source=hp&q=%s&btnG=
Recherche+Google&meta=&aq=f&oq=
'+ IdHTTP1.URL.PathEncode(Edit1.Text)
+'&fp=f8ad601b167bde66');
end;
```



Figure 2. Navigateur simple en Delphi

La partie *Design* de l'application doit donc ressembler à ceci (voir Figure 4). Il est aussi possible avec *Indy* de remplir des formulaire \$ _ POST avec la méthode `Post()` du *TIdHTTP*.

- // Dès que l'on clique sur le bouton 1...

```
procedure TForm1.Button1Click(Sender: TObject);
var
  parametres : TStringList;
begin
  // On créer donc la TStringList.
  parametres := TStringList.Create;
  try
    // On ajoute les paramètres que l'on souhaite envoyer sur le formulaire
    $ _ POST.
```

```
parametres.Add('Param1=Valeur1');
parametres.Add('Param2=Valeur2');
```

- // On envoie les données sur le formulaire.

```
IdHttp1.Request.ContentType := 'application/x-www-form-urlencoded';
IdHTTP1.Post('http://site.com/page.php',parametres);
```

- // Finalement, on libère la TStringList.

```
finally
  parametres.free;
end;
end;
```

Vous pouvez par la même occasion faire un système de mise à jour automatique grâce à *Indy* et PHP comme ceci :

- // À l'ouverture de l'application ...

```
procedure TForm1.FormCreate(Sender: TObject);
begin
```

- // On dit que la version de l'application est la première.

```
version := '1';
```

- // On envoie une requête \$ _ GET à la page index.php avec le numéro de la version de notre application.

Sur Internet :

- <https://downloads.embarcadero.com/free/delphi> – Téléchargement des versions d'évaluation,
- <http://delphi.developpez.com/> – Site d'apprentissage complet sur le langage.

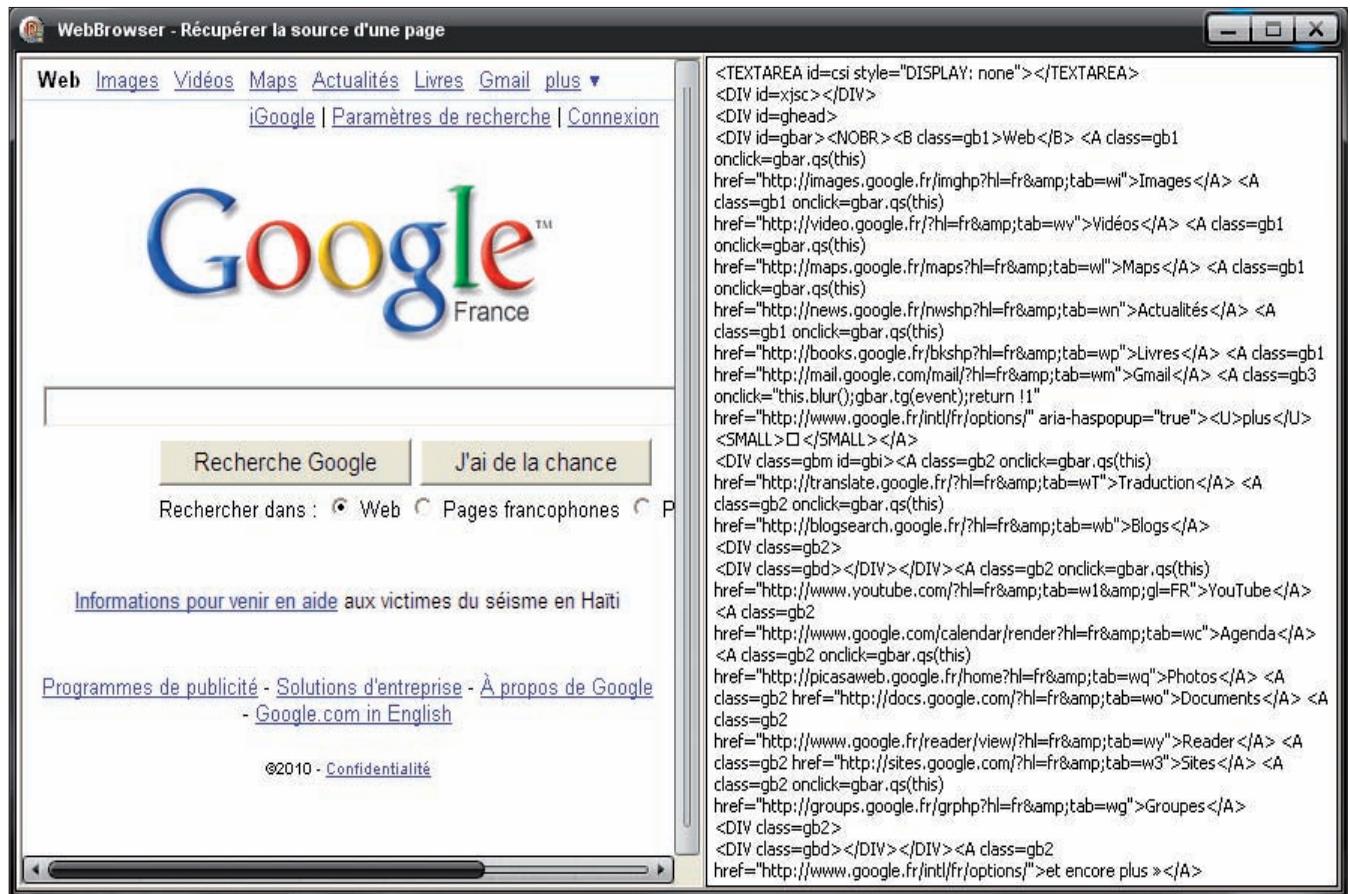


Figure 3. Source HTML avec Delphi

```

reponse := IdHTTP1.Get('http://
localhost/test/index.php?version='+
version);

• // On affiche le résultat de ce que nous
dit la page index.php du serveur.

Label2.Caption := reponse;
end;

```

Code de la page *index.php*:

```

<?php
/* Si la requête $_GET 'version'
reçue est inférieur à 2,
on affiche le lien de téléchargement
de la version 2 de l'application. */
if ($_GET['version'] < 2) {
echo "http://localhost/test/update2.exe";
}

```

```

/* Sinon on dit à l'application qu'il
n'y a pas de nouvelle version. */
else {
echo "Vous possédez déjà la dernière
version.";
}
?>

```

Voici le résultat lorsque la variable *version* est inférieure à 2 (voir Figure 5).

Conclusion

Dans cet article, vous avez normalement pu constater que Delphi et PHP se marient très bien ensemble mais Delphi ne s'arrête pas là, de nombreuses autres interactions avec PHP sont possibles ainsi que différentes utilisations avec d'autres langages comme l'Assembleur, Sql, Perl ou intégrer de l'Ajax à nos pages.

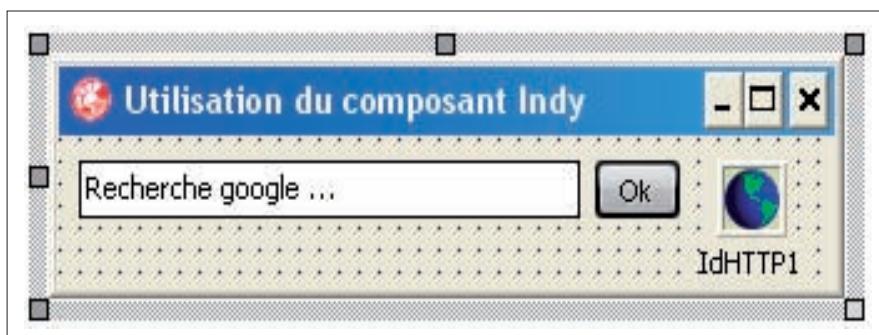


Figure 4. \$_GET avec Delphi

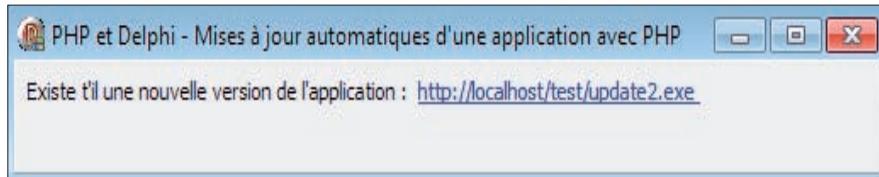


Figure 5. Communication logiciel/serveur

FAURE YANN

Faure Yann est un développeur passionné de sécurité informatique depuis plusieurs années, programmant dans plusieurs langages, il aimerait poursuivre ses études dans la programmation ou dans la sécurité informatique.

Dans le prochain numéro de **phpsolutions**

Le périodique *phpsolutions* est publié par
Software Press Sp. z o.o. SK
Bokserka 1, 02-682 Varsovie, Pologne
Tél. 0975180358, Fax. +48 22 244 24 59
www.phpmag.org

Président de Software Press Sp. z o.o. SK : Paweł Marciniak

Directrice de la publication : Ewa Łozowicka

Imprimerie, photogravure :
ArtDruk www.artdruk.com
Imprimé en Pologne/Printed in Poland

Dépôt légal : à parution
ISSN : 1731-4593

Distribution : MLP
Parc d'activités de Chesnes, 55 bd de la Noirée
BP 59 F - 38291 SAINT-QUENTIN-FALLAVIER CEDEX
(c) 2010 Software Press, tous les droits réservés

Rédacteur en chef : Łukasz Bartoszewicz
Préparation du CD : Andrzej Kuca
Maquette : Agnieszka Marchocka
Couverture : Agnieszka Marchocka

DTP : Sławomir Sobczyk Studio2W@gmail.com
Composition : Sławomir Sobczyk
Correction : Aymeric Lagier

Bêta-testeurs : Fabrice Gyre, Brice Favre, Valérie Viel, Aymeric Lagier, Christophe Milhau, Alain Ribault, Stéphane Guedon, Eric Boulet, Mickaël Puyfages, Christian Hernoux, Isabelle Lopi, Antoine Beluze, Timothée Neullas, Yann Faure, Adrien Mogenet, Jean-François Montgaillard, Turmeau Nicolas, Jonathan Marois, Wilfried Ceron, Wajih Letaief, François Van de Weerd, Jeremy Rafflin, Eric Vincent.

Les personnes intéressées par la coopération sont priées de nous contacter :
editor@phpmag.org

Fabrication : Andrzej Kuca andrzej.kuca@software.com.pl

Diffusion : Ilona Lepieszka ilona.lepieszka@software.com.pl

Publicité : publicite@software.com.pl

La rédaction fait tout son possible pour s'assurer que les logiciels sont à jour, pourtant elle décline toute responsabilité pour leur utilisation. Elle ne fournit pas de support technique lié à l'installation ou l'utilisation des logiciels enregistrés sur le CD-ROM. Tous les logos et marques déposés sont la propriété de leurs propriétaires respectifs.

Pour créer les diagrammes on a utilisé le programme



SmartDraw

Le CD-ROM joint au magazine a été testé avec AntiVirenKit de la société G Data Software Sp. z o.o

AVERTISSEMENT

Les techniques présentées dans les articles ne peuvent être utilisées qu'au sein des réseaux internes. La rédaction du magazine n'est pas responsable de l'utilisation incorrecte des techniques présentées. L'utilisation des techniques présentées peut provoquer la perte des données !

FICHE TECHNIQUE

■ Manipuler les cookies avec PHP

Les cookies sont utilisés par un grand nombre d'applications et de sites web pour stocker les préférences d'un utilisateur (langue, style, ...), pour assurer un suivi de session (panier de commande, intranet, ...) ou pour pister l'utilisateur. Dans cet article vous apprendrez le principe et la manipulation des cookies depuis un script PHP

PRATIQUE

■ Créez votre blog avec Drupal

Drupal est un CMS de plus en plus utilisé à un niveau professionnel. Complet, il bénéficie d'un support communautaire important et de modules aboutis, y compris dans sa version de base. Ce CMS permet de déployer rapidement des sites web dynamiques. Il peut aussi bien être utilisé tel quel, en tant que moteur de contenu, que personnalisé par le biais de modules et thèmes. Vous allez voir dans cet article, comment l'utiliser au mieux.

DOSSIER

■ Programmation orientée objet

Nous allons parcourir ensemble les bases de la programmation orientée objet et les illustrer par des exemples. Nous aborderons la création d'une classe et des ses propriétés et méthodes, les méthodes magiques, l'héritage, la durée de vie des objets ainsi que la gestion des exceptions.

PROJETS

■ Sélection des meilleures extensions sur Joomla!

Le CMS Joomla! est connu pour la diversité de ses extensions. Parmi eux, on distingue les composants et les modules qui représentent la partie fonctionnelle d'un site. Depuis, il existe plus de 5000 extensions et le choix peut être parfois difficile. C'est pourquoi, je vous propose une sélection des meilleures extensions disponibles pour le CMS Joomla en passant par les composants et modules les plus fiables et intéressants pour la création de votre site.

Et de nombreux autres articles à ne pas manquer !

La rédaction se réserve le droit de modifier le contenu de la revue.

Déjà l'été !
Restez zen : on s'occupe de tout.



Topymedia donne vie
à vos projets Web

Web : www.topymedia.com

Tel : 0 899 230 860

1,34€/appel + 0,34€/Minute

POUR PARTICULIERS
ET PROFESSIONNELS

Génialement Simple !



Cet homme
essaie de mettre son
site Internet à jour

Cet homme
a mis son site
Internet à jour avec
WebGazelle CMS 2.0

WebGazelle CMS 2.0 rencontre l'AJAX

www.webgazelle.net

Webgazelle.net, une marque de

