JQUERY | CAKEPHP | MAGENTO | DRUPAL | DOM | KANNEL | ZEND

Le plus grand magazine sur PHP au monde

phpsolutions

Nouvelles technologies et solutions pour les développeurs PHP

PHP N° 7/2010 (43) ISSN 1731-4593

MAGENTO ET DRUPAL
CRÉEZ VOTRE PORTAIL E-COMMERCE

JQUERY
REQUÊTES AJAX ET PLUGIN JQUERY UI

DOCUMENT OBJECT MODEL
MANIPULEZ DU XML À L'AIDE DE L'API DOM DE PHP5

KANNEL
CRÉEZ UN SVA AVEC LA PASSERELLE SMS/WAP

SERVEUR : AMI OU ENNEMI ?
RÉALISEZ VOS PROJETS WEB !

FICHE TECHNIQUE

CAKEPHP
DÉVELOPPEZ RAPIDEMENT VOS APPLICATIONS

OUTILS

PRESTASHOP
SOLUTION E-COMMERCE OPEN SOURCE

HOSTING NEXT LEVEL

IPv6 RUNNING

intel CORE i7 inside

HETZNER DEDICATED ROOT SERVER EQ 4

- Intel®Core™ i7-920 Quad-core avec technologie Hyper-Threading
- 8 Go DDR3 RAM
- 2 x 750 Go SATA-II HDD (Software-RAID 1)
- Système d'exploitation Linux
- Windows Server à partir de 13 € par mois
- Trafic réseau illimité[1]
- Système « Rescue »
- 100 Go d'espace de sauvegarde
- Domain Registration Robot
- Pas de contrat minimum
- Frais d'installation : 126 €

42,- € par mois

HETZNER DEDICATED ROOT SERVER EQ 8

- Intel®Core™ i7-920 Quad-core avec technologie Hyper-Threading
- 24 Go DDR3 RAM
- 2 x 1500 Go SATA-II HDD (Software-RAID 1)
- Système d'exploitation Linux
- Windows Server à partir de 13 € par mois
- Trafic réseau illimité[1]
- Système « Rescue »
- 100 Go d'espace de sauvegarde
- Domain Registration Robot
- Pas de contrat minimum
- Frais d'installation : 126 €

75,- € par mois

HETZNER DEDICATED ROOT SERVER EQ 9

- Intel®Core™ i7-975 Quad-core avec technologie Hyper-Threading
- 12 Go DDR3 RAM
- 3 x 1500 Go SATA-II HDD (Software-RAID 5)
- Système d'exploitation Linux
- Windows Server à partir de 13 € par mois
- Trafic réseau illimité[1]
- Système « Rescue »
- 100 Go d'espace de sauvegarde
- Domain Registration Robot
- Pas de contrat minimum
- Frais d'installation : 126 €

84,- € par mois

1 Le trafic est gratuit. Nous limitons la vitesse à 10 Mbit/s si 5000 Go/mois sont dépassés. En option, une bande passante permanente garantie de 100 Mbit/s est proposée à 6 € par To supplémentaire.

2 En tant que nouveau client, vous pouvez bénéficier de 8 € de réduction sur n'importe quel produit présenté ici. Utilisez, s'il vous plait, le code promo **011908** en remplissant votre bon de commande. Cette offre expire le 01.09.2010.

HETZNER ONLINE

« Hosting next level » signifie simplement que Hetzner Online propose aujourd'hui la plus puissante des solutions d'hébergement dédié actuellement disponibles ! Les plans serveurs dédiés Hetzner Online sont conçus pour une rapidité maximale et une stabilité réseau extrême dans nos propres datacenters basés en Allemagne. Avec nos tarifs compétitifs et un support hors du commun, nous dépassons déjà les exigences de nos clients partout dans le monde.

www.hetzner.info

info@hetzner.com

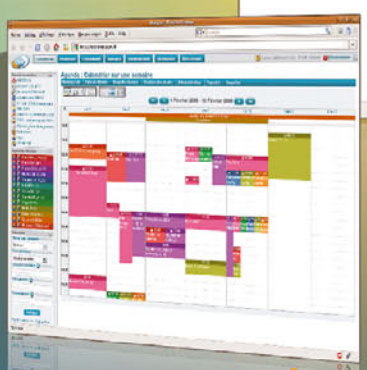
LINAGORA

présente

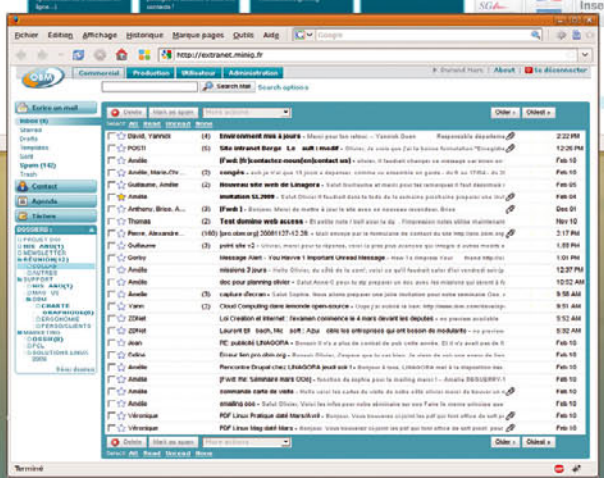
OBM Online

LA MESSAGERIE COLLABORATIVE OPEN SOURCE HÉBERGÉE !

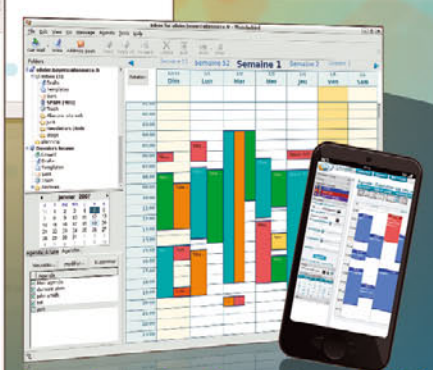
NEW



Agendas partagés



Webmail Web 2.0



Synchronisation et mobilité

http://online.obm.org

Le périodique *phpsolutions* est publié par
Software Press Sp. z o.o. SK
Bokserska 1, 02-682 Varsovie, Pologne
Tél. 0975180358, Fax. +48 22 244 24 59
www.phpsolmag.org

Président de Software Press Sp. z o.o. SK :
Paweł Marciniak

Directrice de la publication :
Ewa Łozowicka

Dépôt légal :
à parution
ISSN : 1731-4593

Rédacteur en chef :
Łukasz Bartoszewicz

Couverture :
Sławomir Sobczyk

DTP :
Sławomir Sobczyk *Studio2W@gmail.com*

Composition :
Sławomir Sobczyk

Correction :
Valérie Viel, Thierry Borel

Bêta-testeurs :
Fabrice Gyre, Brice Favre, Valérie Viel, Cyril David, Christophe Milhau, Alain Ribault, Stéphane Guedon, Eric Boulet, Mickael Puyfages, Christian Hernoux, Isabelle Lupi, Antoine Beluze, Timotée Neullas, Yann Faure, Adrien Mogenet, Jean-François Montgaillard, Turmeau Nicolas, Jonathan Marois, Wilfried Ceron, Wajih Letaief, François Van de Weerdt, Jeremy Rafflin, Eric Vincent.

Les personnes intéressées par la coopération sont priées de nous contacter :
editor@phpsolmag.org

Publicité :
publicite@software.com.pl

Pour créer les diagrammes on a utilisé le programme
smartdraw.com
SmartDraw

TABLE DES MATIÈRES

VARIA

OUTILS

PROJETS

DOSSIER

PRATIQUE

FICHE TECHNIQUE

POUR LES DÉBUTANTS

PHP 5.2.14 et PHP 5.3.3

Deux nouvelles versions de PHP viennent de sortir. La version PHP 5.3.3 apporte de nombreux correctifs mais aussi une amélioration de la stabilité.
La version de PHP 5.2.14 apporte principalement des correctifs de sécurité. Par ailleurs, il est important de signaler que le support de la branche 5.2 de PHP est arrêté et cessera à partir de cette version. Par conséquent si de nouvelles versions doivent apparaître, elles seront liées autour des problèmes de sécurités. C'est pourquoi il est important de migrer vers une version de PHP 5.3
http://www.php.net

xDebug 2.1

La nouvelle version de *xDebug* vient de sortir. Elle est compatible PHP 5.3 et fonctionne à partir de PHP 5.2. Elle ne supporte donc plus les versions de PHP inférieures. Dans les nouveautés : Les erreurs de connexion, interpréter les entêtes HTTP, tracer l'assignation des variables, désactiver les opérateurs @, tracer la pile, débogage à distance.
http://xdebug.org

DotClear 2.2

Dans les nouveautés de la version 2.2 de *Dotclear* pour réaliser un blog, vous trouverez un nouvel assistant d'installation, une évolution des tags et du moteur. Cette nouvelle version est compatible PHP 5.3 et aussi avec la création de *plugins* et de thèmes.
Toutes les évolutions sont présentées dans le communiqué du site.
http://fr.dotclear.org/blog/post/2010/06/30/Dotclear-2.2

Sugar CRM 6

La nouvelle version 6 de Sugar CRM, réalisée toujours en PHP, se veut être plus rapide et plus simple. Bien sûr, cette version apporte de nombreuses nouveautés et améliorations comme l'ouverture vers d'autres applications, une recherche contextuelle, une nouvelle barre de raccourcis, une aide à la configuration.
http://www.sugarcrm.com

PHP commit hooks

PHP commit *Hooks* est une application réalisée en PHP, vous permettant d'avoir un développement avec des numéros de versions comme le versionning. Vos dossiers et fichiers PHP seront alors représentés avec des crochets pour vous permettre d'avoir une conception simple et structurée. Cette application propose par ailleurs les mêmes fonctionnalités qu'un SVN comme laisser un message d'informations...
http://github.com/kore/php-commit-hooks

PHP pour Android

Les téléphones mobiles sont depuis de nombreuses années, les nouveaux supports de communications, mais depuis quelques mois, de nouvelles possibilités vous sont proposées, entre autre avec la sortie en juillet 2010 de PFA. Le projet PFA (*PHP For Android*) est un nouveau projet, qui va vous permettre d'installer un environnement PHP CLI dans un terminal Android. Ainsi, vous pouvez maintenant créer des pages Webs dynamiques en PHP, tout en exécutant ces pages sur vos téléphones Android.
Pour réaliser votre première page PHP sous Android, vous devez posséder :

- Un environnement ACE (Android Script Environnement) qui va vous permettre d'exécuter des scripts.
- Installer le patch PhpForAndroid.apk.

Ensuite, vous pouvez exécuter vos scripts PHP sans serveur web, et même obtenir la liste des fonctions avec `phpinfo()` :

```
<?php
require_once("Android.php");
phpinfo();
```

Mais aussi interagir avec le hardware de votre téléphone mobile comme demander à votre téléphone de vibrer :

```
<?php
require _ once("Android.php");
$vibre= new Android();
$vibre->vibrate();
```

Site officiel : *http://phpforandroid.net/*

AR-PHP

AR-PHP est un ensemble de librairies PHP qui offre de nombreux outils pour les utilisateurs qui souhaitent proposer un site web en version Arabe. Ce projet est réalisé en PHP, sous licence Open source et nécessite PEAR.
Avec cette configuration, vous pourrez répondre aux nombreuses attentes des internautes, lorsqu'un site web veut proposer un site avec une version Arabe.
Cette librairie propose de nombreuses fonctionnalités de bases comme :

- La Conversion des dates entre le calendrier grégorien et hégirien.
- Traduction Anglais vers Arabe et Arabe vers Anglais.
- Différents types de détections (texte en arabe, character set...).
- Utilisation de la location GPS pour définir les heures.
- Clavier virtuel pour écrire en langue Arabe.
- Normalisation du texte arabe.

Site officiel : *http://www.ar-php.org*

Rédaction des actualités :
Christophe Villeneuve

Rejoignez le Club .PRO

Pour plus de renseignement : *editor@phpsolmag.org*

Stonfield Inworld

Stonfield Inworld propose aux entreprises des solutions globale d'intègration 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

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 entierement aux normes W3C (www.w3.org) de sites web de qualité, au graphisme soigné et employant les dernieres 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, integre, déploie et maintient des systemes de Gestion de Contenu Web, de Gestion Documentaire, de Gestion de la Relation Client (CRM), d'ecommerce et de travail ollaboratif.
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 cśur 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

PrestaShop,

solution e-commerce Open Source leader du marché

Forte d'une communauté Open Source de plus de 85 000 membres (dont 50 000 développeurs !), la solution PrestaShop est aujourd'hui leader dans le secteur de l'e-commerce Open Source.

Cet article explique :

- Ce qu'est PrestaShop, comment l'installer, le configurer et le personnaliser pour créer une boutique en ligne.

Ce qu'il faut savoir :

- Connaissances en PHP/MySQL, XHTML / CSS pour les développeurs.

Trois ans seulement après le lancement de PrestaShop, plus de 40 000 boutiques l'utilisent dans plus de 50 pays à travers le monde, et en plus de 45 langues. Sur les neuf derniers mois, PrestaShop a vu sa communauté Open Source tripler, passant de 30 000 membres à plus de 85 000, et propose aujourd'hui plus de 200 fonctionnalités dédiées aux marchands. Focus sur cette solution prometteuse, conçue en PHP/MySQL.

Qu'est-ce que PrestaShop ?

PrestaShop est une solution e-commerce Open Source, téléchargeable gratuitement, permettant d'ouvrir une boutique en ligne en quelques clics. D'une part, elle propose un back-office très puissant permettant de gérer en temps réel la boutique (catalogue, historique des commandes, frais de port, clients, paniers remplis ...). D'autre part, elle offre aux clients une boutique conviviale aux couleurs du marchand, avec de nombreux modules (80 sont inclus en standard) et offre plus de 200 fonctionnalités.

Le Back-Office, puissant et intuitif

PrestaShop offre un panneau d'administration complet, à la prise en main aisée, aussi bien pour les marchands que les développeurs. Outre la gestion des profils utilisateurs et de leurs droits respectifs, ce dernier est le cœur de la boutique : conçu autour d'un système modulaire, la solution PrestaShop propose plus de 80 modules en standard, permettant de compléter ou personnaliser la boutique et ses outils : il est ainsi possible d'activer ou de désactiver certaines fonctions de la boutique, aussi bien sur le Front-Office (affichage des fabricants, d'un bloc newsletter…) que sur le Back-Office : outils de statistiques, recherche rapide…

Quelques fonctionnalités phares :

- Parrainage, points de fidélité, liste cadeaux,
- + de 20 blocs de fonctionnalités configurables pour la page d'accueil.
- Consultation en temps réel des paniers créés par les clients.
- Statistiques détaillées de l'activité du site, et de la qualité du catalogue.
- Alertes e-mail et SMS en cas de nouvelles commandes, ruptures de stocks…
- Nombre de transporteurs et de moyens de paiement illimités.
- Jusqu'à plus de 100 000 ventes mensuelles (scalabilité).
- Modules de paiement intégrés (cartes bancaires et portefeuilles électroniques) : *PayPal, Moneybookers, Hipay, Google Checkout*, chèque, virement, comptant à la livraison.
- Optimisations pour le référencement naturel, gestion des métas etc.

Créer une boutique avec PrestaShop

PrestaShop s'installe sur un serveur web, via FTP. Configuration minimale requise : Linux, Unix, ou Windows ; Apache 1.3 ou supérieur, IIS 6 ou supérieur ; PHP 5.0 ou supérieur ; MySQL 5 ou supérieur. Le Processus d'installation détecte automatiquement la configuration du système, et permet une mise en place en cinq minutes.

A noter, de nombreux hébergeurs (*PrestaBox (http://www.prestabox.com)*, OVH) permettent l'installation de PrestaShop en 1 seul clic pour permettre à tous de se lancer dans l'e-commerce. Une fois installée, la boutique est prête à être configurée (transporteurs, taxes, informations légales…) et son catalogue, rempli. Sa prise en main est rapide et intuitive.

PrestaShop côté technique

L'équipe de développement de PrestaShop a dès le début choisi d'éviter l'utilisation d'un framework PHP existant pour se consacrer à la construction de son propre framework. Avec ce choix technique, PrestaShop n'est pas dépendant d'une solution tierce et les performances sont immédiatement au rendez-vous.

Ainsi, PrestaShop inclut dans son framework tous les éléments essentiels au confort du développeur web PHP :

- Une gestion objet des cookies, très simple à utiliser,
- Sa propre classe d'abstraction de base de données qui va même jusqu'à gérer la réplication SQL,
- Une gestion des moteurs de rendus de graphiques (utilisant différentes librairies comme *Artichow, Visifire, GoogleCharts ou encore xmlswfcharts)*,
- Une gestion de l'envoi d'e-mails personnalisés avec *SwiftMailer*,
- La création de documents PDF avec FPDF,
- Et de nombreux autres outils concernant la sécurité, l'internationalisation, etc.

PrestaShop respecte naturellement autant que possible une architecture MVC, la couche modèle étant gérée avec un design pattern *Active Record* et la vue grâce au moteur de templates *Smarty.* Le travail d'équipe est ainsi largement facilité, et le développement de nouvelles fonctionnalités est généralement très rapide.

La toute-puissance de PrestaShop ne vient cependant pas uniquement de son framework, mais surtout de son système de modules permettant à tout un chacun de développer ses propres fonctionnalités pour sa boutique. Il suffit donc de quelques lignes pour *greffer* une portion de code à un point stratégique de la solution. Lors du passage d'une commande par exemple, PrestaShop va automatiquement appeler les modules qui souhaitent effectuer du traitement. Sur le même modèle, les modules peuvent ajouter des éléments graphiques qui seront positionnés par PrestaShop au bon endroit sur le Front-Office. Le cœur de PrestaShop reste ainsi toujours propre, ce qui facilite grandement les mises à jour.

Front-Office et personnalisation du thème

Le thème standard de PrestaShop est codé en XHTML 1.1, le plus récent des standards HTML, HTML 5 n'étant pas encore la norme. Plus d'une dizaine de plugins de la librairie JQuery sont utilisés, aidant à obtenir un site fluide et dynamique.Le référencement naturel est optimisé, par le respect des balises sémantiques, et sa compatibilité avec les navigateurs Internet s'étend d'IE6 à Chrome.

Il offre une apparence claire et intuitive aux visiteurs. Quelques touches d'Ajax sont distillées sur l'ensemble de la boutique afin de la rendre plus attrayante en gardant toujours à l'esprit que l'ergonomie est primordiale. Par exemple, lorsqu'un produit est ajouté au panier, le déplacement du produit vers le panier est illustré. Il est par ailleurs possible d'installer des blocs de contenu (inscription newsletter, publicité…) à des endroits stratégiques de la boutique, tels que les colonnes de gauche et de droite, le *header* et le *footer.* Ces blocs sont présentés sous forme de modules au sein du Back-Office.

Le thème standard de PrestaShop est entièrement valide W3C, c'est une bonne base lorsqu'il s'agit de développer son propre thème personnalisé. Car personnaliser une boutique PrestaShop est facile : l'utilisateur peut se procurer un thème prêt à installer au sein de PrestaStore, la place de marché officielle de thèmes et modules pour PrestaShop, dont le catalogue dépasse les 700 références. Il peut également créer son thème sur mesure, grâce au système de templates *Smarty*, le HTML, le CSS et Javascript.

Tout le monde peut participer

PrestaShop est un projet Open Source qui a une politique de développement contrôlé : seule l'équipe de PrestaShop peut intégrer de nouvelles fonctionnalités. Cette restriction est nécessaire pour garder un minimum de cohérence technique pour le projet. En revanche, il existe de nombreux moyens de contribuer au développement du projet. N'importe qui peut soumettre un bug dans l'outil de rapport de bug du site communautaire *http://www.prestashop.com/bug_tracker/*, et si vous fournissez le patch avec le problème, c'est encore mieux !

Pour les passionnés, PrestaShop dispose également d'une place de marché, PrestaStore, qui permet aux développeurs, graphistes et intégrateurs de proposer à toute la communauté leurs propres modules et thèmes. Avec un marché de plus de 40 000 marchands avides de nouvelles fonctionnalités, PrestaStore permet à ses supporters la meilleure diffusion possible pour leurs contributions. Enfin, un SVN est bien évidemment à la disposition de tous pour bénéficier de la toute dernière version de la solution : *http://www.prestashop.com/en/downloads/#svn.*

Le DOM avec PHP5

Vous vous êtes toujours demandé s'il était possible de faire des manipulations du DOM avec PHP ? Ou alors vous savez que cela est possible mais vous ne savez pas comment. Dans ce cas, cet article est fait pour vous.

Cet article explique :

- Comment manipuler du XML à l'aide de l'API DOM de PHP5.

Ce qu'il faut savoir :

- Les bases de la programmation avec le langage PHP.
- La POO et son implémentation dans PHP 5.
- Les bases du XML et du XPATH.

Le terme DOM est l'abréviation pour Document Object Model ; il s'agit d'une spécification du W3C(*World Wide Web Consorsium*) définissant la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs de celui-ci. C'est une interface, API (*Application Programming Interface*) indépendante de toute plateforme et de tout langage, permettant à une application de parcourir la structure d'un document et d'agir de manière dynamique sur celui-ci. Les langages comme le javascript par exemple l'utilisent pour naviguer au sein d'un document HTML afin par exemple de récupérer les données d'un formulaire. Nous nous proposons dans cet article de présenter comment l'utiliser pour manipuler du XML en PHP5.

Les classes de l'extension Dom de PHP5

Contrairement à PHP4 qui était fortement porté sur un fonctionnement procédural et non procédurier, du fait des prémices des mécanismes objet qu'il commençait tout juste à implémenter, PHP5 intègre, lui, les notions de POO très avancées et l'extension DOM n'est pas en marge de cette logique. Cette dernière possède donc un ensemble de classes parmi lesquelles nous avons :

- `DOMAttr` : elle représente un attribut dans l'objet *DOMElement*.
- `DOMCharacterData` : elle représente un noeud contenant des données. Aucun noeud ne correspond à cette classe, mais d'autres noeuds en héritent.
- `DOMComment` : elle représente un noeud de commentaire, délimité par <!-- et -->.
- `DomDocument` : elle représente un document HTML ou XML entier.
- `DomDocumentType` : elle représente les types de document.
- `DomElement` : pour représenter un élément XML.
- `DomException` : elle dérive de la classe `Exception`, et permet de gérer les exceptions provoquées par des opérations DOM.
- `DOMNode` : pour la manipulation des nœuds XML.

Listing 1. *compagnie.xml*

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- exemple de fichier XML  -->
<root>
<option value="AA">American Airlines (États-Unis)</
option>
<option value="AB">Air Berlin (Allemagne)</option>
<option value="AC">Air Canada (Canada)</option>
<option value="AD">Air Paradise (Indonésie)</option>
<option value="AE">Mandarin Airlines (Taďwan)</option>
<option value="AF">Air France (France)</option>
<option value="AG">Air Contractors (UK) Ltd. (Irlande)</
option>
<option value="AH">Air Algérie (Algérie)</option>
<option value="AI">Air India (Inde)</option>
<option value="BS">British International Helicopters
(Royaume-Uni)</option>
<option value="BT">Air Baltic (Lettonie)</option>
<option value="BU">Braathens groupe SAS (Norvège)</
option>
<option value="CE">Nationwide Airlines (Afrique du Sud)</
option>
</root>
```

- `DOMNodeList` : pour manipuler une liste de nœuds.
- `DOMText` : elle représente le contenu textuel de `DomElement` ou `DomAttr`.
- `DomXpath` : cette classe permet de faire des requêtes Xpath sur un document XML avec le DOM.

Mise en œuvre

Pour illustrer la manipulation du DOM avec PHP5, nous utiliserons le document XML du Listing 1. Il est constitué d'un ensemble d'options correspondant chacun au nom d'une compagnie aérienne et possédant un attribut *value* ayant pour valeur le code de cette compagnie.

Nous allons donc procéder à un ensemble de manipulation de ce fichier tel que le montre le listing 2 :

- Création d'une représentation DOM d'un document XML la plus simple possible sans préciser de paramètres.
- On complète la déclaration en ajoutant la version du document: 1.0 et l'encodage utf-8.
- Chargement du document XML dans un premier temps depuis une chaîne de caractère, et dans un deuxième temps depuis un fichier (le fichier *compagnie.xml*).
- Parfois, on a besoin de pouvoir récupérer l'ensemble de l'arbre XML à des fins d'affichage ; cela se fait grâce à la méthode `saveXML()`.
- Il est possible via le DOM de valider un document XML relativement à une DTD (*Document Type Definition*) via la méthode `validate()`.
- Si par contre à la place d'une DTD nous avons plutôt un schéma XML alors nous utiliserons la méthode `schemaValidate()` qui prend en paramètre le chemin vers le fichier contenant le schéma.

Manipulation d'un document XML via le DOM

Pour manipuler un document XML via le DOM nous pouvons nous servir des classes que nous avons vues précédemment, telles qu'illustrées dans le Listing 3 :

- affichage du nom de l'élément racine du document,
- retrouver un élément grâce au nom de sa balise : ici on recherche tous les élément ayant pour nom de balise *option*. Avec le code de ce listing, nous obtenons le résultat du Listing 4. Il est également possible de lire la valeur d'un attribut. C'est ce qui est fait dans le Listing 5 où nous obtenons la valeur de l'attribut *value* de nos éléments *option* . Le Listing 6 présente le résultat obtenu.

L'ajout d'un nœud peu également se faire en suivant la procédure présentée dans le Listing 7. Les étapes sont les suivantes :

Listing 2. *Manipulation1*

```
        $dom = new DOMDocument();
//ajout de la version et de l'encodage
        $dom = new DOMDocument('1.0', 'utf-8');
//chargement du document depuis une chaîne de caractères
        $dom->loadXML('<root><node/></root>');
//chargement du document depuis un fichier
        $dom->load('compagnie.xml');
//enregistrement du document sur le système de fichier
        $dom->save('NomFichier.xml');
//récupération de l'arbre xml du document pour
//l'afficher par exemple
        $machaine = $dom->saveXML();
         echo $machaine ;
//test pour la validation du document suivant une DTD
         if ($dom->validate())
         {
             echo "Ce document est valide";
         }
        else
        {
        echo("ce document n'est pas valide") ;

         }
//validation dans le cas d'un schema xml
          $dom->schemaValidate($path) ;
```

Listing 3. *Manipulation2*

```
//manipulation

$dom->load(realpath('compagnie.xml'));
$root = $dom->documentElement;
echo $root->nodeName."<br>";
$compagnie = $dom->getElementsByTagName('option');
foreach ($compagnie as $name)
{
        echo $name->firstChild->nodeValue."<br>";
}
```

Listing 4. *Résultat*

```
root
American Airlines
Air Berlin
Air Canada
Air Paradise
Mandarin Airlines
Air France
Air Contractors
Air AlgÃ©rie
Air India
British International Helicopters
Air Baltic
Braathens groupe SAS
Nationwide Airlines
```

Listing 5. *Lecture des attributs*

```
$compagnie = $dom->getElementsByTagName('option');

        foreach ($compagnie as $name)
        {
                if($name->hasAttribute('value'))
                {
                echo '--code: '.$name->getAttribute('
value').'<br>';
                }
        }
```

- on crée le nœud *option* que l'on souhaite ajouter au document,
- on crée le nœud textuel correspondant en restant conforme à la structure de notre document,
- on crée ensuite l'attribut de l'élément ou alors celui-ci est modifié s'il existe déjà,

Listing 6. *Affichage des attributs*

```
--code: AA
--code: AB
--code: AC
--code: AD
--code: AE
--code: AF
--code: AG
--code: AH
--code: AI
--code: BS
--code: BT
--code: BU
--code: CE
```

Listing 7. *Ajout d'un noeud au dokument*

```
        //création d'un noeud
$newElement = $dom->createElement("option");
//crée le noeud textuel conformément a la structure
de notre document
$textNode = $dom->createTextNode("Toumai Air Tchad");
//création de l'attribut de l'element ou
//alors modification si l'attribut existe déjà

$newAtt = $newElement->setAttribute("value","TT");
//attacher le noeud créer au reste du document
en respectant sa structure //grace a la methode
appendChild
$newElement->appendChild($textNode);
//$root etant l'element racine de notre document
$root->appendChild($newElement);
```

Listing 8. *Verification de l'ajout du noeud au document*

```
$compagnie = $dom->getElementsByTagName('option');

      foreach ($compagnie as $name)
      {
            echo $name->firstChild->nodeValue."<br>";
      }
```

Listing 9. *Résultat de l'ajout du noeud*

```
American Airlines
Air Berlin
Air Canada
Air Paradise
Mandarin Airlines
Air France
Air Contractors
Air AlgÃ©rie
Air India
British International Helicopters
Air Baltic
Braathens groupe SAS
Nationwide Airlines
Toumai Air Tchad
```

Listing 10. *Autres manipulations*

```
$clone = $newElement->cloneNode(false);
$deleteAtt = $newElement->removeAttribute("value");
$deleteNode = $root->removeChild($newElement);

//affichage du texte l'option ayant pour indice 6

$var = $dom->getElementsByTagName("option")->item(6);
 echo $var->firstChild->nodeValue."<br>";
```

Listing 11. *Utilisation du xpath*

```
$xpath =  new DomXPath($dom);
      $xpath _ query = "//option";
      $resultat = $xpath->query($xpath _ query);

      foreach ($resultat as $nom)
      {
            echo $nom->firstChild->nodeValue."<br>";
      }
```

Listing 12. *Résultat avec le xpath*

```
American Airlines
Air Berlin
Air Canada
Air Paradise
Mandarin Airlines
Air France
Air Contractors
Air Algérie
Air India
British International Helicopters
Air Baltic
Braathens groupe SAS
Nationwide Airlines
```

- et une fois ceci fait, le tout est lié à notre document XML de départ.

Pour être certain que le nœud a bien été ajouté au document, le code présenté au Listing 8 sera exécuté. Le résultat est présenté au niveau du Listing 9 ou on constate une nouvelle ligne qui n'existait pas dans notre document XML de départ : la ligne *Toumai Air Tchad.*

Autres manipulations

D'autres manipulations sont également possibles avec le DOM : on a par exemple la possibilité de :

- créer un nœud par clonage d'un nœud existant,
- supprimer un attribut d'un nœud,
- supprimer complètement un nœud,
- énumérer les noeuds d'une liste de nœud : la méthode `item()` de la classe `DomNodelist` permet de le faire aisément en passant en paramètre à cette méthode l'index du nœud recherché. Il est à noter que tout comme avec les tableaux en PHP, l'indexation ici commence à *0*. Ces manipulations sont illustrées dans le Listing 11. Notons que l'affichage du texte avec l'option ayant pour indice 6 nous donne comme résultat *Air Contractors.*

DOM dispose également d'une classe spécifique permettant d'effectuer des requêtes XPATH sur un document XML ; la logique est la suivante :

- on crée un objet *DomXPath* en lui passant en paramètre le document XML sur lequel sera effectuée la requête ;
- on définit la requête à exécuter ;
- on l'exécute et on récupère les résultats que l'on exploite.

Le Listing 12 en est une illustration où il est question de récupérer tous les nœuds *option* de notre document en partant de la racine de celui-ci. Le résultat obtenu est présenté au Listing 13.

Listing 13. *Code récapitulatif de notre exemple*

```
$dom = new DOMDocument('1.0', 'utf-8');
        $dom = new DOMDocument();
        //supprimer les espaces redondants dans le
document :
        $dom->preserveWhiteSpace = false;
        //charger le document xml a partir d'un fichier
existant
        $dom->load(realpath('compagnie.xml'));

        //--------------utilisation des methodes du
DOMDocument-------------//

        //affichage du nom de l'element racine
        $root = $dom->documentElement;
        echo $root->nodeName;

        //retrouver un element grace au nom de la
balise:
        $compagnie = $dom->getElementsByTagName('option');
        echo("<br>nom balise<br>");
        foreach ($compagnie as $name)
        {
                echo $name->firstChild->nodeValue."<br>";
        }
        //obtenir la valeur d'un attribut
        $compagnie = $dom->getElementsByTagName('option');
        foreach ($compagnie as $name)
        {
                if($name->hasAttribute('value'))
                {
                        echo '--code: '.$name->getAttrib
ute('value').'<br>';
                }
        }

        //crée un noeud
        $newElement = $dom->createElement("option");
        //crée le noeud textuel conformément a la
structure de notre document
        $textNode = $dom->createTextNode("Toumai Air
Tchad");
        //création de l'attribut de l'element ou alors
modification si l'attribut existe déjà
        $newAtt = $newElement->setAttribute("value","TT");

        //attacher le noeud créer au reste du document en
respectant sa structure
        //grace a la methode appendChild
        $newElement->appendChild($textNode);
        //$root etant notre l'element racine de notre
document
        $root->appendChild($newElement);

        $compagnie = $dom->getElementsByTagName('option');
        foreach ($compagnie as $name)
        {
                echo $name->firstChild->nodeValue."<br>";
        }
        //création d'un noeud par clonage d'un noeud
existant
        //cette methode prend en entrée un paramètre
booleen qui indique si oui ou non les noeuds fils
        //sont également clonés
        $clone = $newElement->cloneNode(false);

        //on peut également supprimer un attribut de la
manière suivante
        //cette methode prend en entrée le nom de
l'attribut a supprimer
        $deleteAtt = $newElement->removeAttribute("value");

        //suppression d'un noeud grace à la methode
removeChild() qui permet de Supprimer un fils
        //de la liste des enfants d'un noeud
        $deleteNode = $root->removeChild($newElement);

        //la methode item() de la classe DomNodelist
Retourne un noeud spécifié par son index
        $compagnie = $dom->getElementsByTagName('option');
        foreach ($compagnie as $name)
        {
                echo $name->firstChild->nodeValue."<br>";
        }

        echo'---------------------------------';
        $var = $dom->getElementsByTagName("option")->item(6);
        echo $var->firstChild->nodeValue."<br>";

        //manipulation avec le xpath
        $xpath =  new DomXPath($dom);
        //permettra de selectionner tout les noeuds
enfant ayant pour nom "option"
        //en se basant sur le contexte actuel.
        $xpath _ query = "./option";
        $resultat = $xpath->query($xpath _ query);
        foreach ($resultat as $nom)
        {
                echo $nom->firstChild->nodeValue."<br>";
        }
        $dom->load('compagnie.xml');
```

Conclusion

Tout au long de cet article, nous avons pu parcourir les principales fonctionnalités et possibilités que nous offres le dom avec PHP5. Nous sommes désormais capable de construire un document XML, de le stocker, de le parcourir, de l'interroger et même de le modifier. Nous nous sommes focalisés sur des fonctionnalités basiques afin de faire un exemple simple mais il est tout à fait possible de faire des choses plus complexes via cette interface. Cet article n'étant pas exhaustif, nous vous invitons à faire un tour sur Internet afin de compléter vos connaissances sur le sujet. N'hésitez pas également à nous faire part de vos remarques par mail, surtout si vous souhaitez voir un aspect de l'interface approfondit dans les prochains articles.

Sur Internet

- *http://eusebius.developpez.com/php5dom/* – Un article très intéressant sur le DOM pour bien démarrer,
- *http://www.php.net/docs-echm.php* – Le manuel de PHP pour plus de détails sur les classes du DOM,
- *http://www.w3schools.com/dom/dom_text.asp* – Un site didactique sur les technologies du web qui vous permettra d'en apprendre d'avantage sur le DOM en général,
- *http://www.commentcamarche.net/* : l'encyclopédie en ligne de l'informatique.

MBELLA EKOUME K. DEMAZY

Technicien Supérieur en Réseaux & Télécom, il a poursuivi son cursus en faisant une maîtrise en développement logiciel. Très orienté WEB, celui-ci se passionne pour les solutions Open Source et cela fait maintenant 3 ans qu'il travaille en tant que développeur professionnel. Pour le contacter, envoyer un mail à l'adresse dembello85@yahoo.fr.

BYOOS solutions Open Source

Qui sommes nous ?

BYOOS *http://byoos.fr* solutions Open Source est à l'origine une entreprise Française spécialisée dans le logiciel médical libre travaillant sur les technologies WEB ainsi que les outils libres de droits ex LINUX, APACHE, PHP, MYSQL. Ce choix est celui du promoteur de l'activité BOBARD Gabriel Julien Pierre qui, passionné de nouvelles technologies, a permis à BYOOS d'acquérir un savoir faire dans le domaine de l'informatique médicale en proposant la gestion du plateau technique via le WEB ainsi que le site vitrine institutionel pour les Hôpitaux, cliniques, pharmacies, laboratoires d'analyses. Ces organisations préférent faire appel à des spécialistes du domaine.

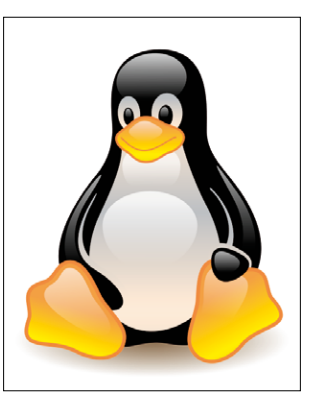
Nous sommes aussi implanté en Afrique Sub-Saharienne afin de promouvoir le logiciel médical Web Based MEDLINES+, MLDOCS, XMEDOO. Ces développements sont issus de briques open sources et disponibles sur la toile par simple téléchargement.

Afin d'atteindre ses objectifs de vulgarisation des logiciels libres, BYOOS solutions s'est dotée d'un centre de formation au cœur de Yaoundé au Cameroun puis à Douala et Kribi, grâce à l'engagement de son partenaire en Afrique CTOOS Cameroun Technologies Open Source.

L'Afrique est en pleine mutation technologique. Le logiciel libre est l'un des vecteurs du développement de ce continent en offrant des applications de type professionnel aux organisations africaines soucieuses de se développer et de rester en phase avec les pays les plus avancés en terme de gestion informatique en investissant massivement dans ce domaine plus que prometteur.

Technique

Bien évidement ceci ne s'est pas réalisé en un jour puisque la création de BYOOS en France date de l'année 2005 et à cette époque les développements Intranet WEB-Based n'étaient pas ttrès répandu voire inexistant. C'est pour cela que nous avons lancé l'étude d'une GED médicale MLDOCS basée sur le Framework de XOOPS. Celui-ci nous paraissant à ce moment le plus avancé (entièrement écrit en PHP4 objet). Xoops représentait un héritage intéressant, permettant de développer dans un cadre modulaire et extensible sans équivalent encore aujourd'hui dans les offres de logiciels libres PHP-MYSQL.



Par la suite nous nous sommes rapprochés de MEDIBOARD *http://mediboard.org,* le logiciel médical open

source de référence porté par la société OpenXtrem, afin de créer le « fork » MEDLINES+ pour l'Afrique avec la prise en compte des spécificités du continent Africain. Actuellement nous traduisons le noyau MEDLINES+ dans l'espoir de supporter en totalité la langue Anglaise. Nous proposons régulièrement aux équipes de développement MEDIBOARD des patchs d'évolution vers l'internationalisation de la solution. MEDLINES+ qui est totalement autonome et embarque les librairies PEAR essentielles à son fonctionnement. Vous pouvez tester les performances de la nouvelle version 2010v2 :*http://medlines.byoos.info* (serveur au États Unis).

Maintenant notre travail est essentiellement basée sur le Framework codeIgniter (C.I) version 1.7.2 qui, à notre sens, est très loin devant en terme de fonctionnalités et simplicité de mise en œuvre *http://www.codeigniter.fr*, La documentation très complète est en cours de traduction de l'anglais vers le français ; les ressources sont nombreuses. C.I apporte une librairie extrêmement riche, de sorte que toutes applications métiers ont une réponse au cœur de C.I.

Nous travaillons à l'intégration des fonctionnalités de XOOPS au niveau application de CodeIgniter,

Un installateur simplifié et complet

Support de multiple base de données (Mysql, Mssql, PostGresSql, Oracle …)
Intégration des plugins modules avec installation et désinstallation
Gestion des versions
Héritage du mode application de C.I
Gestion multi-langue
Support des templates SMARTY la version 3.0 étant déjà intégrée à C.I (voir le tutoriel byoos.fr)

Nos missions ?

- BYOOS propose des formations de Webmaster et administrateur LINUX (Debian 5.0 – Ubuntu 10.04) en salle ou en e-learning via notre plateforme WEB,
- de la vente de matériels pour les étudiants, des revues et livres informatiques traitants des sujets LINUX et logiciels libres,

- du développement d'applications WEB, sites Internet, adaptation de logiciels existant aux besoins du client, *vTiger, Prestashop, Mediboard, IonizeCms, BambooInvoice ...,*
- Installation de serveur LINUX

Nos souhaits ?

Trouver un ou plusieurs investisseurs intéressés par le modèle open source afin de couvrir une zone plus large de l'Afrique Gabon, Guinée EQ, Congo, Côte d'Ivoire, Sénégal …

Voir le système LINUX se répandre dans les ministères et autres administrations afin de faire baisser les coûts d'acquisition des licences (le service lié au déploiement du logiciel libre est en général de 40% inférieur à celui du logiciel propriétaire). Autre argument de choc, l'absence de virus sur les plateformes LINUX en font un choix orienté sécurité.

Nos contacts

EUROPE-FRANCE
BYOOS Solutions France 05000 GAP
+33 645 254 813

AFRIQUE-CAMEROUN
BYOOS-Sarl WEB agency Solutions OPEN SOURCE
Informatique médicale-Entreprises-Technologies Web-Intranet/Extranet
LINUX DEBIAN – UBUNTU – ANTIVIRUS
Situé à l'immeuble Beau Lieu sis au carrefour ABBIA, face entrée manège
B.P 33543 Yaoundé Tel : +237 96 65 47 56,
***e-mail :* contact@byoos.fr, www.byoos.fr**

jQuery:
tour d'horizon de cette librairie JavaScript

jQuery est devenu désormais incontournable lors de la réalisation d'une application web. Utilisée aussi bien avec PHP qu'en développement Windows, on ne peut passer à côté de cette librairie aussi simple d'utilisation que puissante en terme de fonctionnalités.

Cet article explique :
- Les bases de jQuery, afin de vous permettre d'en apprendre plus sur ce dernier ainsi que de vous donner davantage l'envie de pousser votre curiosité autour de ce dernier.

Ce qu'il faut savoir :
- Le lecteur devra avoir des notions en JavaScript.
- La réalisation d'un premier développement web est conseillée.

Dans le numéro de PHP Solutions n°5/2010, Eric Vincent vous parlait des frameworks JavaScript et notamment de jQuery. Je vous propose de (re)lire son article afin d'avoir une idée plus générale sur l'ensemble des frameworks qui sont disponibles en JavaScript.

Pour l'heure, nous traiterons plus en profondeur d'un framework très prisé ces derniers temps : jQuery. Fin août 2005, John Resig publie une première version de cette librairie qui ne comporte que des modifications aux DOM ainsi que des animations basiques. En janvier 2006, le premier plugin jQuery est développé. Ce plugin, JSON for jQuery, sera le premier d'une longue lignée. Un mois plus tard, l'Ajax est ajouté à jQuery. Depuis lors, la dernière version connue et courante est la v1.4.2.

Nous continuerons la présentation de cette bibliothèque, de ce framework, par la mention de son slogan : *write less, do more*. En effet, jQuery a la particularité de réduire le nombre de ligne de code mais de faire toujours plus que si vous écriviez du code *brut*.

Prenons un exemple succinct, afin de vous montrer ce que vous pouvez éventuellement réaliser grâce à jQuery.

En ayant ce fragment de code dans un fichier JavaScript `$('img').addClass('borderDotted');` vous attacherez la classe CSS borderDotted à toutes les images présentes dans le DOM. Ainsi, chaque image qui sera rajoutée (dynamiquement ou non) à l'affichage aura la classe `borderDotted` attachée.

C'est en septembre 2007 que le plugin jQuery UI voit le jour.

Installation

Avant d'aller plus loin dans l'utilisation de jQuery, il faut avant tout procéder à son installation. Je rassure de suite les plus réfractaires de ce genre d'opération : jQuery est – comme pour la plupart des composants tiers pour le web – simple d'installation et de configuration.

Pour commencer, rendez-vous sur le site officiel de jQuery, à savoir *http://jquery.com/,* afin de récupérer la dernière version courante. A l'heure actuelle, il s'agit de la v1.4.2. Vous avez le choix entre deux version : production ou développement (version permettant de modifier le code original facilement). Par défaut, la version de production vous est proposée, et c'est celle que nous utiliserons. Nous n'avons nullement besoin de modifier la bibliothèque directement, et son poids est 85% plus faible. Ainsi, le lien direct pour la version courante ressemblera à celui-ci: *http://code.jquery.com/jquery-1.4.2.min.js.*

Terminologie

Document Object Model (ou DOM): recommandation du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents.

Tableau 1. *Liste des sélecteurs*

Sélecteur	Sélection
*	n'importe quel élément
E	élément de type E
E:nth-child(n)	élément E, le n-ième fils de son père
E:first-child	élément E, le premier fils de son père
E:last-child	élément E, le dernier fils de son père
E:only-child	élément E, seul fils de son père
E:empty	élément E qui n'a pas de fils (en incluant les nœuds textes)
E:enabled	élément de type interface (élément de formulaire) E qui n'est pas désactivé
E:disabled	élément de type interface E qui est désactivé
E:checked	élément de type interface E qui est coché (case a cocher, bouton radio...)
E:selected	élément de type interface E qui est sélectionné (options d'un select)
E.warning	élément E dont la classe est „warning"
E#myID	élément E dont l'id est „myID". Ne retourne qu'un élément maximum
E:not(s)	élément E qui ne répond pas a la condition du sélecteur s
E F	élément F descendant d'un élément E
E > F	élément F fils d'un élément E
E + F	élément F immédiatement précédé d'un élément E
E ~ F	élément F précédé d'un élément E
E,F,G	sélectionne tous les éléments E, F et G
E[foo]	élément E avec un attribut „foo"
E[foo=bar]	élément E avec un attribut „foo" valant „bar"
E[foo^=bar]	élément E avec un attribut „foo" dont la valeur débute par la chaine „bar"
E[foo$=bar]	élément E avec un attribut „foo" dont la valeur se termine par la chaine „bar"
E[foo*=bar]	élément E avec un attribut „foo" dont la valeur contient la chaine „bar"

Une fois celui-ci sauvé, il vous faut maintenant réaliser la prise en charge de la librairie par votre application, au moyen de la ligne de code suivante - placée juste après la balise *<title>* de votre document : `<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>`. Votre application est désormais prête à utiliser jQuery. Toutefois, il manque encore un léger détail qui fera toute la différence.

Utilisation

Afin de pouvoir réellement utiliser et activer jQuery dans votre application, il vous faudra mettre le code à exécuter au sein d'une fonction propre à jQuery. Comme vous pouvez le voir, il n'y a quasiment rien à écrire pour ce faire. Procédons toutefois à l'analyse de ces quelques lignes, très rapidement. Nous demandons au document d'avoir entièrement chargé le DOM (*Document Object Model*) avant d'exécuter du code proprement dit. L'appel à jQuery se fait au moyen du *$* (dollar américain). Pour plus de sûreté et de facilité, surtout dès le moment où vous utilisez d'autres bibliothèques telle que *Prototype*, vous pouvez modifier le code en Listing 1 par celui-ci.

L'instruction *noConflitct()* de jQuery vous permet ainsi de modifier l'alias de ce dernier. Le simple appel de *jQuery.noConflitc()* vous forcera à utiliser le nommage par défaut (à savoir jQuery) en place et lieu de son alias *$* (dollar américain). Vous pouvez, grâce à une assignation de cet appel, modifier l'alias de nommage, comme illustré dans le Listing 3. A partir de maintenant, il vous est entièrement possible de réaliser des traitements sur le document chargé. Afin de réaliser un traitement, jQuery a besoin de savoir sur quel(s) élément(s) vous voulez effectuer le traitement. Ainsi, outre les indéniables sélecteurs d'identifiant ou de classe, il vous est mis à disposition tout un ensemble de sélecteurs.

Prenons par exemple la ligne indiquant que tout élément de type E sera la sélection. En reprenant notre exemple introductif, nous pouvons assimiler E à un élément de type image. La sélection basée sur *$('img')* comporte dès lors autant d'objet d'élément image que le document en comporte.

Il existe aussi un mode de sélection différent : Xpath, langage (non XML) pour localiser une portion d'un document XML. Il permet de pointer (sélectionner) un élément du DOM via un chemin de localisation. Pour

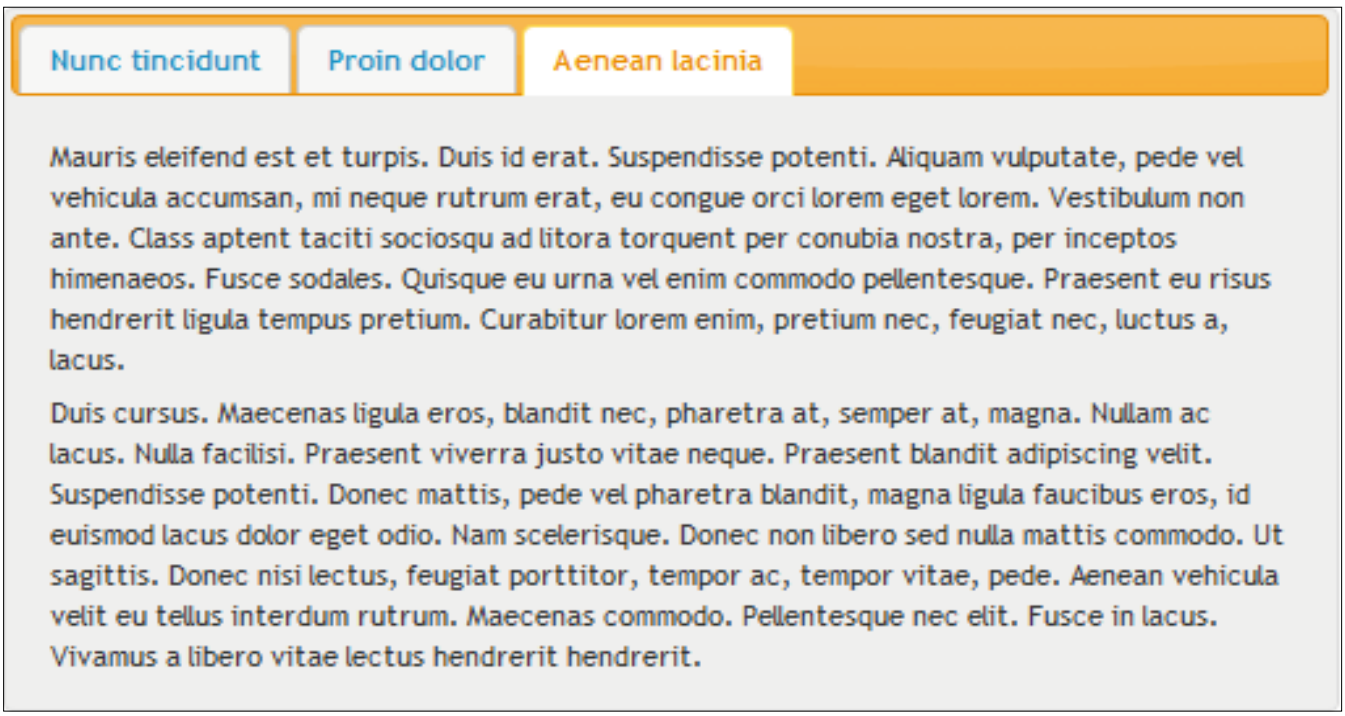


Figure 1. *Rendu visuel du widget Tabs de jQuery UI*

Listing 1. *Fonction d'appel de jQuery*

```
$(document).ready(function(){
  //Du code ici
});
```

Listing 2. *Fonction d'appel de jQuery, mode sans conflits*

```
jQuery.noConflict();
jQuery(document).ready(function(){
  //Du code ici
});
```

Listing 3. *Fonction d'appel de jQuery, mode sans conflits avec assignation*

```
var j = jQuery.noConflict();
// Utiliser jQuery avec son nouvel alias
j("div p").hide();
```

Listing 4. *Constitution du plugin, nommé afficherChaine*

```
jQuery.noConflict();
jQuery(document).ready(function(){
  (function(jQuery) {
    jQuery.fn.extend({          afficherChaine:
function(options) {           var defaults = {
textToElement: «texte mis en option.»          }
                        var options =  jQuery.
extend(defaults, options);                  return
this.each(function() {            var o = options;
var obj = jQuery(this);
 obj.text('Texte à afficher: ' + o.textToElement);
              });        }
    });
})(jQuery);
});
```

exemple, /html/body pointera sur l'élément body de la page. Pour plus d'informations sur l'utilisation de ce mode de sélection, je vous laisse le soin de visiter notamment le site suivant : *http://www.w3schools.com/xpath/default.asp.*

Les plugins

Un des atouts majeurs de jQuery est sa possibilité à s'étendre sans devoir modifier le code source tout en ayant à écrire le moins possible pour ce faire.

Il existe un nombre incalculable de ces plugins. Ces derniers vont tout aussi bien du domaine de la petite aide jusqu'à l'apport de fonctionnalités très complexes.

Nous n'avons malheureusement pas l'occasion d'en couvrir une large sélection et il m'est impossible d'en sélectionner un parmi ceux-ci, du fait qu'ils sont tous aussi utiles les uns que les autres.

Nous allons donc profiter de ce fait pour expliquer le fonctionnement et la création d'un plugin.

Comme je vous le disais au tout début de ce chapitre, la conception d'un plugin se réalise très simplement. Pour l'exemple, nous ne serons pas ambitieux dans le fonctionnement de ce dernier. Celui-ci, lorsqu'il le lui sera demandé, retournera une chaîne de caractères dans un élément donné (celui auquel est attaché l'appel du plugin).

Notre plugin est constitué et près à l'emploi, dès maintenant. Comme vous le constatez, il est très court et de nouveaux éléments sont présents. Nous allons de suite les expliquer.

Afin de créer et de déclarer un plugin pour jQuery, vous devez utiliser sa fonction `fn.extend()`. Le nom du plugin (dans le cas présent, *afficherChaine*) est laissé à votre appréciation. Cependant, il est préférable de trouver un nom unique car cela peut créer des conflits avec des plugins déclarés sous un nom identique.

Le plugin prend en charge deux données : defaults et options. Ce mécanisme vous permet de laisser une certaine liberté lors de l'appel de votre propre plugin, tout

en assurant que ces paramètres ne soient pas oubliés. De même, cela permet à l'appelant de ne pas être obligé de fournir chaque paramètre du plugin, alors que ceux par défaut lui suffise.

Dans notre exemple, nous n'avons qu'un paramètre, à savoir *textToElement*. Nous reviendrons sur ce dernier lors de l'appel à notre propre fonction.

Chaque fonction jQuery retourne un objet : l'objet (la sélection d'objets, plus précisément) modifié. C'est pourquoi il vous est tout à fait possible de chaîner les différentes fonctions en une seule ligne, telle que `jQuery('#foo').css('cursor', 'pointer').addClass('bar');`

De ce fait, nous terminons la constitution du plugin par un return obligatoire, qui retourne l'objet modifié. Dans notre cas, nous modifions l'*innerHTML* via la fonction `text()` de jQuery. Nous pouvons voir que la chaîne de caractères modifiée prend en considération le paramètre textToElement.

Pour l'appel suivant, *jQuery('#foo').afficherChaine('le texte mis en option.');* nous aurons la chaîne de caractères, *Texte à afficher : le texte mis en option*, affiché dans l'élément d'identifiant *foo*.

Jquery UI

Voyons maintenant un plugin des plus connus dans la communauté jQuery : *Jquery UI*. Ce plugin est développé par l'équipe de développeur propre à jQuery. *UI* voulant dire *User Inteface*, ou encore interface utilisateur, beaucoup d'élément liés à ce plugin sont purement visuel.

Nous pouvons aussi parler de nouvelles intéractions lorsque nous introduisons ce plugin. De fait, ce dernier amène la possibilité de faire du *drag & drop*, du *resize*, du *selectable* ou encore du *sorting*. Grâce à ces intéractions, divers plugins amènent un traitement particulier basé sur ces dernières.

De nombreux widgets sont disponibles via ce plugin. Nous pouvons citer par exemple *Dialog*, *DatePicker*. Comme vous pourrez le constater, colede jQuery se résume en une ligne : l'appel de la fonction implémentant le widget. Voici le rendu de ce code. Citons aussi *Tabs*, via l'exemple présenté dans le Listing 1.

Listing 5. *Code Html, démo du widget tabs*

```
<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula
tempus, commodo a, risus. Curabitur nec arcu. Donec
sollicitudin mi sit amet mauris. Nam elementum
quam ullamcorper ante. Etiam aliquet massa et
lorem. Mauris dapibus lacus auctor risus. Aenean
tempor ullamcorper leo. Vivamus sed magna quis
ligula eleifend adipiscing. Duis orci. Aliquam
sodales tortor vitae ipsum. Aliquam nulla. Duis
aliquam molestie erat. Ut et mauris vel pede varius
sollicitudin. Sed ut dolor nec orci tincidunt
interdum. Phasellus ipsum. Nunc tristique tempus
lectus.</p>
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis
feugiat, odio metus gravida ante, ut pharetra massa
metus id nunc. Duis scelerisque molestie turpis.
Sed fringilla, massa eget luctus malesuada, metus
eros molestie lectus, ut tempus eros massa ut
dolor. Aenean aliquet fringilla sem. Suspendisse
sed ligula in ligula suscipit aliquam. Praesent
in eros vestibulum mi adipiscing adipiscing. Morbi
facilisis. Curabitur ornare consequat nunc. Aenean
vel metus. Ut posuere viverra nulla. Aliquam erat
volutpat. Pellentesque convallis. Maecenas feugiat,
tellus pellentesque pretium posuere, felis lorem
euismod felis, eu ornare leo nisi vel felis. Mauris
consectetur tortor et purus.</p>
    </div>
    <div id="tabs-3">
      <p>Mauris eleifend est et turpis. Duis id erat.
Suspendisse potenti. Aliquam vulputate, pede vel
vehicula accumsan, mi neque rutrum erat, eu congue
orci lorem eget lorem. Vestibulum non ante. Class
aptent taciti sociosqu ad litora torquent per conubia
nostra, per inceptos himenaeos. Fusce sodales.
Quisque eu urna vel enim commodo pellentesque.
Praesent eu risus hendrerit ligula tempus pretium.
Curabitur lorem enim, pretium nec, feugiat nec,
luctus a, lacus.</p>
    <p>Duis cursus. Maecenas ligula eros, blandit
nec, pharetra at, semper at, magna. Nullam ac lacus.
Nulla facilisi. Praesent viverra justo vitae neque.
Praesent blandit adipiscing velit. Suspendisse
potenti. Donec mattis, pede vel pharetra blandit,
magna ligula faucibus eros, id euismod lacus dolor
eget odio. Nam scelerisque. Donec non libero sed
nulla mattis commodo. Ut sagittis. Donec nisi lectus,
feugiat porttitor, tempor ac, tempor vitae, pede.
Aenean vehicula velit eu tellus interdum rutrum.
Maecenas commodo. Pellentesque nec elit. Fusce
in lacus. Vivamus a libero vitae lectus hendrerit
hendrerit.</p>
    </div>
</div>
```

Requêtes Ajax

A l'ère du web de nouvelle génération, où le visuel est tout aussi important que le fonctionnel, alliant rapidité et fiabilité, rien n'est plus utile et efficace que les requêtes de type Ajax. jQuery embarque des fonctions simples et puissantes pour réaliser vos traitements Ajax en un tour de main.

L'Ajax (*Asynchronous JavaScript and XML*) est l'utilisation conjointe de technologies déjà existantes (Html, Css, DOM, ...) permettant des requêtes HTTP de types asynchrones. Lorsqu'un visiteur se rend sur une page avec formulaire et qu'il soumet ce dernier, une requête synchrone s'opère et un ensemble de contenu est rechargé. Afin d'être plus performant et rapide, certains navigateurs ont instaurés un système de cache. Toutefois, l'utilisation de requêtes asynchrones permettant de ne (re-)charger les seuls éléments dont nous avons à nouveau besoin est de loin la méthode la plus performante.

Je ferai un détour rapide par la fonction `load()` qui vous permet de charger le contenu d'un fichier et de le restituer dans un élément du DOM. Pour exemple, si le fichier *mon_fichier.txt* comporte la chaîne de caractères

Listing 6. *Code jQuery, démo du widget « tabs »*

```
$(function() {
  $("#tabs").tabs();
});
```

Listing 7. *Contenu du fichier returnPost.php*

```
<?php

  print_r($_POST);

?>
```

Listing 6. *Appel de la fonction post()*

```
jQuery.post('http://mon_url.com/plete/returnPost.php',
                    { foo: "bar", titre: "jQuery" },
                    function(data){
                      alert("Le document retourne: " +
data);
                    });
```

Listing 8. *Texte affiché dans l'alerte JavaScript*

```
Array

(
  [foo] => bar
  [titre] => jQuery
)
```

Ceci est dans le fichier et que vous effectuez un appel du style *jQuery('#foo').load('http://mon_url.com/plete/mon_fichier.txt'*); vous aurez le contenu de mon_fichier.txt affiché dans l'élément d'identifiant foo. Il vous est possible de mentionner un deuxième paramètre indiquant l'identifiant de l'élément contenu dans la page distante à afficher.

Il existe une fonction nommée tout simplement `ajax()` vous permettant d'effectuer une requête ajax, aussi bien en mode `GET` qu'en mode `POST`. Je fais le choix de ne pas vous l'expliquer en privilégiant deux méthodes distinctes mais semblables et vous laisse le soin de voir le détail de cette fonction par vous-même.

Les deux fonctions que je vais vous décrire sont `get()` et `post()`. Si vous lisez cet article, et notamment ce chapitre, vous êtes normalement familiarisés des formulaires en (x)HTML. Le nom de ces deux méthodes devraient donc vous mettre sur la voie de leur utilité respective.

La fonction `jQuery.post( url, data, success())` sera donc utilisée lorsque l'on veut effectuer une requête sur un fichier (*url*) en lui transmettant des données (*data*) et effectuer une opération spécifique lors de la réussite de cette requête. Si l'on effectue un appel de la fonction `post()` par le biais du code indiqué dans le Listing 7, nous obtiendrons une alerte JavaScript indiquant un texte semblable au Listing 8.

Il vous est donc possible d'effectuer un traitement plus complexe au niveau du serveur, lorsque la page PHP est appelée et de fournir à votre visiteur une indication sur la réussite de l'opération, tout en permettant à ce dernier de ne pas recharger le document en entier. Ce processus est notamment utilisé pour valider l'utilisation d'un login en vérifiant dynamiquement et automatiquement que ce dernier ne soit pas déjà utilisé.

La fonction `get()` n'est pas différente de celle que l'on vient de voir ensemble. Un seul détail la différencie et il ne s'agit pas de n'importe quel détail : son fonctionnement est somme toute pareil, mais il demande un retour de données à la page sans pour autant fournir des données à cette dernière. Nous pourrions par exemple utiliser cette fonction afin de récupérer le nombre de messages non lus d'un utilisateur, sans pour autant que ce dernier recharge sa page.

Conclusion

Nous avons vu dans cet article l'univers de jQuery. Pourtant, nous sommes encore loin de l'avoir couvert entièrement, vu l'ampleur des fonctions présentes au sein de jQuery ainsi que la multitude de plugins qui s'articulent autour de ce dernier.

De même, les possibilités qui nous sont offertes grâce à l'ajax implémenté au sein de jQuery sont nombreuses et vous permettent une utilisation variée et complète de cette bibliothèque.

J'espère toutefois que les quelques exemples indiqués dans cet article vous donnerons l'envie d'en découvrir d'avantages, et – qui sait – devenir l'un des développeurs de plugins sous jQuery les plus acharnés et réputés de la communauté jQuery.

Sur Internet

- *http://jquery.com/* – Site officiel de jQuery,
- *http://plugins.jquery.com/* – Dépôt (*repository*) de plugins jQuery,
- *http://jqueryui.com/* – Site officiel de jQuery UI.

JONATHAN DANSE

Autodidacte en matière de développement de sites en PHP, l'auteur a toujours poussé plus loin sa curiosité sur le sujet et les outils que font le web. Sa volonté d'apprendre et d'aller toujours plus loin dans ses réalisations lui a permis de forger un bagage conséquent sur le développement d'application PHP. Soucieux de réaliser des applications valides (aux normes du W3C), il utilise le CSS et l'(x)HTML avec parcimonie. Dans la même lignée, il s'attaque à l'apprentissage de jQuery et s'en fait un allié dans ses développements.

L'auteur est actuellement en train de finir ses études en informatique de gestion lui permettant d'être polyvalent et de voir d'autres méthodes de développement existantes.

Développer
un portail ecommerce Communautaire avec Magento et Drupal

Les solutions Magento et Drupal sont aujourd'hui deux poids lourds de l'ecommerce et de la gestion de contenu Open Source. On compte aujourd'hui plusieurs dizaines de milliers de boutiques utilisant Magento avec des références importantes telles que Wonderbox, Tf1shopping ou encore Discounteo.

Cet article explique :

- Le fonctionnement de la brique Magento – Drupal, les logiques de développement et d'amélioration de celles-ci, les outils indispensables...

Ce qu'il faut savoir :

- Connaissances sur les logiques d'e-commerce et de gestion de contenu.

La popularité de Drupal est encore plus éclatante : des centaines de milliers de sites tournent sous Drupal, et pas des moindres : la *Maison Blanche, MTV, Rue89*... Pourtant, lorsque les e-commerçants souhaitent ajouter du contenu, faire participer leurs consommateurs, offrir des outils qui ne sont pas uniquement destinés à la vente, il devient alors nécessaire de réaliser d'importants développements sur la plate-forme Magento. De manière symétrique, lorsque les drupaliens souhaitent monétiser leur audience, la solution *Ubercart* (bien que bientôt remplacée par Drupal Commerce) reste trop peu fournie en fonctionnalités propres aux processus d'ecommerce. C'est ainsi que, depuis quelque temps, de nombreuses voix se sont faite entendre pour coupler ces deux solutions. C'est la société *Adyax* qui a posé la première brique en développant une API que nous avons mis en œuvre sur plusieurs de nos projets.

Magento – Drupal : Pourquoi ?
La montée en puissance de l'ecommerce 2.0

Le choix de réunir ces deux briques fonctionnelles s'est fait à partir d'un constat assez simple : l'un des leviers essentiels de performance des boutiques en ligne réside dans la mise en œuvre d'une stratégie 2.0. Il devient ainsi crucial d'adapter les pratiques d'ecommerce aux révolutions de l'ère du participatif où les internautes deviennent acteurs des produits qu'ils achètent.

Les grandes marques ne s'y sont pas trompées : on voit fleurir ainsi les pages *Fan de* sur Facebook, Voyages-sncf vient de lancer un outil de consultation publique de ses voyageurs, les voyagistes en ligne proposent des outils de forum et d'échange en ligne de bons plans, les pages Twitter des e-marchands leur permettent de relayer les derniers bons plans et promotions, les catalogues produit mettent de plus en plus en avant les fonctions de recommandation, des applis qui permettent de poster directement des achats sur les sites communautaires, voire même d'acheter sur Facebook ...

Faire converger les meilleurs outils de l'ecommerce et du communautaire

Drupal et Magento sont chacun, dans leur domaine, les outils les plus performants du marché. Pour Magento, il n'existe pas, à ce jour, d'équivalent Open Source disposant d'une telle communauté d'utilisateurs et de contributeurs, ni de solution aussi riche en fonctionnalités. Pour Drupal, la question peut être plus discutable au regard de plate-formes telles que Wordpress, eZpublish ou encore Joomla, dont les philosophies de conception divergent. Néanmoins, son principe de fonctionnement modulaire et la richesse de ses extensions communautaires en fait une solution privilégiée pour des portails de type 2.0

Magento – Drupal : comment ça fonctionne ?

Lorsque l'on souhaite faire cohabiter deux solutions ensemble, plusieurs architectures techniques sont possibles. En fonction des objectifs du projet, chacune des structures peut être pertinente.

Deux sites différents avec SSO et CAS

La première solution, la plus habituelle pour faire dialoguer différentes solutions, est de déployer un mécanisme de SSO avec l'ajout d'un serveur CAS. Cette logique de fonctionnement oblige à conserver deux *Front-Office* différents (bien que l'on puisse les habiller avec une même charte graphique) et à mettre en place des procédures de réplication de comptes utilisateurs.

Il faut ensuite prévoir de déléguer à un serveur CAS les fonctions d'authentification pour éviter que les utilisateurs n'aient à renouveler leur authentification lorsqu'ils passent d'un *front-office* à un autre.

Mais l'intérêt de coupler Drupal – Magento ne réside pas que dans la synchronisation des comptes utilisateurs. Il faut aussi pouvoir multiplier les accès aux contenus : proposer des produits depuis les outils communautaires et proposer des contenus depuis la boutique. Il faut aussi prévoir de multiplier les contenus proposés selon deux mécanismes, de plus ou moins grande simplicité :

- mettre en place des flux RSS pour tous les contenus que l'on souhaite afficher sur l'autre *Front-Office*,
- déployer des imports – exports XML permettant d'alimenter la base Drupal / Magento, avec la principale contrainte que la structuration de données de Magento n'est pas forcément adaptée à un principe de structuration de contenu tel que l'on peut l'entendre dans une logique CMS.

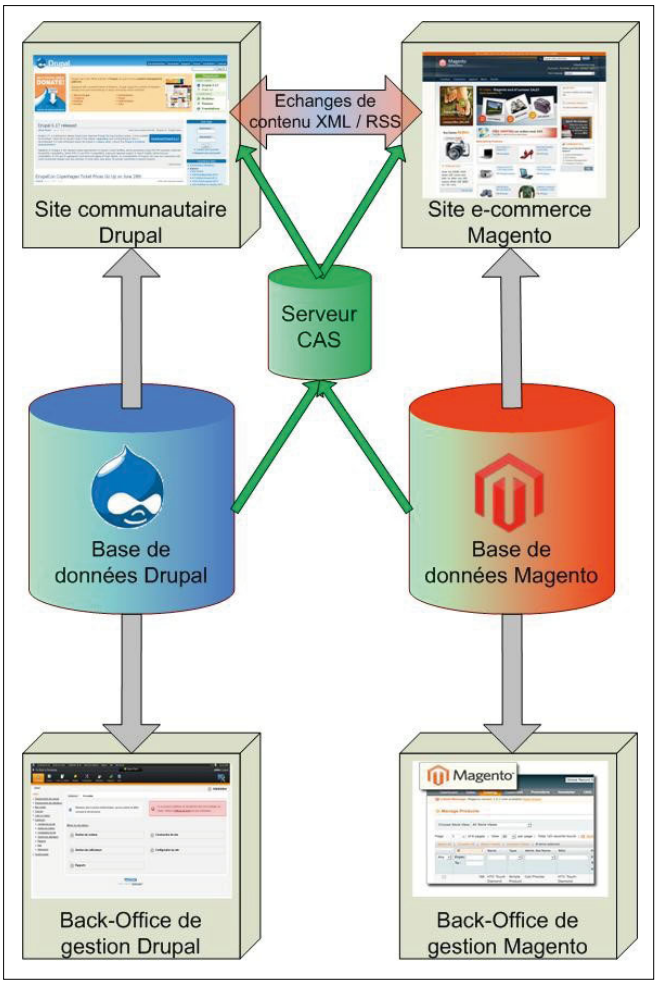
Nous vous présentons dans la Figure 1 une structuration possible de cette architecture.

Avantages de cette logique de fonctionnement :

- les deux plate-formes restent indépendantes : il n'y a pas besoin d'installer de modules complémentaires et donc pas de risques liés aux montées de version de ces plate-formes.
- Pas de complexité particulière de synchronisation. Chaque plateforme conserve ses avantages.

Inconvénients :

- L'intégration d'un serveur CAS sur des outils PHP n'est pas d'une grande simplicité. CAS est à l'origine une application développée en Java. Pour Drupal, il est donc nécessaire d'utiliser la librairie *phpCAS*, qui se retrouve intégrée dans le module CAS (ici : *http://drupal.org/project/cas).* Et ce module n'est pas compatible dès lors que vous utilisez des fonctions d'extension de profil sous Drupal. Pour Magento, il n'existe tout simplement pas d'extension CAS. Donc à vous d'en développer une...



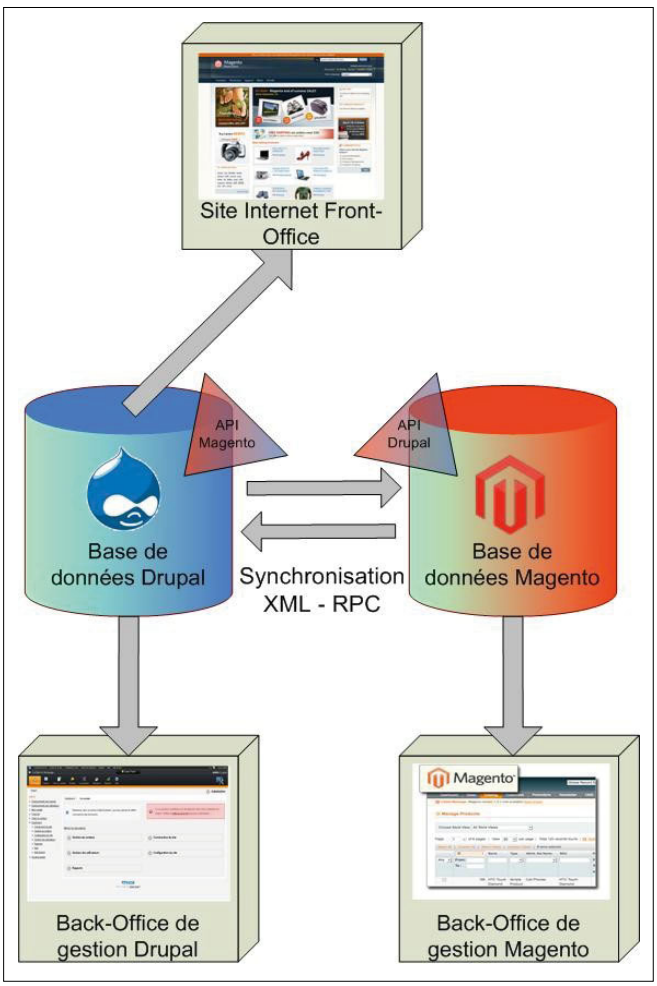
- Si vous souhaitez intégrer un moteur de recherche unique au site, ce n'est pas possible, à moins d'aller installer un moteur tiers (et ces solutions sont souvent payantes).
- Dès que vous êtes amenés à faire des modifications graphiques, il vous faut répercuter ces modifications sur les deux instances, avec une double charge de travail puisque les moteurs de *templating* entre Drupal et Magento sont différents.
- …

En bref, outre la contrainte d'avoir à développer une extension CAS pour Magento, cette solution n'est pas très efficace pour les internautes. Nous vous proposons donc une deuxième solution.

Un seul Front, deux Back-Office

La deuxième solution, celle que nous avons retenue, réside dans l'utilisation d'un seul *Front-Office*, celui de Drupal. Pour les développeurs, il en résulte que l'ensemble du site devra être conçu en utilisant les fonctions de theming de l'outil, ce qui constitue certainement un avantage, au regard de la relative complexité de theming de Magento.

En revanche, les administrateurs devront utiliser deux *Back-Offices* de gestion : le *back-office* Magento



est utilisé pour alimenter le catalogue, gérer les règles de promotion, consulter les commandes clients, … Le *Back-Office* Drupal sert à gérer l'intégralité des contenus du site ainsi que les fonctions communautaires.

Nous vous présentons dans la Figure 2 un schéma de fonctionnement de cette logique.

Utilisation de deux API : une API Magento, une API Drupal

Cette deuxième solution utilise deux API, installées respectivement sur Magento et Drupal. Ces deux API permettent de réaliser la synchronisation XML – RPC entre les bases de données Magento et Drupal.

L'API Magento permet de mettre à disposition de l'ensemble du catalogue produit et des règles de prix, qui sont ensuite répliquées dans Drupal. Côté Drupal, cette API permet de récupérer l'ensemble des informations catalogue saisies dans Magento et redistribue ensuite les commandes clients passées depuis le *Front-Office* Drupal vers le *Back-Office* Magento.

Cette logique de fonctionnement impose une structuration équivalente du catalogue et des produits : l'ensemble du catalogue, à la fois présent sur Magento et sur Drupal, doit présenter des attributs strictement équivalents. Si, du côté de Magento, cette structuration est native, il faut, du côté de Drupal, installer un certain nombre d'extensions pour pouvoir réaliser cette structuration, notamment avec l'utilisation du classique module CCK et *Views*.

L'intérêt d'utiliser Drupal côté Front-Office. Des performances au rendez-vous

L'ensemble des pages du site ainsi conçu est généré depuis Drupal. Quand on connait la certaine lourdeur des applications Magento, un tel système est fortement recommandé pour optimiser les temps de réponse, et ce d'autant plus que la communauté Drupal propose de nombreux modules d'optimisation, avec des fonctions de générateur de cache statique, de load *balancing*, de *clustering*...

Un moteur de recherche puissant

Cet avantage est renforcé par l'utilisation du moteur de recherche Drupal, permettant notamment d'utiliser SolR, extension du moteur de recherche de Drupal, (reposant sur la technologie *Lucène*).

Un certain nombre de grands commerçants ayant décidé de migrer vers Magento se sont en effet vite trouvés contraints par l'utilisation du moteur de recherche natif de Magento, ce dernier devenant inefficace dès que l'on dépasse un nombre de produits ou références supérieures à 50 000.

Ainsi, au lieu d'intégrer directement un moteur de recherche externalisé, l'utilisation de Drupal permet de pallier aux faiblesses actuelles de Magento.

Une gestion avancée du profilage des internautes

Enfin, l'utilisation de Drupal permet de recueillir un nombre très important d'information sur les internautes du portail. Tous les professionnels du marketing le savent bien : plus vous en savez sur vos consommateurs, plus vous pourrez leur proposer des offres adaptées qu'il auront autant de chance d'acheter.

Si Magento permet d'augmenter sensiblement le nombre d'informations demandées aux clients, ces derniers seront d'autant plus enclin à les diffuser s'ils peuvent ainsi enrichir leur profil, et que vous leur donnez l'occasion de s'exprimer au travers d'outils communautaires.

Les fonctionnalités pertinentes

Drupal est un outil qui propose différentes logiques de fonctionnement multisite, permettant ainsi de créer, avec une boutique, différents espaces communautaires. Dans cette logique, il est ainsi très efficace de créer plusieurs instances thématiques, composées de plusieurs boutiques, d'un espace communautaire, d'un site institutionnel et éventuellement d'un blog. Cette logique multisite permet de multiplier les liens internes (pratique favorable au référencement), positionner des produits sur toutes les pages de contenu, et offrir de l'information de qualité à vos visiteurs qui ne reviendront

Les sites à consulter

- *www.drupal.org,*
- *www.magentocommerce.com ,*
- *http://drupal.org/project/magento,*
- *http://www.magentocommerce.com/extension/1020/drupal.*

pas seulement pour acheter, mais aussi pour s'informer et participer aux communautés ainsi créées.

Nous vous listons ci-dessous les fonctionnalités qu'il peut être intéressant de proposer :

- outils de gestion de blogs, ou multi-blog avec la gestion des commentaires et des notations des internautes,
- outils de forum et d'échanges entre internautes, qui peut servir également de service client et relayer le support téléphonique,
- outils de newsletter, permettant d'exploiter la richesse de composition de Drupal : newsletters thématiques, newsletters produits, newsletters ciblées par profil et intérêt des internautes, …
- ...

Conclusion

Nous vous avons fait découvrir ici le fonctionnement de Magento – Drupal en utilisant les API proposées par la communauté. Ces dernières restent cependant encore à améliorer et de nombreux patchs sont à appliquer avant de pouvoir vraiment les utiliser en production. Néanmoins, cette logique d'association montre encore une fois la force des logiques Open Source contributives, et combien également aussi les internautes attendent d'une boutique en ligne : il ne s'agit plus de reproduire des logiques marketing traditionnelles mais bien d'offrir de nouveaux moyens de participation et d'échange.

PIERRE MOURIER

Lead technique au sein de la société Core-Techs, l'auteur intervient régulièrement auprès de nombreux clients pour des missions de développement de sites Internet, de plate-forme d'ecommerce ou de solutions métiers interactives. Il utilise quotidiennement les solutions Magento et Drupal et a collaboré de façon active à l'amélioration des API Drupal / Magento.

Créer un SVA
avec la passerelle SMS/WAP : Kannel

Les services à valeur ajoutée sont devenus incontournables dans le monde de la télécommunication. Dans cet article vous apprendrez à créer un service à valeur ajoutée depuis un script PHP et aussi à administrer votre serveur à travers un script bash par l'envoi de sms.

Cet article explique :
- Ce qu'est un service à valeur ajoutée.
- Comment installer, configurer la passerelle SMS Kannel.
- Comment créer un service à valeur ajoutée simpliste.
- Comment créer un service d'administration à distance par sms.

Ce qu'il faut savoir :
- Vous devez connaître les bases du langage PHP, du bash et des notions sur la télécommunication.

L'ère de la téléphonie mobile a entrainé ces dernières années l'emploi de service à valeur ajoutée. Ce dernier a permis aux usagers de pouvoir participer à des **services** que l'opérateur et/ou une entreprise mettent à leur profit en temps réel soit à la demande des clients sur un service donné soit par des informations en provenance d'une entreprise via l'opérateur.

Il faut comprendre que le jargon de la télécommunication parle de **SMS-MO** (*Message Originated*) si le message envoyé provient de l'usager de la téléphonie mobile ou dans un autre sens une requête dite **PULL**. Si le message provient de l'opérateur ou centre de messagerie, il s'agira de **SMS-MT** (*Message Terminated*) ou dans un autre sens, un envoi de type **PUSH**.

Le principe de base est que les messages doivent être confiés à un centre de messagerie (SMSC ou *Short Message Service Center*) du réseau **GSM**. C'est lui qui se chargera de les faire parvenir dès que possible à leurs destinataires, où qu'ils se trouvent. Dès que possible, signifie en l'occurrence dès que le mobile sera tout à la fois sous tension, dans une zone couverte par un réseau GSM, et autorisé à communiquer sur celui-ci.

Tant que ces conditions ne sont pas remplies, le message est archivé par le centre de messagerie jusqu'à concurrence de la durée de validité qui a été spécifiée par son émetteur (souvent 72 heures, mais éventuellement beaucoup plus ou beaucoup moins) ou refuser l'accés à d'autres SMSC autres que le sien, tout en mettant celui-ci à la disposition de ses seuls clients.

Quoiqu'il en soit, la taille actuelle des messages ne doit pas depasser les 160 caractères mais peut être quadruplée par concaténation.

La mise en place de ce service nécessite une interconnexion entre les systèmes GSM et informatique, ce qui impose le déploiement d'une passerelle SMS.

Prise au sens large, une passerelle est un outil permettant de passer d'un système à un autre. Au sens strict du terme, une passerelle est un dispositif destiné à connecter des systèmes de téléinformatiques ayant des architectures différentes, des protocoles différents, ou offrant des services différents.

La passerelle doit alors dépouiller la trame des informations spécifiques aux protocoles émetteurs et les remplacer par leurs équivalents dans les protocoles récepteurs. L'inconvénient majeur de ce système est

Listing 1. *kannel.conf*

```
group = core
admin-port = 13000
admin-password = admin
status-password = zoroastre
smsbox-port = 13001
# Fichier dans lequel les logs sont ecrits
log-file = "/var/log/kannel/log/kannel.log"
log-level = 1
# Fichier dans lequel les informations sur les SMS reçus/
envoyés sont stockés, en vue des statistiques.
access-log = "/var/log/kannel/access/bearerbox-access.
log"
# Fichier dans lequel tous les messages reçus sont
stockés jusqu'à la fin du traitement de messages.
store-file = "/var/log/kannel/store/kannel.store"
```

qu'une telle application doit être disponible pour chaque service (FTP, HTTP, Telnet, SMS, etc.).

L'une des difficultés de la messagerie SMS est que les SMSCs sont gérés par des protocoles de communication propriétaires. En d'autres termes, il n'existe pas encore de normalisation proprement dite.

Il existe une pléiade de passerelles SMS dont certaines sont propriétaires (*Alligata, Ozeki SMS, Wapme, Jataayu SMS Gateway*), tandis que d'autres sont dans le monde de l'Open Source et en perpétuel developpement (Kannel, Gammu, ...).

Revenons sur le principe plus etendu du service à valeur ajoutée. Le client envoie un sms contenant un mot-clé (keyword) suivi de texte vers un numéro court (short number) pour bénéficier d'un service proposé par un éditeur. Le numéro court identifie le destinataire du message et peut être obtenu suivant la régularisation des agences de télécommunication en vigueur.

Dans cet article, nous parlerons de service à valeur ajoutée avec l'utilisation de la passerelle Open Source Kannel à travers le SMS.

Kannel une passerelle Open Source

Kannel, passerelle Open Source sous licence *FreeBSD* conçue pour pouvoir communiquer avec plusieurs centre de messagerie utilisant différents protocoles est à l'initiative de la compagnie finlandaise *Wapit.* Elle est écrite en langage C, et fourni une passerelle mixte SMS et WAP. En outre, elle permet également de gérer le *push,* le *pull* ou *pull-push* SMS c'est-à-dire respectivement, l'envoi des messages aux entités extérieures, la réception d'un SMS ou la réception de requête, puis l'envoi de la réponse après traitement à partir de la plateforme. Au niveau de l'application se fait le traitement de la requête, puis l'envoi de la réponse via le SMSC de rattachement à l'abonné demandeur. L'architecture technique et fonctionnelle de cette passerelle n'est pas l'objet de cet article pour plus d'informations consulter le site *http://www.kannel.org.*

Pre-requis

Kannel est développé sous les systèmes Linux, et devrait facilement être exporté sur d'autres systèmes *Unix-like*. Cependant, les autres plateformes ne sont pas intégrés. Kannel exige l'environnement logiciel suivant :

- Compilateur C et les bibliothèques pour ANSI C, avec des extensions *Unix* normales telles que les sockets BSD et outils relatifs.
- La bibliothèque Gnome XML (*gnome-xml et libxml*), version 2.2.5 ou plus récente. Voire *http://xmlsoft.org/xml.html*.

Si l'installation se fait à partir des paquets de distribution, il sera exigé *libxml2-dev* en addition du *run-time libxml2*.

Listing 2. *kannel.conf (suite)*

```
# Section SMSC
group = smsc
smsc = http
system-type = "kannel"
#Définit où envoyer les messages MT, l'emplacement de
la passerelle qui va relayer les messages.
send-url = "http://localhost:13013/cgi-bin/sendsms"
# Port d'écoute des messages MO venant de la
passerelle
port = 13020
# Adresse IP (nom hôte) autorisé.
connect-allow-ip = "*.*.*.*"
# Nom d'utilisateur associé à la connexion
smsc-username = bargny
# Mot de passe pour l'utilisateur
smsc-password = maodo
```

Listing 3. *kannel.conf (suite)*

```
#section smsbox, variable obligatoire
 group = smsbox
 #Identifiant d'instance smsbox, parametre optionnel.
 smsbox-id = boxsms
 #La machine sur laquelle le bearerbox se trouve
 bearerbox-host = localhost
# Indique si le smsbox se connecte au bearerbox
directement ou s'il passe par un sqlbox.
 bearerbox-is-sqlbox = true
 # Numéro de port auquel les requêtes HTTP d'envoie
de messages, sendsms, sont faites.
 sendsms-port = 13013
 # Seuls ces caractères sont autorisés dans le
paramètre "to" dans la requête HTTP de soumission de
message.
 # Le caractère espace a une signification
particulière, utilisé pour séparer les numéros de
téléphones dans un multi-send.
 sendsms-chars = "0123456789 +-"
 #Conversion des messages reçus avec un charset UCS-
2 en UTF-8,
# simplifiant ainsi le travail du serveur externe.
 mo-recode = true
 # Pour fixer l'expéditeur des messages (mettre en
commentaire , si on utilise le paramètre "from" dans
la requête d'envoie de SMS).
 global-sender = bargny _ sender
 log-file = "/var/log/kannel/log/smsbox.log"
 log-level = 0
 access-log = "/var/log/kannel/access/smsbox-access.
log"
 sendsms-url = /cgi-bin/sendsms
```

Listing 4. *kannel.conf (suite)*

```
group = sendsms-user
username = bugsbunny
password = bipbip
max-messages = 3
concatenation = true
```

Listing 5. *kannel.conf (suite)*

```
group = sms-service
keyword = poids
post-url = "http://localhost/sav _ poids.php?fr=%p&to=%
P&txt=%a&smscid=%i"
catch-all=true
concatenation = true
```

- *GNU Make*.
- *GNU Bison* 1.28, si vous voulez modifier le compilateur *WMLScript*.
- Les outils de traitement *DocBook*.
- *GNU Autoconf*, s'il est nécessaire de modifier le script de configuration.

Listing 6. *sav-poids.php*

```
<?php
<?php
$host = "localhost";
$db = "services";
$db _ user = "root";
$db _ pwd  = "sav _ sms";

  $from = $ _ GET['fr'];
  $to = $ _ GET['to'];
  $text = $ _ GET['txt'];
  $smsc = $ _ GET['smscid'];
       //Connection au serveur de la base de données
      $link = mysql _ connect($host,$db _ user,$db _ pwd)
or  die('POIDS : Not connected : ' . mysql _ error());
      mysql _ select _ db($db) or die("POIDS : Could not
select database");
        $date _ recep = date("Ymd");
        $id _ short _ numbers="0";
        if($to == "70070" && $smsc == "platon"){
            $id _ short _ numbers="66";
            $smsc = "1";
         }
        if($to == "70070" and $smsc == "zeus"){
                    $id _ short _ numbers="67";
                    $smsc = "2";
          }
            $from = $from*1;
        //Insérer MO
          $query = "INSERT INTO recep _
msg set message='$text',date _ recep='$date _
recep',date=now(),date _ recep2=now(),num _
expediteur='$from',num _ dest='$to',statut _ traitement
='10',compteur='0',smsc='$smsc',id _ service='1746',id _
short _ numbers='$id _ short _ numbers'";
          $result = mysql _ query($query) or die("IMC :
Query failed");
      // Libération des résultats
      //  mysql _ free _ result($result);
     // Fermeture de la connexion
       mysql _ close($link);
      if ( $to!="70070"){
                          //MT : message envoyé à
l'usager du service
          echo "POIDS : erreur, merci d'envoyer votre
message au numero court indique. ";
          exit;
       }
        $tab=explode(" ",$text);
       if ( count($tab)<3 ){
                          //MT : message envoyé à
l'usager du service
          echo "POIDS : message incorrect. Envoyer
POIDS espace TAILLE espace CIRCONFERENCE poignet. ";
          exit;
        }
      //Calcul de l'IMC formule de Quetelet
      $taille=$tab[1];
      $circonf=$tab[2];
      $imc=($taille-100+4*$circonf)/2;
      $maxi=$imc+5;
      $mini=$imc-5;
          //MT : Reponse envoyée à l'usager via la
passerelle
      echo "Votre poids ideal est $imc kg. Pour une
meilleure sante, essayez de maintenir votre poids
entre $mini kg et $maxi kg. Merci!.";

?>
```

Installation

L'installation peut se faire à travers la récupération du code source soit par l'utilisation des utilitaires d'installation rpm, apt. Nous allons récupérer le code source à l'adresse suivante pour la version courante *1.4.1 gateway.1.4.1.tar.gz*.

Il suffit de faire :

- décompression de l'archive : *tar xzfv gateway.1.4.1.tar.gz,*
- configuration avec l'option d'utiliser mysql et le mettre dans le répertoire, d'installation */usr/local : ./configure* `--with-mysql --enable-start-stop-daemon -bindir=/usr/local/gateway`,
- installation proprement dit : *make && make install*.

Configuration

Une fois l'installation faite, il suffit de configurer la passerelle selon les besoins liés à l'implémentation de service à valeur ajoutée. Pour le cas de notre article, nous utiliserons la configuration minimale :

- configuration du ***bearerbox*** obligatoire,
- configuration du ***smsbox*** pour l'envoi de SMS.

Configuration du bearerbox

Il suffit de configurer le groupe , l'accès à l'interface d'administration (nom et mot de passe), le port d'écoute pour l'administration de kannel, son protocole, les fichiers de log et leur niveau de sévérité, le répertoire de stockage des messages reçus etc... Le Listing 1 detaille la configuration du bearerbox.

Il suffira de configurer aussi les centres de messagerie (**SMSC**) que Kannel pourra intercepter. Il suffira d'ajouter ces lignes dans le fichier du Listing 1. Le Listing 2 permet de configurer les centres de messagerie.

Configuration du smsbox

Le groupe smsbox doit être définit dans le fichier de configuration pour utiliser la passerelle SMS. Sa configuration permet de gérer l'envoi et la réception de sms et est détaillée dans le Listing 3. Pour les besoins de test d'envoi de SMS via l'interface web, nous configurons le nombre maximal de SMS autorisé à envoyer par l'utilisateur concerné à travers le Listing 4.

C'est de là qu'interviennent les services à valeurs ajoutés. Chaque service utilisé doit être configuré en mettant le mot clé dédié à ce service et l'application qui se chargera de traiter les requêtes. Notre service à valeur ajoutée consiste à informer l'usager de son état de santé par la formule de *Quetelet*. Le Listing 6 fournit une implémentation en php de notre service à valeur ajouté.

La table de réception des messages en vue de faire des statistiques à l'avenir sur le service est donnée dans le Listing 7.

Le Listing 5 permet de définir le service à valeur ajouté.

Parmi les variables qui spécifient le type de traitement on peut citer :

- ***post-url*** définit l'application http qui traitera la requête Celle-ci est utilisée dans cet article,
- ***file*** donne le fichier local à retourner,
- **text** indique le texte à retourner comme réponse à la requête,

- ***exec*** permet de spécifier la commande shell à exécuter lorsque le mot clé est envoyé. Variable utilisée en fin d'article pour une administration.

Une fois la passerelle configurée, il suffit de démarrer par ordre en tache de fond :

- Démarrer d'abord la *bearerbox,*
- ***/usr/local/gateway/gw/bearbox -v1 /etc/kannel/kannel.conf &.***
- Enfin démarrer le smsbox qui attends les connexions du *send_sms* et envoies les données au bearerbox pour former le paquet adéquat.
- ***/usr/local/gateway/gw/smsbox -v1 /etc/kannel/kannel.conf &.***

Une fois que la passerelle est bien démarrée, il suffit de faire le test qui consiste à l'envoi de sms via le navigateur avec l'utilitaire *lynx* ou un autre curl, *elinks, w3m* etc...

```
[root@bargny]curl "http://localhost:13013/cgi-bin/sendsms?username=bargny&password=maodo&text=POIDS 65 190 2 &to=+221777872806&from=70070".
```

Si le message arrive à destination, cela veut dire que notre service à valeur ajoutée peut être testée avec le mobile en envoyant ***POIDS [poids] [taille] [circonférence]*** **au numéro court 70070.**

Au niveau des messages de logs de Kannel il suffit de faire la recherche sur le numéro 221777872806 par un *grep.*

[root@zorosatre /$ cat /var/log/kannel/access/bearerbox-access.log | grep 221777872806

*2010-06-13 13:13:41 Receive SMS [SMSC:zeus] [SVC:] [ACT:] [BINF:] [FID:] [from:+221777872806] [to:70070] [flags:-1:0:-1:0:-1] [msg:**POIDS 65 120 6**] [udh:0:]*

2010-06-13 13:13:42 Sent SMS [SMSC:zeus] [SVC:bargny] [ACT:] [BINF:] [FID:] [from:70070] [to:221777872806] [flags:-1:0:-1:-1:-1]

Listing 7. *recep_msg.sql*

```
CREATE TABLE IF NOT EXISTS 'recep_msg' (
  'ref_msg' int(10) unsigned NOT NULL auto_
increment,
  'message' varchar(255) default '0',
  'date_recep' varchar(12) NOT NULL default '',
  'heure_recep' varchar(12) NOT NULL default '',
  'date' datetime NOT NULL default '0000-00-00
00:00:00',
  'date_recep2' datetime NOT NULL default '0000-00-00
00:00:00',
  'num_expediteur' varchar(12) NOT NULL default '',
  'num_dest' varchar(15) NOT NULL default '',
  'msg_id' varchar(10) default '0',
  'statut_traitement' int(2) default '0',
  'compteur' int(2) NOT NULL default '0',
  'smsc' int(3) unsigned default '1',
  'id_short_numbers' int(11) NOT NULL default '0',
  'categorie_msg' varchar(20) NOT NULL default '',
  'id_categorie' int(11) NOT NULL default '0',
  'id_categorie_service' int(7) default NULL,
  'id_service' int(11) NOT NULL default '0',
  'ref_msg_init' int(11) NOT NULL default '0',
  PRIMARY KEY  ('ref_msg'),
  KEY 'id_service' ('id_service'),
  KEY 'id_categorie' ('id_categorie'),
  KEY 'id_short_numbers' ('id_short_numbers')
) ;
```

Listing 8. *kannel.conf (suite)*

```
#Ajouter un groupe sms-service
group = sms-service
#Définir un mot clé
keyword = service
#Définir le type d' application associé
exec = service %s %s
```

[*msg:82:Votre poids ideal est 14kg*. **Pour une meilleure santé, essayez de maintenir votre poids entre 9kg et 19 kg. Merci !**] *[udh:0:]*

Les logs nous font part du message envoyé et celui reçu par l'utilisateur.

Arrêter la passerelle

La manière classique d'arrêter la passerelle Kannel est faite en ordre inverse de son démarrage à savoir.

- **Smsbox**,
- **bearerbox**.

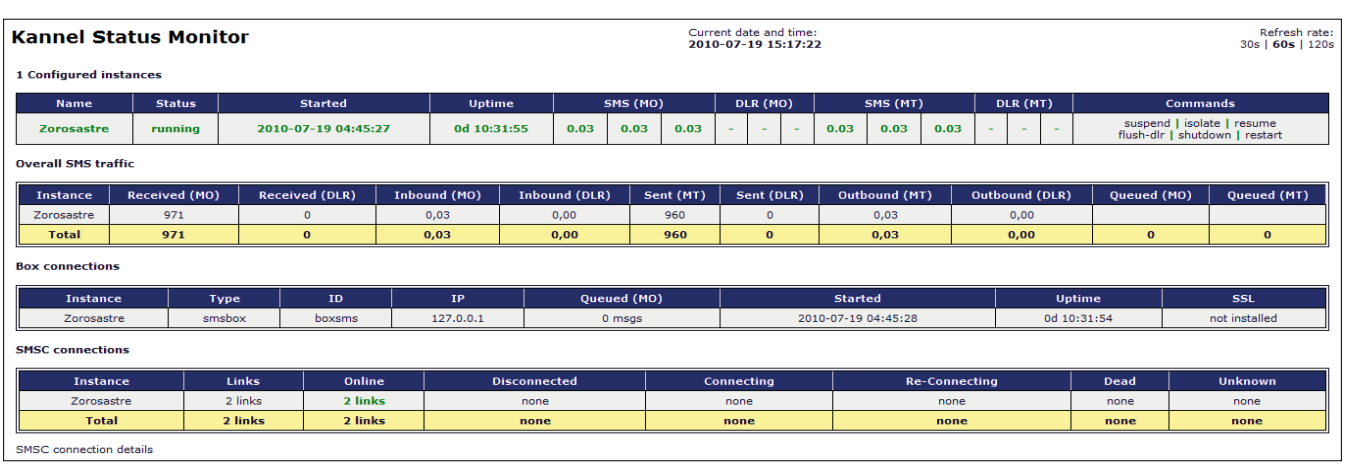
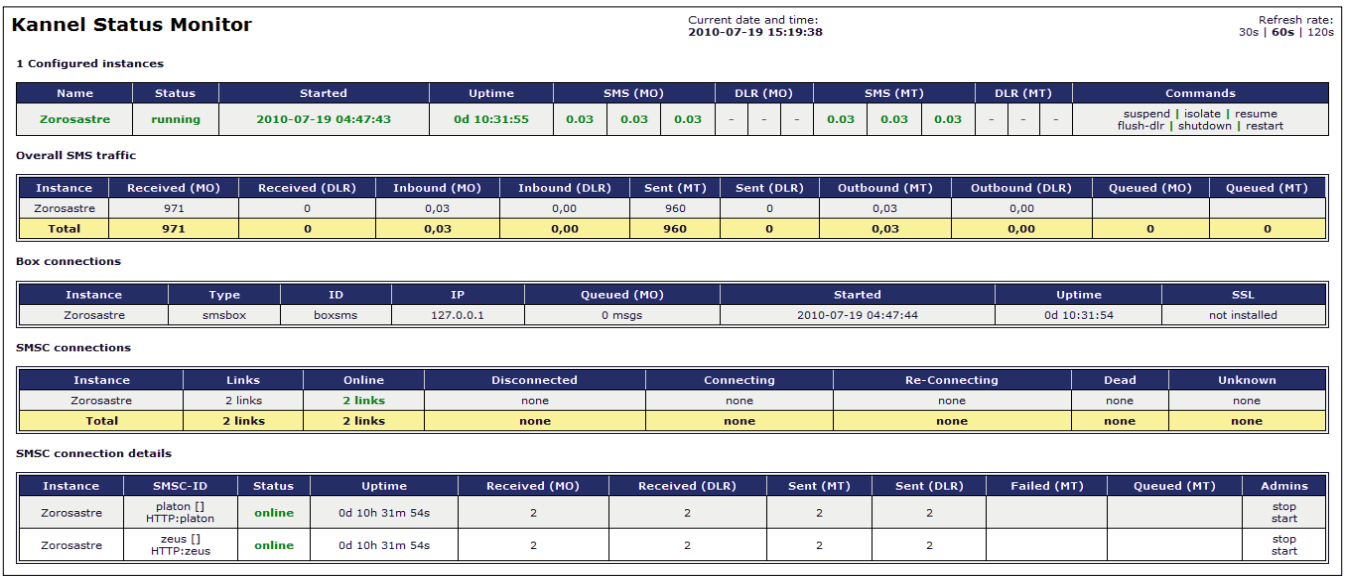


Figure 1. *Le status global de la passerelle*



Kannel Status Monitor

Current date and time: **2010-07-19 15:19:38**

Refresh rate: 30s | **60s** | 120s

1 Configured instances

Name	Status	Started	Uptime	SMS (MO)			DLR (MO)			SMS (MT)			DLR (MT)			Commands
Zorosastre	running	2010-07-19 04:47:43	0d 10:31:55	0.03	0.03	0.03	-	-	-	0.03	0.03	0.03	-	-	-	suspend | isolate | resume flush-dlr | shutdown | restart

Overall SMS traffic

Instance	Received (MO)	Received (DLR)	Inbound (MO)	Inbound (DLR)	Sent (MT)	Sent (DLR)	Outbound (MT)	Outbound (DLR)	Queued (MO)	Queued (MT)
Zorosastre	971	0	0,03	0,00	960	0	0,03	0,00		
Total	**971**	**0**	**0,03**	**0,00**	**960**	**0**	**0,03**	**0,00**	**0**	**0**

Box connections

Instance	Type	ID	IP	Queued (MO)	Started	Uptime	SSL
Zorosastre	smsbox	boxsms	127.0.0.1	0 msgs	2010-07-19 04:47:44	0d 10:31:54	not installed

SMSC connections

Instance	Links	Online	Disconnected	Connecting	Re-Connecting	Dead	Unknown
Zorosastre	2 links	**2 links**	none	none	none	none	none
Total	**2 links**	**2 links**	**none**	**none**	**none**	**none**	**none**

SMSC connection details

Instance	SMSC-ID	Status	Uptime	Received (MO)	Received (DLR)	Sent (MT)	Sent (DLR)	Failed (MT)	Queued (MT)	Admins
Zorosastre	platon [] HTTP:platon	online	0d 10h 31m 54s	2	2	2	2			stop start
Zorosastre	zeus [] HTTP:zeus	online	0d 10h 31m 54s	2	2	2	2			stop start

Figure 2. *Le detail des centres de messageries configurés au niveau de la passerelle*

Administration d'une machine par SMS

La passerelle Kannel nous offre aussi la chance d'administrer à distance nos machines via le paramétrage de service à valeur ajoutée. Le mécanisme est semblable à celui du paragraphe précédant mais à la différence que les paramètres seront transmis au shell qui se chargera de l'exécution.

La procédure est la suivante :

- Ajouter un groupe sms-service :
- *group = sms-service.*
- Définir un mot clé :
- *keyword = service.*
- Définir le type d'application associé :
- *exec = **service** %s %s.*

Où :

- ***service*** est la commande ***service Linux*** qui permet de gérer les services par les options suivantes start, stop, restart .
- le premier %s représente la première chaîne de caractère après le mot clé. Il s'agit des services disponibles tels ssh, mysql, vsftp, ldap.
- le second ***%s*** représente la seconde chaîne après le mot clé pour les options des services *start, stop, restart.*

Ainsi lorsqu'on envoie à partir d'un mobile : ***service ldap restart*** l**e** système exécutera la même commande. La configuration complète du service d'administration se trouve dans le Listing 8 .

Sur Internet

- *http://www.kannel.org/userguide.shtml* – Manuel de la passerelle SMS.

NB : Pour éviter de rendre vulnérable le système, nous n'allons réserver l'accès à ce service qu'à un seul utilisateur (l'administrateur) en mettant son numéro seul dans white-list.

white-liste = "http://localhost/reserved/list.txt"

Evidemment dans le fichier *list.txt* il y a les numéro de ceux qui peuvent utiliser le service.

Pour finir nous ajoutons un petit service qui permet aux usagers de PHPSolmag de recevoir sur leur mobile la parution d'un nouveau numéro en envoyant **magazine** au 7070 très simple. Service qui sera rendu disponible une fois que le numéro est disponible. Le sms reçu est *Bienvenue* ! Le numéro courant est disponible en version pdf ….

group =sms-service
keyword = magazine
exec = /bin/echo 'Bienvenue ! Le numéro courant est disponible en version pdf ….'

Ce petit service peut être adapté suivant les besoins du magazine PHP le plus populaire au monde !

Conclusion

On ne peut se passer des services à valeur ajoutée. Ils sont devenus incontournables aujourd'hui. Le seul inconvénient de Kannel est qu'il est écrit en langage C sous Linux non portable et que les fichiers de configuration ne sont pas au format universel qu'est le XML.

MAODO DIOP

Maodo DIOP, Consultant en Développement Logiciel a un background de maintenance logicielle et sur l'audit des systèmes Linux compromis. Il s'intéresse aux SVA et à l'infographie.
Pour le contacter : dmaodo@{gmail,yahoo}.com

Acquérir de la vitesse
avec Zend Framework – 2ème partie

Voici le second article de la série Acquérir de la vitesse avec Zend Framework. Dans celui-ci, je souhaite vous présenter la connexion d'une base de données et l'accès aux données de celle-ci, la connexion avec Doctrine, puis l'usage des helpers Zend !

Cet article explique :

- Comment choisir et utiliser les librairies PHP du framework Zend.
- Les règles de programmation à respecter pour chacune des librairies présentées.
- Les pièges à éviter pour avoir un développement propre et sécurisé.

Ce qu'il faut savoir :

- Rien, à part connaître PHP et je suppose que vous avez lu les autres articles sur Zend parus dans PHP Solutions.

Pour pouvoir accéder à une base de données la première étape consiste à configurer les connecteurs du framework. Cela peut se faire de plusieurs manières. Mais c'est avec la classe `Zend_Db_Adapter` que l'on va créer un pont entre les classes de manipulation des données et votre base de données physique. Pour cela, cette classe s'appuie sur l'interface objet PDO (*PHP Data Objects*). Pour configurer ce connecteur vous devrez définir le type de base, les données de connexion : adresse du *Host* (*Localhost* pour un WAMP ou l'adresse IP de votre serveur qui porte votre base de données), le login et mot de passe de connexion à votre base et le nom de la base de données à utiliser.

L'exemple du Listing 1 donne comment configurer une base de données locale en Mysql. Cette exemple pose un problème, car il consiste à écrire, en dur, dans le code, vos données d'usage vers votre base de données. Ceci est une très mauvaise habitude surtout si vos environnement de développement, de test et de production ont des bases différentes..

J'en vois déjà quelques-uns qui se disent mais pourquoi ne pas utiliser le fichier *Application.ini* ? Cela est possible et il faut utiliser la *Factory* de la classe `Zend_Db`. Le Listing 2 donne le code à utiliser dans *application.ini*.

Le gros avantage de cette pratique est que vous pourrez facilement modifier les valeur de connexion environnent par environnement. Rappelez vous que dans le précédent article, je vous avais décrit la structure du fichier *application.ini* avec une zone *[production]*, une zone *[developpement]*, … toute zone dont vous avez besoin. Dans le cas présent, cela nous permet d'avoir une base données locale sous localhost pour le dev, une avec une adresse IP différente pour la production et cela sans modifier le moindre fichier de code source. Il suffit de modifier la valeur de la variable d'environnement d'exécution.

Bonne habitude n°1 : Toute les valeurs constantes doivent être mises dans des constantes centralisées dans un fichier dédié. Ceci est une bonne pratique de développement, qui n'a aucun lien avec PHP ou le Framework Zend. Si on la pousse au maximum, cela revient à mettre tous les textes utilisés dans notre site dans un fichier dédié. Cela nous facilitera l'internationalisation ultérieure.

Bonne habitude n°2 : utiliser les fonctions de gestion des fichiers de configuration de Zend. La Zend_config permet d'utiliser des fichier *.ini* ou *.xml* pour gérer les constantes du projet. Le fichier *Application.ini* (vu dans notre précédent article) utilise ces mécanismes mais de façon masqué pour le développeur. J'aime cette classe `Zend _ config` et je vous invite à l'utiliser. En utilisant des fichiers *.ini* ou *.xml*, selon vos habitudes. J'utilise les deux. Les fichiers en *.xml* pour les constantes propres à l'application, car le langage xml offre plus de possibilité de structuration que le *.ini*. Le *.ini* pour la configuration du framework car, a l'inverse, utiliser le xml pour cela c'est un peu lourd.

Le code du *boostrap* est améliorable. Le *try catch* est améliorable et on verra cela plus tard, mais surtout on peut s'interroger sur l'usage de `Zend_registry`.

L'objet `Zend_registry` est un singleton (c'est à dire qu'il n'existe qu'un seul objet – une seule instance – de ce type) qui permet d'enregistrer des données partagées ensuite par tous les fichiers php traversés pendant l'appel en cours (associé à une URL ou une session *http*). Ce n'est pas une mise en cache et ne fait que limiter à un, le nombre de connexion ouverte lors de l'appel d'une URL. C'est très utile, même pour ne pas avoir à recréer des objets métiers. Ainsi, un petit test sur la présence d'un objet dans le registry (*Zend_Registry::isRegistered('nomdedonnée')*) avant sa création (son instanciation) évite une surconsommation de la mémoire.

Mais dans notre cas, cela ne servira à rien puisque dans notre fichier de configuration nous avons défini que db serait notre adapter par défaut avec `resources.db.isDefaultTableAdapter = true`. La question suivante à aborder est : comment accéder aux données de la table. L'approche du framework est assez simple, peut être même simpliste, car on va créer un objet par table. Là, deux possibilités s'offrent à nous : utiliser les outils de *Zend_tools* ou écrire les classes à la main.

Avec l'outil zf, dont nous avons parlé le mois dernier, il est possible de créer automatiquement les classes `Zend_Db_Table` associées aux tables de notre base de données. Pour cela nous utilisons les fonctions de l'univers dbtable avec *zf create dbtable*. Mais il faut avant tout donner à zf les moyens de se connecter à la base de données en lui donnant les mêmes informations que celles déjà placées dans le fichier *application.ini* … Attention, il n'y a pas de lien entre votre application et l'outil zf. Le Listing 3 donne l'ensemble des lignes de commande dont nous allons parler.

La première, configure la connexion de l'outil zf à la base de données, alors que la seconde, permet de créer la classe `Users` qui sera associée à la table user de votre base de données. Les fichiers ainsi créés seront rangés automatiquement dans le répertoire *application/models/* de votre projet.

Un tel travail peut être fastidieux si vous avez de nombreuses tables dans votre schéma de base, l'outil zf permet de créer entièrement une table en utilisant *dbtable.from-database*. Je vous conseille d'utiliser l'option *-p* qui vous donnera plus de visibilité pendant la phase de création des classes.

Une fois les classes du modèle créées, leur usage est très simple, du moins tant que l'on reste dans des usages basiques. J'avoue ne pas avoir eu à l'utiliser récemment et que j'ai plus utilisé la méthode manuelle pour créer mes classes de connexion. Mais je reste convaincu que zf est la méthode la plus rapide.

Le connecteur est directement utilisable dans vos contrôleurs à l'aide d'un simple new. Comme le montrent les dernières lignes du Listing 3, la création d'un

Listing 1. *Connecter une base de données Mysql avec Zend Framework*

```
    $db = new Zend _ Db _ Adapter _ Pdo _ Mysql(
HYPERLINK "http://www.php.net/array"array(
    'host'=> '127.0.0.1', // ou 'localhost'
    'username' => 'userlogin',
          'password' => 'BDDPassword',
          'dbname'=> 'test'
));
```

Listing 2. *Connexion propre d'une base de données*

```
    // inclure dans application.ini dans
l'environnement d'usage
        # Database
resources.db.adapter = "pdo _ mysql"
resources.db.params.host = "localhost"
resources.db.params.username = "userlogin"
resources.db.params.password = "BDDPassword"
resources.db.params.dbname = "test"
resources.db.isDefaultTableAdapter = true

    //Dans le bootstrap il faut ajouter la fonction
d'init suivante
      function _ initDb() {
  $ressource = $bootstrap->getPluginResource ('db');
  try {
$db = $ressource->getDbAdapter();
$db->getConnectipon();
  } catch (Exception $e) {
 exit ($e->getMessage());
  }
  Zend _ registry::set ('db',$db);
}
```

Listing 3. *Lignes de commande pour créer les tables de votre application*

```
  // ligne de commande pour connecter zf et la base
de données
  zf configure dbadapter "adapter=Pdo _ Mysql&username
=userllogin&password=BDDPassword&dbname=test"

  //ligne de commande pour créer la classe Users
associée à la table user
  zf create dbtable User user

  //lignes de commande pour créer les classes
associées à un modèle de base complet
  zf create dbtable.from-database

  // ou
  zf -p create dbtable.from-database

  // creation des objets d'accès à une table

  $users = new Users ($db);
  // ou si $db est défini comme adapter par défaut

  $users = new Users();

  // recherche par clé primaire

  $result = $users->find(ID);
```

Conseil pour programmer efficacememnt ou plus proprement (dans le sens j'ai pas des tonnes de fichiers à changer) : Il est préférable d'éviter d'utiliser les objets d'accès aux tables directement dans les contrôleurs et de passer par des objets de mapping qui vont venir abstraire la structure de la table. Pourquoi vous demandez-vous ? Tout simplement pour éviter qu'à chaque modification de la structure de la base, vous soyez obligé de modifier le code de toutes vos actions.

Listing 4. *Bootstrap de Doctrine*

```
  /*
   * Configuration de Doctrine dans l'application.ini
   */
  doctrine.dsn = " HYPERLINK "mysql://root"mysql://
userlogint:BDDPassword@localhost/user"
  doctrine.data _ fixtures _ path = APPLICATION _ PATH "/
doctrine/data/fixtures"
  doctrine.sql _ path = APPLICATION _ PATH "/doctrine/
data/sql"
  doctrine.migrations _ path = APPLICATION _ PATH "/
doctrine/migrations"
  doctrine.yaml _ schema _ path = APPLICATION _ PATH "/
doctrine/schema/schema.yml"
  doctrine.models _ path = APPLICATION _ PATH "models"

function _ initDoctrine(){
  //Launch Doctrine

  $dbManager = Doctrine _ Manager::getInstance (); //
permet de définir des attributs de Doctrine

  // on récupère les options

  $doctrineConfig = $this->getOption('doctrine');

  // On lance la connexion

  $db = Doctrine _ Manager::connection
($doctrineConfig['dsn']);

  return $db;

 }

// On récupère les utilisateurs de la table user

$users = Doctrine _ Query::create()
         ->select ('*')
         ->from ('user');

// ou pour faire comme dans Zend _ Db _ Table

$userTable = Doctrine _ Core::getTable('user');
$user = $userTable->find($userId);
$users  = $userTable->fetchAll();
```

Listing 5. *Exemple d'usage de Redirector*

```
// initialisation du Helper dans le contexte d'une
classe controller

  $this->redirector = $this-> _ helper->getHelper('Red
irector');

// utilisation de ce redirector dans une action

  $this->redirector->gotoSimple($url)

// exemple d'initialisation plus complexe

class MonController extends Zend _ Controller _ Action
{
  /**
  * redirector contient le helper de redirection
d'URL, de module, de controller ou d'action
  * @var Zend _ Controller _ Action _ Helper _ Redirector
  */
  protected $ _ redirector = null;

  function _ init () {
    $this-> _ redirector = $this-> _ helper->getHelper(
'Redirector');
  }

  function MonAction () {
    $this-> _ redirector->gotoSimple('mon _
action','mon _ controler','mon _ module',
array('param'=>'data'));
  }
}
```

objet d'accès à la table est très simple (surtout si on définit un adapter par défaut comme c'est notre cas). Nous disposons maintenant de toutes les fonctions nécessaires pour faire des recherches *(find(), fetchAll(), fetchRow()* ou *select())* ou des enregistrements (*save($user), insert(), update(), delelet()*).

Dans la plupart des cas d'usage, l'objet `Zend_Db_table` suffira, mais si vous avez des bases très complexes ou vous devez vous interfacer avec des bases de données relationnelles comme Oracle, il se peut que les fonctions proposées soient trop limitées et que vous ayez à utiliser une interface plus évoluée comme Propel ou Doctrine. Doctrine étant le plugin qui offre le plus de fonctions sur ce registre, nous allons voir comment l'installer et l'utiliser avec ZF.

Utiliser Doctrine

La première chose à faire est d'ajouter Doctrine dans notre environnement de travail. Le meilleur endroit pour le faire, vous l'aurez sans doute deviné … Allez, pour les plus curieux ou les plus assidus, essayez de trouver par vous même avant de lire la suite … relisez l'article de la semaine dernière si vous voulez … Non … vous ne trouvez pas ou vous pensez avoir trouvé et voulez avoir la solution ? … c'est très simple ! Il faut mettre le code php de Doctrine (ce serait pareil pour Propel) dans le répertoire */library/Doctrine* de votre projet ! *C'est so simple* comme dirait le chef Chaudard.

La première opération va consister à ajouter le répertoire de Doctrine dans notre include *path* et notre *Autoloder*.

Pour ce faire on ajoute dans le boostrap et dans la fonction d'init de l'autoloader :`$autoloader->registerNamespace('Doctrine_')`. Ce qui nous permettra d'accéder aux classes de Doctrine qui débutent par *Doctrine_* et sont rangées dans le répertoire */library/Doctrine*. Il nous faut maintenant, lancer Doctrine lui même.

L'enregistrement dans *Zend Registtry* de la configuration de la structure de Doctrine sous entend que vous ayez créé la structure de fichier dans application comme montré dans la Figure 1. Par la suite, l'usage de Doctrine dans votre application est assez simple. A la différence de *Zend_Db*, où il faut construire un objet par table, nous allons devoir considérer Doctrine comme un singleton et accéder à ses fonctionnalités via ses méthodes en mode statique avec l'opérateur *'::'*. Ainsi, pour faire une requête sur notre table *user*, il faut utiliser la méthode `Doctrine_Query::create()`. Mon objectif n'étant pas de vous faire un cours pour utiliser Doctrine mais comment vous pouvez l'interfacer avec Zend Framework, je me contenterai de vous donner deux exemples d'usage dans le Listing 4.

On peut aller beaucoup plus loin (notamment avec l'usage de Doctrine en ligne de commande pour générer des modèles – c'est encore plus puissant que zf)

et je vous y invite car Zend a décidé fin 2009, d'intégrer Doctrine à son Framework plutôt que de développer les fonctionnalités absentes du Framework. Donc cela deviendra un passage obligé à plus ou moins court terme !

Cela tombe bien car Doctrine est l'ORM (*Object Related Manager*) le plus complet en PHP du moment.

Les Helpers

J'ai déjà évoqué certains helpers dans mon précédent article, quand je vous présentais l'usage du JSON. Je voudrais aller un peu plus loin maintenant et vous présenter les Helpers ZF dans le détail. Ils sont très utiles et sont répartis en deux groupes :

- action Helpers,
- view helpers.

On va les étudier un à un dans les deux prochains paragraphes. Je ferai un focus plus important sur ceux qui me semble indispensables pour débuter. Ce qui fera que la partie helper de vue sera plus fournie que la partie action.

Ce qui est notable dans les Helpers c'est qu'il sont utilisables selon le contexte mais initialisables (ou chargeables) à tout moment. Dans le bootstrap, dans les init des controllers ou au moment même de leur utilisation. C'est pour cette raison que l'on dit qu'ils peuvent être appelés à la demande.

Les actions Helpers

La description complète de ces helpers est disponible dans la documentation de la librairie *Zend_controller*. Un rapide survol de ces Helpers nous donne :

- *Zend_Controller_Action_HelperBroker* qui gère (charge et ajoute) les Helpers personnalisés ou non.
- *ActionStack* qui comme sont nom l'indique permet d'empiler des actions à réaliser. Ainsi si une action consiste dans la réalisation de plusieurs que vous avez déjà codées et testées pas besoin de faire du copier coller vous définissez la liste d'action à réaliser avant votre rendu
- *AjaxContext* (que nous avons vu dans le précédent article) et qui comme *ContextSwitch* permet de définir des formats de réponse particulier à Ajax ou un context défini dans l'url.
- *AutoComplete* qui facilite l'autocompletion en mode Ajax avec des frameworks Javascript comme Dojo ou scripta.coulo.us. Si comme moi vous préférez Jquery alors il vous faudra utiliser les aides du projet *ZendX_JQuery.*
- *ContextSwitch* qui vous permet donc de définir des contexts de vue prédéfinis ou de forcer celui de votre réponse.

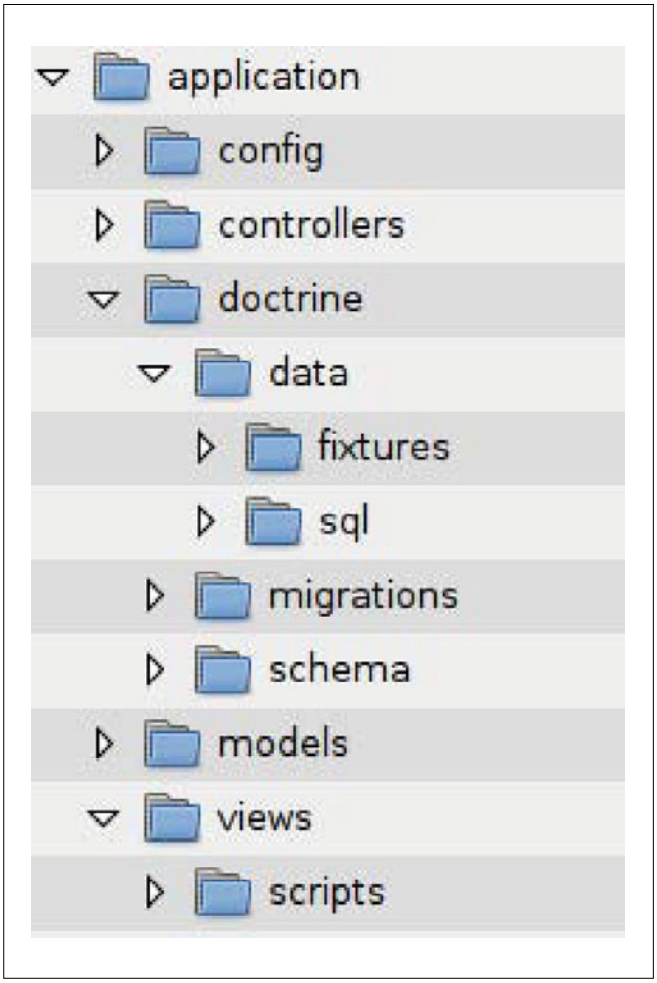


Figure 1. *Ajout de doctrine dans votre arborescence projet*

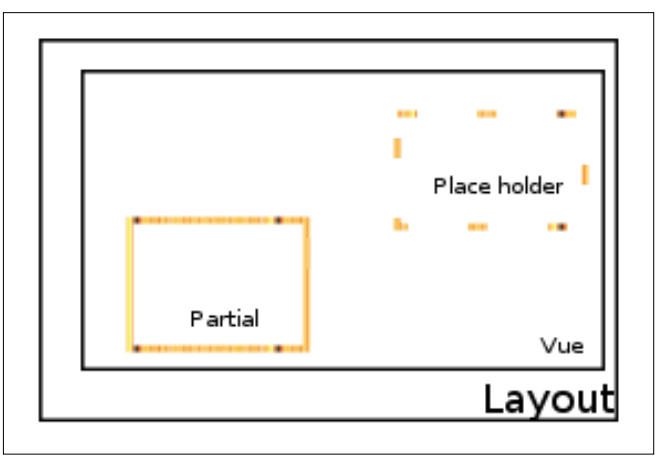


Figure 2. *Exemple de structure de vue*

- *FlashMessenger*, n'a rien à voir avec le Flash mais permet d'échanger via la session des messages entre requêtes (URL) ou redirections successives.
- JSON celui-ci aussi vous le connaissez déjà.
- Redirector qui vous permet de rediriger vers un autre couple *Controller/action* de façon logiciel (sans changer l'URL initiale).
- *ViewRender* permet de définir des modes de rendu. Ainsi si vous n'avez pas différents modules

Listing 6. *Exemple de Partials tirés de la doc Zend*

```
<!--partialLoop.phtml-->
    <dt><?php  HYPERLINK "http://www.php.net/echo"echo
$this->key ?></dt>
    <dd><?php  HYPERLINK "http://www.php.net/echo"echo
$this->value ?></dd>

$model =  HYPERLINK "http://www.php.net/array"array(
     HYPERLINK "http://www.php.net/array"array('key' =>
'Mammifère', 'value' => 'Chameau'),
     HYPERLINK "http://www.php.net/array"array('key' =>
'Oiseau',    'value' => 'Pingouin'),
     HYPERLINK "http://www.php.net/array"array('key' =>
'Reptile',   'value' => 'Asp'),
     HYPERLINK "http://www.php.net/array"array('key' =>
'Poisson',   'value' => 'Flounder')
);

<dl>
<?php  HYPERLINK "http://www.php.net/echo"echo $this->
partialLoop('partialLoop.phtml', $model) ?>
</dl>

<dl>
    <dt>Mammifère</dt>
    <dd>Chameau</dd>

    <dt>Oiseau</dt>
    <dd>Pingouin</dd>

    <dt>Reptile</dt>
    <dd>Asp</dd>

    <dt>Poisson</dt>
    <dd>Flounder</dd>

</dl>
```

Listing 7. *Les PlaceHolders*

```
// Dans l'action

<?php $this->placeholder('MaDonnée')->set('LaValeurDeM
aDonnée'); ?>

// Dans la vue

<?php echo $this->placeholder('MaDonnée'); ?>

//Meta de notre Vue avec un exemple pour gérer le
chargement de vos css
//issu de la doc Zend.
<?php // régler les liens dans votre script de vue :
$this->headLink()->appendStylesheet('/styles/basic.
css')
                          ->headLink( HYPERLINK
"http://www.php.net/array"array('rel' => 'favicon',
                                                'href' =>
'/img/favicon.ico',
                                                'PREPEND')
                          ->prependStylesheet('/styles/
moz.css','screen',true,
                                                array('id'
=> 'my _ stylesheet'));
?>

<!-- effectuer le rendu --!>
<?php  HYPERLINK "http://www.php.net/echo"echo $this-
>headLink() ?>
```

associés à différents terminaux de rendu (Web, mobile, Ipad, …) vous pouvez quand même définir des règles de rendu différents associés à des extensions particulières.

- Vos propres Helpers d'action que vous pourrez créer, en respectant quelques règles, et manager avec le *Broker Zend_controller_Action_HelperBroker*.

Mon expérience fait que les Helpers de context et *Redirector* sont ceux que l'on utilise le plus vite. Nous avons vu l'usage des premiers dans le précédent article, alors voyons maintenant Redirector.

Le Listing 5 donne un exemple simple d'usage du *Helper Redirector*.

Pour éviter d'avoir à l'initialiser (le charger) dans chaque action, il est préférable d'utiliser la fonction `_init()` de votre classe controler comme dans la troisième partie de l'exemple du Listing 5. Pour cela, il est plus sage de créer une variable de type protected voire même private pour limiter les risques d'usage externe et donc les attaques de sécurité ! Une variable publique pourrait être utilisée par un Hacker pour vous rediriger chez lui sans que vous le sachiez !! Enfin, un bon *Hacker* … voire même un très bon.

Notez qu'il n'y a pas de return à la fin de l'action car par défaut, l'usage de *GotoSimple* entraine un exit de la fonction courante. Cela peut être paramétré, si vous le souhaitez. Pour cela il faut utiliser la méthode `setExit()`. Vous pouvez aussi changer le code de retour HTTP avec `setCode()` ...

Les view Helpers

Le nombre de helpers de vue est très grand, car ils couvrent de nombreux sujets. Ceux que je souhaite mettre en avant concernent la structure de notre vue. Pour cela, prenons la Figure 2 qui décrit une structure de vue possible.

On y voit qu'une vue est construite à partir d'un Layout et du template de vue dans lequel (associé au couple control/action) on peut utiliser des helpers pour y ajouter des morceaux de code définis dans d'autre fichiers templates spécifiques. C'est une structure assez standard, que vous connaissiez *Symphony* ou *Smarty* ces principes existent plus ou moins. Mais qu'est-ce qu'un *Partial* et qu'est-ce qu'un *PlaceHolder* ?

Le *Partial* est une portion de code constante qui peut être réutilisée de vue en vue alors que le placeHolder est une portion de code dont les données peuvent persister depuis les scripts d'action jusqu'aux instances de vue permettant de les manipuler alors … Intéressant non. Il existe un troisième type de Helper de vue très pratique c'est celui (ou ceux) associé à la classe de navigation `Zend_Navigation_Container` qui permettent notamment de faire des menus ou des fils d'*Ariane*.

Les Partials

Voyons plus en détail les partials. Je vous disais, ci-dessus, qu'ils permettent de réutiliser du code HTML et ou Javascript qui se répète. Par exemple, vous devez afficher le même genre d'information de page en page, qu'elle soit statique ou un peu dynamique, alors le partial vous permet de faire cela facilement.

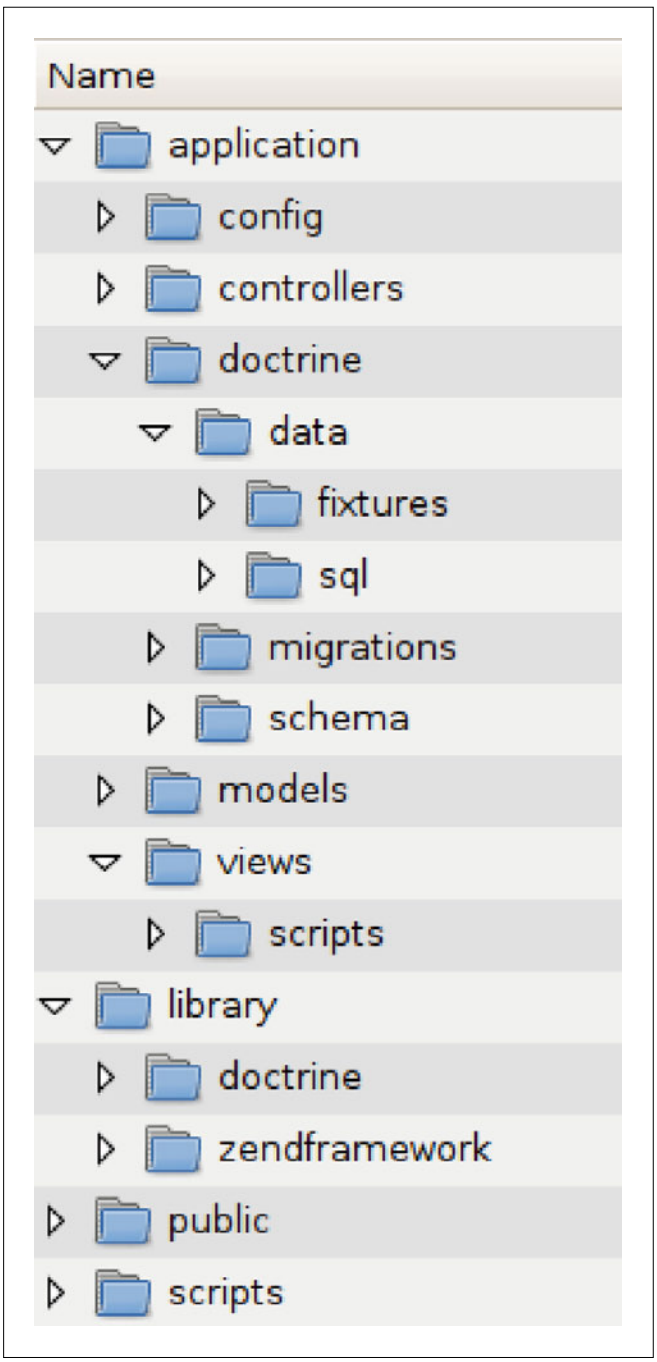


Figure 3. *Structure d'une vue*

Il existe deux types de partial appelez par deux helpers différents :

- *$this->partial (partial.html, array('param'=>'data'))*
- *$this->partialLoop (partial.html,$dataList)* où *dataList* est un tableau contenant plusieurs couples clé/valeur sur lesquels appliquer le contenu de *partial.html*.

Les données sont à passées par des tableaux ou des objets mais à condition que celui-ci dispose d'une fonction de type `ToArray()`. Sans cette fonction, toutes les variables membres récupérées par la fonction PHP `object_get_vars()` seront passées au partial.

Sur Internet

- Télécharger Doctrine – *http://www.doctrine-project.org/projects/orm/download.*
- Télécharger la documentation du Framework Zend : *http://zendframework.com/download/latest.*
- Forum Français autour de Zend Framework : *http://www.z-f.fr/.*

Les PlacesHolders

Les comprendre n'est pas aussi trivial qu'il y paraît, car le contexte influe sur leur rôle. L'usage n°1 permet, dans vos scripts d'action, de placer des valeurs qui seront utilisées plus tard par la vue ou d'autres scripts d'action, comme le montre la première partie du Listing 6. On peut tout aussi bien passer un tableau de données avec `exchangeArray($array)`. Tableau que l'on pourra toujours utiliser dans la vue mais aussi dans une autre action. Permettant ainsi d'avoir un réceptacle où ranger toutes les données (des textes par exemples) utilisables par une vue. On peut même, pour les blocs de textes ou de données, utiliser *captureStart* et *captureEnd* pour placer le contenu HTML placé entre ces deux méthodes dans une variable de type *placeHolders* que nous pourrons utiliser dans d'autres morceaux de vue.

Cette technologie sert de base à des *placeHolders* dédiés qui vont gérés les meta de la page ou les scripts comme le montre la seconde partie du Listing 6 qui donne un exemple avec le meta link placé dans le header de la page.

Conclusion

Voici le second volet de la série. Dans le prochain numéro, je me pencherai notamment sur la gestion du cache, les tests unitaires et le débugage d'application sous Zend. A moins que vous ne préféreriez un approfondissement de l'accès aux données d'une base.

STÉPHANE GUÉDON

L'auteur travaille au sein d'une société de développement d'applications informatiques depuis quinze années. Aujourd'hui, consultant pour une grande société de service d'un groupe international, il anime le projet PHP de la Direction Technique Nationale de cette société et participe aux projets WEB 2.0 de cette même direction. Adepte du PHP depuis ... l'ère des dinosaures, il œuvre à la communication et à la promotion du PHP au sein de cette entreprise.
En complément de ces activités, l'auteur est également Coach Agile et assiste les entreprises dans le déploiement des méthodes de développement Agile.

CakePHP
le framework pour développer rapidement vos applications

Démarré en 2005 lorsque Ruby On Rails devenait populaire, le projet CakePHP compte désormais de nombreux utilisateurs et une communauté toujours plus active. Désormais en version 1.3, le framework se veut avant tout simple et efficace, en se basant sur le paradigme des conventions plutôt que de la configuration.

Cet article explique :
- Les principales conventions utilisées par CakePHP.
- La génération automatique des actions de base (CRUD) d'une application.
- Les opérations usuelles sur les modèles et leurs relations.
- Les principaux composants livrés avec le framework … pour se simplifier la vie dans les actions courantes (modélisation, formulaires, authentification, gestion des droits, i18n).

Ce qu'il faut savoir :
- Connaissance du langage PHP 4+.
- Notions de l'architecture MVC.

CakePHP a été conçu afin de faciliter au maximum la vie du développeur qui l'utilise. Cependant il faut au préalable comprendre la philosophie du framework et la suivre au départ de manière presque aveugle. Ce n'est qu'une fois les principes clés assimilés et les applications de base réalisées avec succès que l'on pourra se rendre compte de la réelle valeur ajoutée de CakePHP et de tous ses composants. Voici donc les points qui vous permettront de débuter sans encombre.

Des conventions plutôt que de la configuration

Un des principaux atouts de CakePHP contribuant à la mise en place rapide et à la magie du framework est son fonctionnement par conventions. En effet, le framework ne nécessite que très peu de configuration comme nous le verrons plus tard et se base sur une découverte intelligente de l'application d'après certaines conventions. Les conventions permettent également de sous-traiter certaines tâches classiques et rébarbatives (intégrité des relations, récupération et associations des modèles, mise à jour de champs …) à CakePHP qui est livré avec un bon nombre de fonctionnalités utiles.

Avantages des conventions

En tant que développeur, chacun a pu se créer ses propres règles de codage et façons de travailler au fur et à mesure de ses expériences. Il est ainsi vu comme difficile (ou d'un intérêt limité) de changer sa manière de coder afin de se plier à une technologie (en l'occurrence CakePHP), mais nous espérons que ce paragraphe et toutes les belles choses décrites ci-après vous feront tenter l'expérience : vous avez beaucoup à y gagner !

Les conventions sont tout d'abord le meilleur moyen de pérenniser un code. En effet, à l'heure actuelle le code d'une application est amené à être maintenu, réutilisé ou partagé avec d'autres développeurs. D'autre part, la lecture d'un code développé par une autre personne peut s'avérer initialement difficile car la manière de développer (le style, l'organisation des fichiers …) est rarement documentée. C'est là tout l'intérêt des conventions ! Elles permettent d'avoir une base commune et d'éviter de sur-documenter une application afin de se focaliser sur la valeur ajoutée et les parties importantes du code. À l'échelle d'une entreprise, ces conventions de codage et de nommage sont bien souvent en place en interne … avec CakePHP cependant, *vos* conventions seront désormais compréhensibles par tous !

Enfin, le respect des conventions permet une réduction non négligeable du code produit ce qui augmente la lisibilité de celui-ci. Finis les fichiers de configuration à rallonge, les bibliothèques de fonctions *pratiques* qui passent d'un projet à un autre, les bugs issus de méthodes de ces bibliothèques … CakePHP est livré avec

de nombreuses fonctionnalités pratiques, grandement testées et bien documentées !

Principales conventions

Nous détaillerons ci-après la plupart des conventions à suivre dans un projet CakePHP, dans un but de compréhension et d'apprentissage. Cette liste n'est cependant pas exhaustive, nous vous invitons à lire la documentation ou les articles sur le web afin de connaître tous les détails.

Base de données et champs automagiques

La modélisation de la base de données est l'élément primordial d'un projet. Si les conventions sont respectées, il sera ensuite possible de générer la plus grande partie du code (CRUD) associé en quelques minutes ! Nous détaillerons ceci plus loin.

Concernant les conventions, elles sont les suivantes :

- le nom de la table doit être en minuscules et au pluriel (exemple : `my _ messages`),
- un champ `id` devra être présent et pourra être de deux types : entier auto-incrémentable ou une chaîne de 36 caractères (`CHAR(36)`) auquel cas l'identifiant généré sera un UUID,
- une clé étrangère devra porter le nom du modèle référencé au singulier suivi de `_ id` (exemple : `user _ id`).

D'autre part on trouve des champs *automagiques*. On les nomme de cette manière car ces champs seront renseignés automatiquement par Cake lors de la sauvegarde d'informations (ils ne sont en aucun cas obligatoires !).

- le champ `created` de type `DATETIME` contiendra la date de création de l'élément,
- le champ `modified` ou `updated` contiendra la date de dernière modification de celui-ci,
- le champ `title` ou `name` sera utilisé comme intitulé de l'élément lors de son affichage sur le site. Dans le cas où aucun de ces champs n'existe, c'est l'identifiant qui sera utilisé,

Un dernier exemple un peu plus poussé : si un utilisateur est associé à plusieurs messages (c'est-à- dire que la table `messages` contient un champ `user_id`), il suffit d'ajouter un champ de type entier (`INT`) nommé `message_count` dans la table `users` afin que celui-ci soit automatiquement incrémenté et décrémenté … et contienne à tout instant le nombre de messages associés à l'utilisateur.

Modèles et contrôleurs : Nommage des classes

Afin d'utiliser votre base de données, vous aurez besoin d'une classe de modèle qui gérera tous les aspects liés à votre modèle (validation, associations, actions métier …) et une classe contrôleur chargée de gérer les évènements et d'appeler les méthodes adéquates du modèle pour transmettre les informations utiles à la vue. Renseignez-vous sur le modèle MVC pour mieux comprendre ces concepts !

Pour CakePHP les classes de modèle doivent porter le nom de la table associée au modèle, au singulier et `CamelCased` (c'est-à-dire que l'on supprime les *underscores* du nom de fichier et on met la première lettre de chaque mot en majuscule … *comme les bosses d'un chameau !).* De plus, les modèles doivent hériter de la classe `AppModel`. De même, les contrôleurs doivent se nommer comme les modèles mais au pluriel (également `CamelCased`), suivis du mot `Controller` et doivent hériter de la classe `AppController`. Par exemple : la table `my_messages` aura pour modèle la classe `MyMessage extends AppModel` et le contrôleur sera `MyMessagesController extends AppController`.

Organisation des fichiers

Étant dans un framework MVC, les fichiers sont répartis dans trois principaux dossiers à la racine de l'application : */models*, *views* et *controllers* ! Un dossier */webroot* contiendra toutes les ressources utilisées (javascript, css …) et tiendra lieu de racine de documents. Cette structure est également valable pour un plugin, pour lequel les dossiers seront regroupés dans le répertoire */plugins/nomduplugin* de l'application. Les noms de fichiers doivent être en minuscule et les mots séparés par des *underscores* («_»). Le nom du fichier correspond en général à la classe qu'il contient (modèle ou contrôleur), ou à la vue à laquelle il est associé.

Chargement automatique de fichiers

Ainsi, rien qu'en suivant les conventions précédentes et sans ligne de code supplémentaire votre application est prête à être exécutée ! (Note : en réalité il faudra simplement relier votre application au cœur de CakePHP et configurer la connexion à la base de données dans un fichier *database.php*). Voyons ce qui se passera à l'appel d'une page :

- appel de l'URL *http://www.helloworld.com/my_messages/hello*,
- exécution de la méthode `hello()` de la classe `MyMessagesController` située dans le fichier `/controllers/my _ messages _ controller.php`,
- le contrôleur fera sans doute appel à une méthode du modèle `MyMessage` située dans le fichier `/models/my _ message.php` qu'il connaîtra automatiquement,
- le modèle interagira avec la table `my _ messages` de la base de données et renverra des données au contrôleur,

De base, l'anglais est préférable

CakePHP est livré avec des mécanismes de reconnaissance singulier / pluriel optimisés pour la langue anglaise (exemple : *people* / *person*). Ainsi, afin de ne pas avoir de mauvaise surprise il est recommandé d'utiliser des mots anglais. (C'est également une bonne chose pour le partage de votre code !) Bien que la plupart des mots français soient gérés (exemple : *personnes* / *personne*), des cas propres à notre langue ne pourront être reconnus (exemple : *choux* / *chou* ou *bases_de_donnees* / *base_de_donnees*) dans la version actuelle du framework. Par un système de configuration d'inflexions il est néanmoins possible de tenir compte de ces cas particuliers et la communauté française travaille actuellement en ce sens afin de proposer une version francisée du framework.

Listing 1. *Création de la base de données*

```
CREATE TABLE 'users' (
        'id' INT UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY ,
        'name' VARCHAR( 100 ) NOT NULL ,
        'email' VARCHAR( 100 ) NOT NULL
) ;

CREATE TABLE 'messages' (
        'id' INT UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY ,
        'title' VARCHAR( 150 ) NOT NULL ,
        'content' TEXT NOT NULL ,
        'user_id' INT UNSIGNED NOT NULL ,
        'created' DATETIME NOT NULL ,
        'modified' DATETIME NOT NULL
) ;
```

Listing 2. *Création du modèle user dans le fichier /models/user.php*

```
<?php
class User extends AppModel {
        var $name = 'User';
}
?>
```

Listing 3. *Création du modèle «message» dans le fichier /models/message.php*

```
<?php
class Message extends AppModel {
        var $name = 'Message';
        var $belongsTo = 'User';
}
?>
```

Listing 4. *Création du contrôleur «users» dans le fichier /controllers/users_controller.php*

```
<?php
class UsersController extends AppController {
        var $scaffold; // Cette ligne permet de
passer en mode prototypage rapide pour ce contrôleur
}
?>
```

Listing 5. *Création du contrôleur «messages» dans le fichier « /controllers/messages_controller.php »*

```
<?php
class MessagesController extends AppController {
        var $scaffold;
}
?>
```

- enfin le contrôleur transmettra ces informations à la vue pour un affichage de la page, ce qui affichera le contenu du fichier `/views/my _ messages/hello.ctp`.

Nous voyons par ce simple exemple qu'il est ainsi possible de séparer clairement chaque partie de l'application. Ceci rend plus simple le travail collaboratif, la rédaction de tests unitaires et la recherche d'une ligne de code : suivant ce que l'on cherche on sait exactement dans quel fichier chercher. Note avancée : à chaque étape de ce processus il est possible d'intercaler des actions grâce à l'existence de méthodes de *callback,* pour une meilleure souplesse.

Si vous ne voulez pas des conventions

Il faut cependant noter que le suivi des conventions n'est pas obligatoire (de la même manière qu'une personne peut construire sa maison toute seule !). Les irréductibles ou ceux qui n'ont pas la main sur les conventions de nommage utilisées, dans le cas de la reprise d'une application existante par exemple, trouveront donc une solution à leur situation. Nous ne pouvons tout détailler ici, mais sachez qu'il est possible de modifier le comportement du framework et de se servir de sa puissance sur un code ne respectant pas les conventions … il suffira simplement de rajouter un peu de configuration à votre application !

Le scaffolding ou prototypage rapide

CakePHP est avant tout un outil facilitant la mise en place rapide d'une application (framework de RAD). Comme nous l'avons vu précédemment, une bonne modélisation de la base de données respectant certaines conventions peut vous faire gagner un temps précieux. Il est désormais temps de voir tout ce que CakePHP vous propose !

Le prototypage rapide est une fonctionnalité permettant de vérifier la cohérence de votre base de données et de vous fournir des formulaires permettant toutes les actions CRUD (ajout, visualisation, mise à jour et suppression) d'un objet et de ceux qui y sont associés. Prenons l'exemple d'une base de données gérant des utilisateurs et leurs messages de manière simpliste. Nous devrons donc créer deux tables telles que décrit dans le Listing 1.

Le plus difficile est effectué, analysons désormais les modèles associés dans lesquels nous indiquons qu'un message appartient à un utilisateur. Cette étape est présentée dans les Listing 2 et 3, notez au passage la simplicité de définition des associations (Listing 3). Enfin demandons à Cake de faire le travail de prototypage rapide, ceci depuis les contrôleurs associés aux *Utilisateurs* (Listing 4) et aux *Messages (*Listing 5).

C'est tout ! Désormais, en vous rendant sur les pages *http://www.votresite.com/users* et *http://www.votresite.com/messages* vous pourrez tout gérer. Les formulaires permettant les actions CRUD sont disponibles.

La Figure 1 est une capture d'écran du formulaire d'ajout d'un nouveau message. Nous voyons que la liste des utilisateurs existants (un utilisateur a été créé auparavant) s'affiche, afin de relier le message à son auteur. La Figure 2 quant à elle montre le résultat de la création : le lien avec l'utilisateur est bien sauvegardé et les champs automatiques ont été renseignés. Magique n'est-ce pas ?

CakePHP en ligne de commande

CakePHP offre une interface en ligne de commande permettant de réaliser certaines tâches très rapidement, comme la création du squelette d'application, la génération interactive des fichiers pour chaque entité (modèle, contrôleur et vues) ou encore l'extraction des chaînes gettext présentes dans tous les fichiers de l'application dans le cadre d'une application multilingue. Il est également possible de créer ses propres scripts exécutables via cette console, les *shells* et les *tasks*.

Squelette d'application

Après avoir ajouté l'exécutable php dans les variables d'environnement (sous Windows) et créé une nouvelle base de données, nous pouvons ouvrir un terminal (*Démarrer* > *Exécuter* > *cmd.exe*) et appeler la console : `cake bake`.

Le script nous propose de créer une nouvelle application : il suffit d'indiquer le répertoire de destination et la console crée toute l'arborescence de base, et génère une nouvelle clé de sécurité dans le fichier *config/core.php* qui servira lors de la mise en place un système d'authentification.

Le script nous demande ensuite les codes d'accès à la base de données, puis se termine. Nous pouvons nous rendre à la racine de l'application dans un navigateur, et nous voyons apparaître la page d'accueil de notre nouvelle application.

Création d'un modèle

Contrairement au scaffolding, qui ne crée pas les fichiers nécessaires dans l'application, la console permet de générer effectivement ces fichiers de manière interactive. Appelons à nouveau la console, de la même manière que plus haut : `cake bake`.

Cette fois la console a détecté que notre application existe, et nous propose une liste d'actions possibles : ajouter une configuration de base de données, créer un modèle, une vue, un contrôleur ou un nouveau projet. Nous choisissons de créer un modèle : la console propose la liste des tables de la base de données. Nous choisissons l'une des tables : la console demande si nous voulons définir des règles de validation pour les champs de cette table. Nous répondons oui : pour chaque champ, nous pouvons choisir parmi 28 règles prédéfinies (*notEmpty, alphaNumeric, email*, etc.), écrire notre propre expression régulière, ou bien ne pas définir de règle de validation.

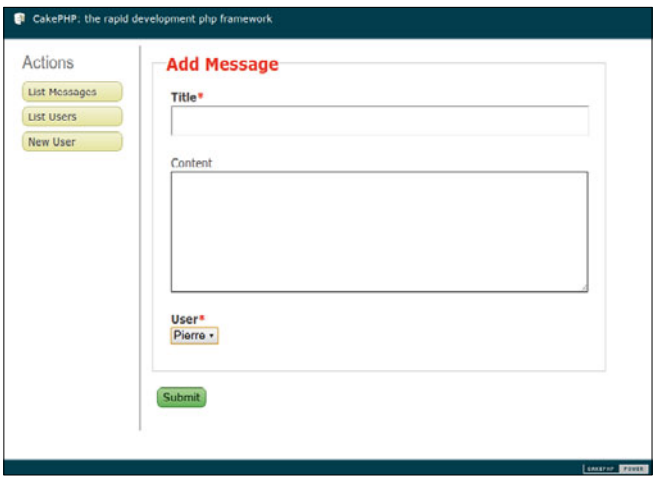


Figure 1. *Formulaire de création de message généré par CakePHP*

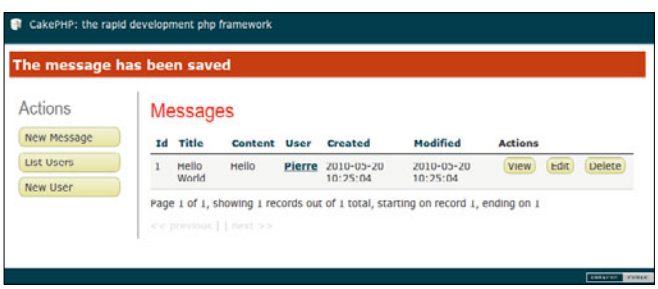


Figure 2. *Récapitulatif du message créé*

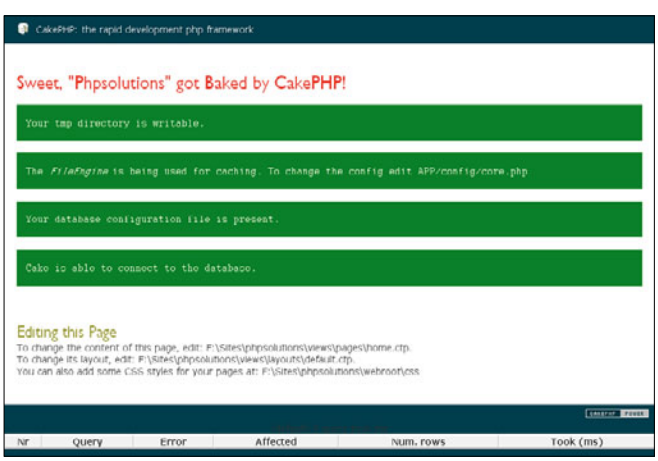


Figure 3. *Aperçu de l'application créée en ligne de commande*

La console propose ensuite de définir les associations existantes entre cette table et les autres tables de la base. Quatre types d'associations sont gérées par CakePHP : *hasOne* (possède un seul), *hasMany* (possède plusieurs), *belongsTo* (appartient à) et *hasAndBelongsToMany* (possède et appartient à plusieurs). Si nous respectons les conventions de nommage des tables et des champs (nom des tables au pluriel, '`id`' pour les clés primaires, nom de table au singulier suivi de '`_id`' pour les clés étrangères), CakePHP détecte seul les paramètres de chaque association, mais il reste tout à fait possible de créer les modèles à partir de tables venant d'une ancienne application qui ne respecte pas ces conventions. Notre premier modèle est créé !

Tableau 1. *Relations possibles entre différents modèles*

Relation	Nom de la relation	Exemple
1 à 1	hasOne	Un utilisateur a un profil
1 à n	hasMany	Un utilisateur peut être l'auteur de plusieurs recettes
n à 1	belongsTo	Une recette appartient à un utilisateur
n à n	hasAndBelongsToMany	Une recette est associée à plusieurs tags, et un tag est associé à plusieurs recettes

Création d'un contrôleur

Passons maintenant à la génération d'un contrôleur : la console nous donne la liste des contrôleurs possibles en fonction des tables de la base, puis propose de créer dans ce nouveau contrôleur les actions de base (`index`, `add`, `edit` et `delete`), et éventuellement les actions de base d'administration (`admin_index`, `admin_add`, `admin_edit` et `admin_delete`).

Création d'une vue

Terminons par la création des vues : CakePHP se charge de créer une vue pour chaque action du contrôleur. Nous pouvons à présent parcourir l'application dans un navigateur : tout est déjà prêt, il ne reste qu'à affiner la présentation des données ou certains traitements, mais l'application en l'état est totalement fonctionnelle, après avoir tapé une ligne de commande et répondu à quelques questions !

Note avancée : dans le cadre d'industrialisation de vos développements, sachez que le code généré peut être personnalisé en créant de nouveaux *templates*. Vous pouvez donc améliorer la productivité de votre équipe en ajoutant de meilleurs commentaires, des exceptions ou des tests unitaires pour le code généré.

Les modèles

Les modèles, l'une des trois couches du modèle MVC, constituent le cœur de l'application. Ils encapsulent l'accès aux tables de la base de données et sont responsables de la plupart des manipulations de ces données : lecture, écriture, suppression et validation du format des données avant sauvegarde. Le framework est prévu pour pouvoir travailler avec un grand nombre de SGBD différents de façon transparente (MySQL 4 et 5, Postgres, Firebird, SQL Server, Oracle, SQLite, ODBC et ADOdb).

Les modèles sont conçus dans l'optique d'une grande souplesse puisqu'il est possible d'utiliser un modèle sans table (gestion d'un formulaire de contact par exemple), que l'on peut spécifier des bases de données différentes d'un modèle à l'autre, et qu'un modèle peut manipuler des données issues d'une autre source qu'une table de base de données, par exemple une API externe. Il suffit pour cela de créer sa propre *datasource* et de brancher le modèle dessus. Depuis la version 1.3, un projet officiel regroupe des sources de données communautaires maintenues par les utilisateurs telles que Amazon, fichiers CSV, SOAP ou XML-RPC.

Note : toutes les classes de modèles de l'application étendent la classe `AppModel`, il est donc facile de factoriser certaines actions redondantes sur plusieurs modèles à l'intérieur de cette classe mère.

Opérations usuelles sur les données

Examinons un contrôleur créé plus tôt via la console : nous avions demandé à CakePHP de créer pour nous les actions de base de manipulation de données; penchons-nous dans le Listing 6 sur l'action `add()` permettant d'ajouter une entrée à la base de données.

A l'intérieur d'une action d'un contrôleur, les données issues d'un formulaire présent dans la vue de cette action sont toutes regroupées dans `$this->data`. La première étape vérifie donc si cette variable existe (le formulaire a été soumis) et dans ce cas, la méthode `create()` du modèle `Recette` est appelée : CakePHP prépare une enveloppe vierge qui va accueillir les données à sauvegarder. Il essaie ensuite de sauvegarder effectivement les données soumises, et redirige sur l'index du contrôleur en cas de succès (c'est-à-dire d'une part que les données ont passé tous les tests de validation, et d'autre part que la ligne a bien été ajoutée dans la base).

Relations entre modèles

Une des grandes forces de CakePHP réside dans la gestion automatique des relations entre les modèles. Les quatre relations possibles sont décrites dans le Tableau 1.

Une fois les relations définies dans le modèle, on peut accéder directement à l'un des modèles associés : si par exemple dans l'action `add()` du contrôleur des recettes, nous souhaitons obtenir la liste des utilisateurs nous utiliserons l'instruction présentée dans le Listing 7. La gestion de ces relations permet aussi de rapatrier en un seul appel les données d'un ou plusieurs enregistrements du modèle ainsi que des modèles associés. La *distance* jusqu'à laquelle CakePHP doit parcourir les associations peut être réglée grâce à la variable de modèle `$recursive`.

Analysons l'action `view()` de notre contrôleur des recettes dans le Listing 8. La variable `$recette` contiendra non seulement les données de la recette d'identifiant

`$id`, mais aussi les informations sur l'utilisateur qui l'a écrite et les tags associés à la recette ! Ce formidable automatisme se retrouve également dans la sauvegarde des enregistrements, puisque l'on peut sauvegarder l'enregistrement d'un modèle et de ses modèles associés en un seul appel à `save()` ou `saveAll()`.

Les callbacks

Les *callbacks* sont des méthodes spéciales qui sont déclenchées automatiquement par le framework lors de chaque appel à une page. On trouve par exemple les méthodes de modèle `beforeSave()` ou `afterFind()`, qui peuvent accueillir des traitements supplémentaires en plus des méthodes de base. D'autres *callbacks* sont disponibles dans les contrôleurs (`beforeFilter`, `beforeRender`...).

Extensions de modèle

Dans l'optique de produire du code le plus réutilisable possible, CakePHP permet d'étendre les fonctionnalités de base d'un modèle avec des Behaviors (Comportements), qui ajoutent des méthodes à un modèle existant ou utilisent les *callbacks* pour effectuer des traitements supplémentaires. CakePHP en fournit quelques-uns, comme par exemple le comportement *Tree* qui permet de gérer une table comme une arborescence. La gestion des nœuds se fait en toute simplicité : la logique se trouve dans des *callbacks* du Comportement, appelés avant et après la sauvegarde classique d'un enregistrement.

Exemple de Behavior : Translate

Ce comportement, inclus dans le framework, permet de gérer une application multilingue. Il faut avant tout préparer la base de données à accueillir plusieurs langues : pour ce faire nous devons créer une table `i18n` (dont le schéma est fourni dans le répertoire *config/schema*) qui va contenir tous les champs textuels des autres tables, et ces autres tables ne doivent pas contenir de champs susceptibles d'être traduits. Nous associons ensuite le comportement *Translate* au modèle en précisant quels champs textuels devront être traduits grâce à l'attribut `$actsAs` du modèle, tel que présenté dans le Listing 9.

La gestion des traductions se fait ensuite de manière transparente : avant toute action sur un enregistrement, il suffit de préciser sur quelle(s) langue(s) on travaille, et *Translate* s'occupe du reste. Ainsi pour afficher une recette en anglais :

```
function view($id) {
        $this->Recette->locale = 'eng';
        $this->Recette->id = $id;
        $recette = $this->Recette->read();
…
```

Et CakePHP renvoie un enregistrement avec les champs '`titre`' et '`description`' traduits, comme si nous avions affaire à la table recettes avec les champs traduits en anglais, alors qu'en réalité ces deux champs n'existent pas dans la table recettes mais dans la table `i18n`.

Listing 6. *Méthode classique de création d'un nouvel objet en base de données*

```
function add() {
        if(!empty($this->data)) {
                $this->Recette->create();
                if ($this->Recette->save($this->data)) {
                        $this->Session->setFlash(__('The Recette has been saved', true));
                        $this->redirect(array('action'=>'index'));
                } else {
                        $this->Session->setFlash(__('The Recette could not be saved. Please, try again.', true));
                }
        }
}
```

Listing 7. *Exemple de récupération de tous les utilisateurs existants*

```
class RecettesController extends AppController {
        function add() {
                $utilisateurs = $this->Recette->Utilisateur->find('all');
...
```

Listing 8. *Récupération d'une entrée de la base de données et des informations associées*

```
class RecettesController extends AppController {
        function view($id) {
                $this->Recette->id = $id;
                $this->Recette->recursive = 1;
                $recette = $this->Recette->read();
…
```

Listing 9. *Association du comportement Translate à un modèle et paramétrage*

```
class Recette extends AppModel {
        var $actsAs = array(
                'Translate' => array(
                        'titre' => 'Titres',
                        'description' => 'Descriptions'
                )
        );
…
```

Les vues

Les vues de CakePHP n'utilisent pas de langage spécifique, et sont donc du pur (x)HTML au milieu duquel on utilise le PHP classique pour afficher les données dynamiques. Il est cependant possible d'utiliser des moteurs de templates comme Smarty, mais c'est en pratique très peu utilisé à cause des nombreux *helpers* disponibles qui permettent de garder un code propre, minimal, réutilisable et facile à prendre en main !

Différents éléments composant une vue

Dans un but de ré-utilisabilité du code et pour diminuer la redondance, les vues sont composées de plusieurs fichiers.

- *Layouts* : un fichier gabarit qui contient toute la structure commune aux pages du site (menus, entête et pied de page …), il est donc très simple de changer radicalement de structure de page entre sections du site ou versions,
- *Elements* : certains éléments communs à plusieurs pages (exemple : formulaires de connexion) peuvent être mis sous forme d'un fichier séparé qui sera inclus dans chacune des pages, avec possibilité de passage de paramètres,
- *Corps de la page* : associés à chacune des actions, les fichiers de vue contiennent le code de la page strictement nécessaire à l'action demandée (formulaire, affichage des informations) ce qui permet de les réutiliser simplement d'un projet sur l'autre.

Des helpers pour se simplifier la vie

Comme vous commencez à le voir, le framework contient énormément de fonctionnalités n'ayant qu'un seul but : faciliter le développement en s'abstrayant des tâches rébarbatives ! Ainsi, CakePHP est livré avec des classes contenant des méthodes vraiment pratiques : les *helpers*.

Le FormHelper

Afin de gérer simplement les formulaires affichés sur votre site et leur lien avec les données issues ou destinées à la base de données, une classe vous fournit un tas de méthodes *automatiques*. Voici dans le Listing 10, à titre d'exemple, le code permettant de générer le formulaire d'ajout de la Figure 1 (code généré automatiquement depuis la ligne de commande).

Nous pouvons voir dans cette portion de code que la création d'un formulaire se fait en définissant le modèle associé, et que Cake se chargera ensuite de déterminer le type d'élément de formulaire à afficher en fonction du type de données de la base de données : ici le champ `title` est un `VARCHAR`, `content` un `TEXT` et `user_id` une clé étrangère vers une entrée de la table `users`. Sachez également que vous pourrez insérer directement un type d'élément précis (*select, radio, checkbox* …) par l'appel de la méthode correspondante du *helper* … et qu'il existe beaucoup d'autres méthodes et options pour faire ce que vous voudrez !

Le HTMLHelper

De la même manière, une autre classe contient de nombreuses méthodes utiles pour vous faciliter l'affichage de code HTML. Quelques exemples : insérer des balises doctype conformes, un charset uniforme au site, des liens (vers des pages, images, fichiers) qui ne seront pas cassés par un changement de schéma d'url rewriting, créer simplement des tableaux de données et bien d'autres choses.

CakePHP contient un système de gestion d'url très fin (les routes) qui vous permettra de personnaliser entièrement vos urls en modifiant quelques lignes dans le fichier */config/routes.php*. L'intérêt du HTMLHelper est qu'il se chargera de générer automatiquement un lien correspondant à ces paramètres afin de pouvoir modifier très simplement votre système d'urls. Note avancée : dans la version 1.3 du framework, il est possible de créer des classes personnalisées pour gérer les routes. Cela apporte de nombreuses nouvelles possibilités qui ne peuvent être décrites dans le cadre de cet article, mais nous vous invitons à vous documenter sur ce sujet (cf ressources mentionnées en fin d'article).

Le JsHelper

Introduit dans la version 1.3 du framework, ce *Helper* vous permet d'utiliser du Javascript et de l'Ajax dans vos vues en tirant pleinement partie de la magie du framework. Il est ainsi très simple de mettre en place des formulaires, liens, glisser-déposer et autres effets en un seul appel de méthode.
La nouveauté apportée par ce *Helper* est qu'il est agnostique : il suffit d'un changement de configuration pour générer du code jQuery, Mootools ou Prototype. De plus, un système de buffer vous permet de regrouper l'affichage de vos scripts en un seul et même endroit, et ce peut importe d'où le helper a été appelé (layout, vue, *helper* ou élément).

…. et bien d'autres

RSS, Paginator (pour paginer simplement vos résultats), Number, Session, XML … vous permettront de mettre en place en toute simplicité des fonctionnalités pratiques et répandues. Ceci est une liste non exhaustive se basant sur les *helpers* du cœur du framework : la communauté en propose beaucoup d'autres, et vous pouvez très simplement créer les vôtres !

Des composants livrés avec le framework

Au même titre que les Behaviors sont des extensions de modèle, les Components (composants) sont des extensions de contrôleur. Parmi ceux livrés avec CakePHP (Cookies, Sessions, Email, Security, etc.), nous en aborderons deux : Auth et ACL.

Auth

Le composant *Auth* permet de mettre en place une authentification par login / mot de passe sur certaines parties du site. L'utilisation typique de cette fonctionnalité est la protection d'un espace d'administration du site, c'est-à-dire l'authentification requise pour les actions préfixées par '`admin_`'. Commençons pas activer le routage automatique des actions réservées à l'administration en décommentant une ligne dans *config/core.php* :

```
Configure::write('Routing.prefixes',
array('admin'));
```

Ainsi, une URL du type '*/admin/recettes/edit/12*' renverra sur l'action `admin_edit($id = null)` du contrôleur

`RecettesController`, avec comme paramètre '12', l'identifiant de la recette à éditer. Depuis la version 1.3 vous pouvez créer et gérer différents préfixes, ce qui est très pratique pour créer divers espaces membres ou sections d'une application. Toujours dans l'esprit *convention au lieu de configuration*, CakePHP attend que les utilisateurs autorisés soient stockés dans une table '`users`' avec un champ '`username`' (le login) et un champ '`password`'. Ce dernier devra être de type `varchar (40)` pour stocker la clé de hashage du mot de passe, composée de la clé de sécurité générée lors de la création de l'application avec la console, suivie du mot de passe, le tout crypté en SHA1 ce qui nous donne une chaine de 40 caractères.

Afin de restreindre l'accès à toutes les actions de nos contrôleurs commençant par `admin_` aux seuls utilisateurs logués, nous allons placer la logique d'authentification dans la classe `AppController` (Listing 11) dont héritent tous les autres contrôleurs. Comme cette logique doit intervenir avant toute autre opération lors de l'appel à une action, nous utiliserons le callback `beforeFilter`, qui comme son nom l'indique est appelé avant l'action elle-même.

Il suffit de définir quelques propriétés du composant *Auth* (les URL de connexion, de déconnexion et de redirection en cas de succès, les messages d'erreur, la condition d'autorisation ou de refus) et l'accès à une action d'administration demande maintenant de s'identifier. Il ne nous reste qu'à créer les actions de *login* et *logout* dans le contrôleur des utilisateurs, `UsersController` présenté dans le Listing 12.

L'action *login* reste vide, CakePHP se charge lui-même de vérifier si le couple login / mot de passe, qui sera fourni par le biais du formulaire à créer dans la vue de cette action, correspond bien à un utilisateur de la table users. L'action *logout* ne fait que rediriger sur l'URL définie comme URL de déconnexion dans le `beforeFilter`, avec un message.

ACL

Le composant ACL (*Access Control List*) permet un contrôle poussé des autorisations d'accès en considérant d'une part des rôles (aros) et d'autre part des ressources (acos), avec des relations d'autorisation ou d'interdiction entre un rôle et une ressource donnés. Notons bien qu'il ne s'agit pas de la même chose que l'authentification vue plus haut, mais d'un processus de contrôle d'accès aux ressources une fois qu'un utilisateur a été reconnu.

La solution proposée par CakePHP offre une granularité importante, puisqu'il est possible qu'un rôle soit un utilisateur ou un groupe d'utilisateurs, et qu'une ressource soit un enregistrement précis d'une table, une action, un contrôleur complet ou même l'application dans son ensemble. Le framework gère ces données sous forme de deux arborescences, une pour les rôles et une pour les ressources, il est donc possible (et préférable !) de ne pas définir toutes les autorisations d'accès pour tous les rôles vers toutes les ressources, mais de définir une autorisation pour un groupe d'utilisateurs entier vers une ressource, et chaque membre du groupe héritera des mêmes droits. Il reste ensuite possible de créer une exception sur l'un des membres du groupe sans influer sur les autres membres.

Le composant fournit des méthodes pour d'une part autoriser ou interdire l'accès d'un rôle à une ressource, et d'autre part pour contrôler si une autorisation d'accès

Listing 10. *Formulaire d'ajout d'un «message» dans le fichier / views/messages/add.ctp*

```
<?php echo $this->Form->create('Message');?>
        <fieldset>
                <legend><?php __('Add Message');?></
legend>
        <?php
                echo $this->Form->input('title'); //
Affichera un "input text"
                echo $this->Form->input('content');
// Affichera un "textarea"
                echo $this->Form->input('user_id');
// Affichera un "select" contenant les couples id/
name des utilisateurs existants
        ?>
        </fieldset>
<?php echo $this->Form->end('Submit');?>
```

Listing 11. *Configuration du composant d'Authentification et restriction des pages d'administration*

```
class AppController extends Controller
{
        var $components = array('Auth');

        function beforeFilter()
        {
                $this->Auth->loginAction =
array('plugin' => null, 'controller' => 'users',
'action' => 'login');
                $this->Auth->loginRedirect =
array('controller' => 'recettes', 'action' =>
'index');
                $this->Auth->logoutRedirect = '/';
                $this->Auth->loginError =
"Identifiant ou mot de passe incorrects.";
                $this->Auth->authError = "Vous
n'avez pas accès à cette page.";

                if(empty($this->params['prefix']) or
$this->params['prefix'] != 'admin')
                {
                        $this->Auth->allow();
                }
        }
}
```

Listing 12. *Contrôleur basique de gestion des utilisateurs, tout est fait par Cake !*

```
class UsersController extends AppController {
        function login() {

        }

        function logout() {
                $this->Session->setFlash("Vous êtes
maintenant déconnecté.");
                $this->redirect($this->Auth-
>logout());
        }
}
```

Sur Internet

- *http://www.cakephp.org/* – Le site officiel du framework contenant entre autres la documentation et l'API,
- *http://www.cakephp-fr.org/* – Site de la communauté francophone principalement constitué d'un forum d'aide et de conseils,
- *http://www.formation-cakephp.fr/* – Blog francophone présentant clairement diverses fonctionnalités de CakePHP et leur mise en application,
- *http://www.planetcakephp.org/* – Site recensant les flux des différents blogs CakePHP.

existe entre un rôle et une ressource donnés. De plus, il existe un comportement du même nom (ACL) qui permet de gérer automatiquement les rôles, par exemple si nous voulons créer un rôle à chaque création d'un utilisateur.

La communauté de CakePHP

CakePHP est un projet open source avec une communauté développé et très active. De nombreux blogs abordent diverses astuces ou donnent quelques conseils, et les ressources officielles sont de très bonne qualité. Voici donc les principales sources d'informations proposées par la communauté.

La documentation

La documentation (également appelée le *CookBook*) couvre la quasi-intégralité des sujets et est mise à jour / améliorée régulièrement. Chacun peut y contribuer en soumettant des modifications qui seront ensuite validées par les responsables. La traduction française couvre environ 90% du contenu, et ne cesse d'augmenter. Url : *http://book.cakephp.org/fr*

L'API

Basée sur la version actuelle du framework, l'API est automatiquement générée depuis le code de l'application et ses commentaires. Un moteur de recherche vous permet de trouver rapidement une méthode afin de voir précisément quelle est sa spécification, et même d'en lire le contenu. La navigation dans le code du cœur est ainsi très simple ! Url : *http://api.cakephp.org/classes*

Forums d'aide

La communauté anglophone utilise un groupe Google afin de fournir de l'assistance aux développeurs. En France un forum permet de dialoguer, ou de demander de l'aide sur l'utilisation du framework. Url : *http://forum.cakephp-fr.org/*

CakeQs

Un nouvel outil international a récemment été mis en place par la Cake Software Foundation. Dans l'esprit de stackoverflow.com, il permet de poser une question à la communauté et d'obtenir une aide rapide et précise. Url : *http://cakeqs.org/fre*

Blogs et autres ressources

De nombreux blogs anglophones et francophones traitent du développement CakePHP. En France on trouve notamment le blog *http://www.formation-cakephp.com* qui est une mine d'or de tutoriaux de qualité. Pour les blogs non francophones, le site *http://planetcakephp.org/* recense les flux des meilleurs blogs dédiés à CakePHP.

D'autre part, un channel IRC vous permettra de dialoguer avec les développeurs du cœur du framework et d'autres utilisateurs pour avancer dans le développement de vos applications : *#cakephp-fr* (ou *#cakephp* pour la version anglophone) sur *irc.freenode.net.* C'est un passage recommandé lorsque vous débutez avec le framework … la communauté vous attend !

Participez au projet !

Après la récente publication de la version 1.3 du framework, l'équipe de CakePHP a lancé des appels aux utilisateurs pour contribuer au développement du projet. Ainsi, toutes les suggestions et contributions pour la version 2.0 entièrement PHP5 sont les bienvenues. Les outils de la communauté vont également être actualisés et harmonisés pour devenir (à l'image de la documentation ou CakeQs) entièrement internationalisés et répondre aux attentes des utilisateurs.

Conclusion

Comme nous l'avons vu tout au long de cet article, CakePHP a été conçu de manière à permettre un développement rapide d'applications web fonctionnelles sous tous les aspects. Historiquement anglophone mais désormais international, sa communauté est très vaste et réactive. Vous pourrez ainsi débuter en vous basant sur de nombreux tutoriaux, la documentation et l'assistance à votre disposition, notamment en Français. Nous vous invitons à suivre les quelques exemples présentés ici, puis à découvrir les nombreuses autres fonctionnalités au fur et à mesure de vos besoins.

PIERRE-EMMANUEL FRINGANT
PIERRE MARTIN

Pierre-Emmanuel Fringant est webmaster indépendant à Caen dans le Calvados (Normandie). Après un DUT informatique obtenu à l'IUT de Ifs, il acquiert six ans d'expérience dans plusieurs entreprises (agence de communication, webagencies), puis se lance dans l'aventure indépendante début 2007.

Pierre Martin est développeur web, expert CakePHP au sein de la société CakeDC. Passionné de programmation, il s'est tourné vers le développement web lors de stages et par son activité de développeur web indépendant ces deux dernières années avant d'intégrer CakeDC. Il travaille désormais à plein temps autour du framework, pour des clients internationaux.

Serveur : un ami ou un ennemi ?

Un serveur est indispensable pour réaliser un projet web. C'est lui qui va mettre à disposition votre projet. Cependant, si celui-ci est correctement paramétré, il sera votre ami, mais il peut très vite devenir votre ennemi s'il n'est pas configuré ou sécurisé correctement.

Cet article explique :
- Comment adapter votre application au serveur.

Ce qu'il faut savoir :
- Notions de PHP.

Suivre la vie de votre serveur correspond à prendre soin de celui-ci car il peut cacher de nombreux services. Ces différents services peuvent s'exécuter séparément ou être associés à votre projet. Mais lorsqu'un serveur existe, il peut aussi proposer d'autres projets réalisés par d'autres équipes. Cependant lorsque vous réalisez un projet informatique, sous la forme d'une application Web, qui utilisera les différents réseaux informatiques comme l'intranet, l'internet, l'extranet, il peut être judicieux d'écouter votre serveur.

Le but de ces tests n'est pas de remplacer les administrateurs, ni les personnes qui gèrent le réseau. Il s'agit plutôt d'adapter votre application au serveur pour définir des choix. Si ces choix ne sont pas appliqués rapidement, cela pourrait être lourd de conséquences lors de la mise en production. Ces conséquences concernent avant tout, le risque provenant de visiteurs extérieurs.

Décomposition d'un serveur

Un serveur informatique peut comprendre de nombreux services comme :

- Un service de messageries.
- Un serveur web.
- Un serveur de base de données.

Un service de messageries

Un service de messageries correspond à une gestion de *Webmail*. Suivant le choix de la messagerie, les protocoles peuvent être IMAP ou encore SMTP. Ce service est souvent associé à un projet web, mais il peut aussi être utilisé pour la redirection des emails auprès des destinataires enregistrés.

Un serveur web

Le serveur web, appelé AMP, comprend un environnement Apache, avec un langage PHP et une base de donnée MySQL. Des nombreuses extensions sont disponibles et peuvent être installées pour contribuer au bon fonctionnement de votre projet. Les extensions les plus courantes lors de l'installation d'un serveur Web concernent les fonctions de messageries, de FTP, de

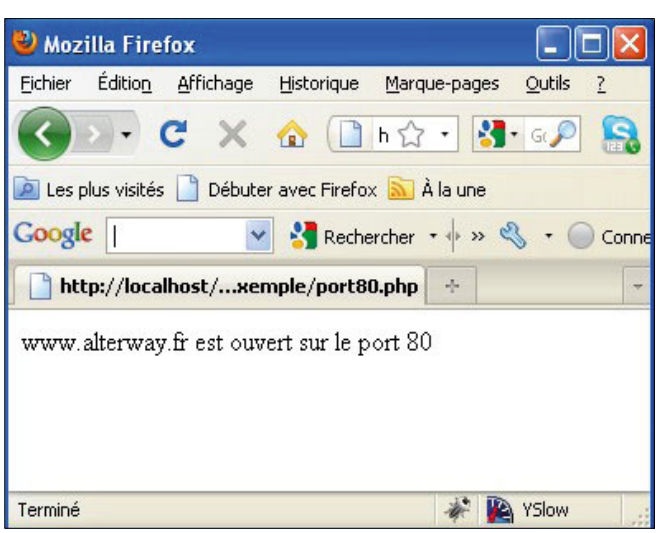


Figure 1. *Le site existe*

mémoire cache, de communication externe (cURL par exemple)...

Un serveur de base de données
Le serveur de base de données peut correspondre à une interface, permettant de gérer vos bases de données. Ce type d'interface peut correspondre à PHPmyAdmin.

Le savoir

Lors de l'installation d'un nouveau serveur web, la configuration de celui-ci propose de nombreuses informations qui vont vous être utiles. Sans ces informations, vous n'aurez pas le savoir requis pour continuer votre projet. Les informations principales concernent :

- La version du langage PHP.
- Le format de la base de données.
- Les extensions disponibles.

Grâce à cela, vous possédez exactement les fonctions mises à votre disposition. Or, pour utiliser les fonctions de votre langage, vous devez penser aux protocoles disponibles comme :

- FTP,
- HTTP,
- SMTP,
- POP3,
- MySQL.

La connaissance
Lors du contrôle de votre serveur, les informations qui vous sont proposées permettent de connaître exactement les ports ouverts. Ces ports sont uniques pour un protocole donné. Il s'agit par exemple :

- du port 80 dans le mode HTTP,
- du port 21 qui sera utilisé pour l'accès FTP.

Bien entendu, les ports peuvent être modifiés suivant les orientations et le choix des administrateurs des serveurs. La définition de ces ports permet de savoir si le service est actif ou pas. La théorie permet d'effectuer un test interrogeant un service, en demandant si celui-ci est présent ou absent. Suivant la réponse retournée, vous saurez si celui-ci est disponible.

Vérification
Pour effectuer la vérification, vous pouvez utiliser le langage PHP pour contrôler si la connexion est toujours active. En PHP, la fonction disponible est `fsockopen()`. L'utilisation de la fonction permet de vérifier si l'adresse d'un serveur fourni est accessible par un numéro précis et utilise un port spécifique comme celui de la navigation HTTP (port 80).

Listing 1. *Vérifier si un site internet est ouvert*

```
<?php

$url="www.alterway.fr";
$port="80";

$cnx = @fsockopen($url, $port);
if ($cnx)
{
    echo "$url est ouvert sur le port $port";
    fclose($cnx);
} else {
    echo "$url est fermé sur le port $port";
}
?>
```

Listing 2. *Test les ports*

```
<?php

$adr="127.0.0.1";

for($i=1;$i<=100;$i++)
{
        $cnx = @fsockopen($adr, $i, $errno, $errstr,
0.1);
        if ($cnx)
        {
        echo "Port $i est ouvert.<br>";
        fclose($cnx);
        }
}
?>
```

La fonction `fsockopen()` se déclare de la façon suivante :

```
fsockopen ( nom du serveur,
            numéro du port,
            numéro de l'erreur du serveur
système,
            le message d'erreur,
            le délai d'attente)
```

Dans notre premier exemple, nous utilisons les 2 premiers arguments. L'exemple donné pour vérifier si un site internet existe, consiste à fournir une URL si vous ne connaissez pas l'adresse IP de la machine, et un numéro de port (voir Listing 1). Ici, nous utilisons le port 80 car il s'agit du port de votre navigateur par défaut.

Comme le montre l'exemple (Listing 1), si le site existe et c'est bien le cas, un message de succès sera affiché de la façon suivante (voir Figure 1).

Contrôle des ports
Lorqu'un nouveau serveur est monté, une des étapes que vous devez aussi effectuer pour votre projet web consiste à vérifier les ports. Ces ports peuvent être considérés comme des portes et donc des voies d'accès à une application spécifique. Cette opération est complémentaire aux tests qui sont déjà réalisés au niveau de la sécurité.

Bien entendu, ce test s'effectue par l'intermédiaire du navigateur et permet de vérifier si un des services

Listing 3. *Fonction scan*

```
<?php

function scan _ port($adresse,$port)
{
    $cnx = @fsockopen($adresse, $port, $errno,
$errstr, 0.1);
    if ($cnx)
    {
        fclose($cnx);
        return true;
    }
    return false;
}
?>
```

Listing 4. *Sélection des ports*

```
<?php

$port = array(
          '80'=>'HTTP'
                ,'21'=>'FTP'
         ,'22'=>'SSH'
         ,'25'=>'SMTP'
         ,'110'=>'POP3'
         ,'143'=>'IMAP'
         ,'3306'=>'MySQL'
                );
?>
```

Listing 5. *Boucle de test*

```
<?php

echo "<table>";
foreach ($port as $id=>$nom)
{
         echo "<tr>";
         echo "<td>".$nom."</td>";
         echo "<td>";
         $row=scan _ port ("127.0.0.1",$id);
         if ($row == TRUE)
                echo "Ouvert";
         else
                echo "Fermé";
         echo "</td>";
         echo "</tr>";
}
echo "</table>";
```

proposés sur votre serveur possède une faille de sécurité. Si le cas se présente, il peut être corrigé en cours de réalisation de votre projet. L'opération de test est présentée dans le Listing 2.

L'exemple montre que vous testez votre serveur local, du port 1 à 100. Ici la fonction `fsockopen` est utilisée avec l'ensemble des options pour accélérer l'affichage des résultats. On a défini 1 seconde d'attente si le port ne répond pas de suite.

Le résultat obtenu est montré dans la Figure 2.

Le miroir

La sécurité est très importante, mais vouloir trop protéger peut bloquer certaines fonctions essentielles en PHP. Si vous bloquez le port 3306, vous ne pourrez pas utiliser la base de données MySQL sauf si vous avez décidé d'utiliser un port différent.

Pour effectuer ces tests, vous devez posséder un autre serveur pour vous permettre d'effectuer des tests à distances. Ce deuxième serveur peut être votre ordinateur avec un environnement AMP déjà installé. L'opération présentée propose 2 avantages :

- La possibilité de tester le réseau à tout moment même en cas de panne du serveur de production.
- Celle de contrôler les ports dont vous aurez besoin lors des évolutions de votre projet.

La réalisation de ces tests peut s'effectuer avec un petit script en PHP. Le script devra proposer :

- Le test des ports.
- La liste des numéros à tester.
- Le résultat obtenu.

Les ports

Pour faciliter les tests des ports, vous pouvez définir une fonction qui sera interrogée à chaque fois qu'un port sera sollicité. La fonction nous retournera une valeur positive si celui-ci est ouvert (Listing 3) . Sinon il n'y aura aucun message de retour.

La liste

Il est important de définir la liste des ports dont vous aurez besoin (voir Listing 4). Ces ports permettent de s'assurer du bon fonctionnement de votre projet web.

La liste présentée est une liste indicative car il s'agit des ports classiques. Vous pouvez ajouter des ports supplémentaires à partir d'informations provenant entre autres du site frameip (*http://www.frameip.com/liste-des-ports-tcp-udp/*). Ce site propose une description de l'ensemble des 65543 ports qui existent actuellement.

Les résultats

Pour obtenir un résultat clair, vous pouvez afficher ligne par ligne dans un tableau les ports que vous avez déclarés dans le chapitre précédent. A partir d'une boucle `Foreach()`, l'exemple (Listing 5 – boucle de test) va afficher le nom du port et signaler si celui-ci est ouvert ou fermé.

A chaque passage de la boucle, l'analyse du port sera effectuée par rapport à l'adresse du serveur. Ici, le test porte sur le serveur local. Le résultat obtenu est présenté dans la Figure 3. Le résultat montre que le port de navigation et l'accès à la base de données sont ouverts.

Vérifier son code

Après avoir contrôlé le serveur au niveau des accès des ports et donc de la sécurité, vous devez aussi contrôler votre code. La sécurité de votre application est très importante, car elle peut aussi être la cause d'infections de votre serveur.

Pour s'assurer d'un bon fonctionnement de votre serveur, vous devez réaliser une programmation sécurisée. Pour cela, de nombreuses fonctions existent, mais la majorité doivent se limiter au serveur de développement et non à la production.

Attention cependant : cette partie ne va traiter que les points de sécurité propres à un projet web, or lorsque vous réalisez du code, vous pouvez être amené à effectuer des relations extérieures comme l'échange de données ou l'exécution de programmes annexes. Et c'est ce point qu'il ne faut pas négliger envers votre serveur.

Les exécutions

Les fonctions d'exécution disponibles en PHP 5 permettent d'exécuter des commandes sur le systeme lui-même. Les fonctions disponibles vont vous permettre d'exécuter un programme et d'obtenir un résultat, comme les fonctions `System()`, `passthru()`, `exec()`, `shell_exec()`.

Il est possible d'approfondir ces éxécutions de programmes avec un degré de contrôle plus important. Vous pouvez trouver les fonctions comme `proc_open()`, `proc_close()`... mais avec la possibilité de modifier les processus.

Contrôle du processus

Le contrôle du processus de PHP implémente un système de création, de gestion des processus comme sous Unix. Le contrôle des processus s'utilise comme un mécanisme de rappel du gestionnaire de signaux.

Les fonctions disponibles concernant les fonctions '`pcntl_()`' sont nombreuses, comme pour l'ouverture, la duplication ou la fermeture. Mais vous pouvez en plus autoriser des interactions avec des processus à travers PTY ou les standards IEEE 1003.1 (POSIX.1).

L'extension sémaphore

L'extension sémaphore propose une interface pour les fonctions de type *System V IPC*. Les fonctions intégrées vont vous permettre d'utiliser de la mémoire partagée et un système de communication inter-processus (le IPC). Les fonctions proposées sont les fonctions '`shm()`', '`msg_send()`', '`ftok()`'...

Les fonctions communes

Les fonctions communes sont des fonctions standards à PHP, et donc très utilisées. Vous pouvez rencontrer des fonctions telles que `require()`, `include()`, `file()`, `fopen()`, `preg_replace`, `$_GET` / `$_POST` / `$PHPSELF`...

Si vous n'utilisez pas correctement ces fonctions en plaçant des filtres aux variables pour la sécurité, elles peuvent aussi provoquer des dommages pour votre projet, cela à moyen/long terme en permettant d'infecter votre serveur.

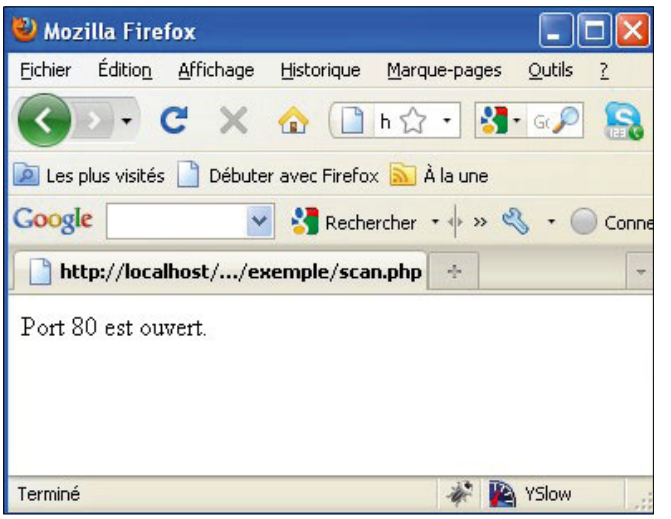


Figure 2. *Liste des ports ouvert*

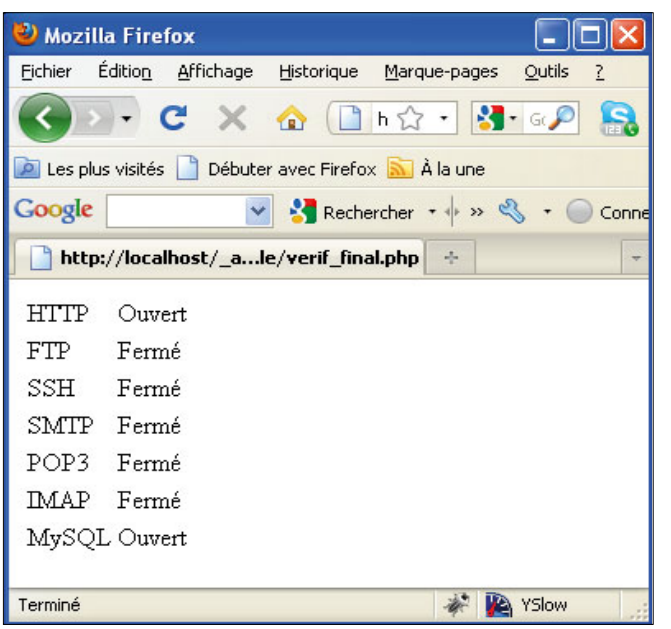


Figure 3. *Les ports ouverts et fermés*

Conclusion

Le serveur dans une entreprise est un axe très important. Il est même considéré comme le cœur de l'entreprise. Par ailleurs, il permet d'effectuer la relation entre les visiteurs et les différents services internes.

C'est pourquoi il est important de bien le paramétrer, de contrôler les ports à ouvrir et à bloquer, mais aussi de contrôler certaines fonctions qui pourraient indirectement vous poser des problèmes à cause d'une programmation non sécurisée.

CHRISTOPHE VILLENEUVE

Consultant, auteur du livre PHP & MySQL-MySQLi-PDO, Construisez Votre Application, livre français aux Éditions ENI, 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

Mettez du piment dans vos solutions Web !

- 1 pincée d'Open Source
- 1 zeste de créativité
- 1 cuillère à café d'interactivité
- 1 concentré de technologie

Core-Techs
Ouverture facile !

Tél. 01 42 01 34 05

www.core-techs.fr

Portails Internet - Ecommerce - Intranet - Extranet - GED - CRM - Développements spécifiques

WEB Gazelle

Génialement rapide

Votre site est-il vraiment rapide ?

Les sites WebGazelle sont conçus pour s'afficher en moyenne 20% plus vite que la majorité des sites Internet

En savoir plus : www.webgazelle.net

N°Azur 0 810 810 815
PRIX APPEL LOCAL

Grand jeu concours
12 sites internet à gagner
Tirage au sort tous les mois

Lot d'une valeur de 2498.20 € TTC

Vous gagnez un site Internet qui comprend :

- le conseil et l'étude de votre projet,
- le graphisme de votre site,
- l'optimisation du référencement,
- l'intégration des contenus,
- le logiciel WebGazelle CMS 2.0,
- le support et l'hébergement.

Extraits de règlement
Jeu Concours sur tirage au sort, organisé par la société Cognix Systems, à destination des personnes morales et personnes physiques ayant la qualité de commerçant, artisan et auto-entrepreneur, ayant leurs siège social en France métropolitaine (Corse comprise ; DOM-TOM compris). Pour participer, il faut s'inscrire sur Internet à l'adresse http://www.webgazelle.net/jeu-concours. Le règlement est disponible sur notre site Internet : http://www.webgazelle.net/reglement.php. Il est déposé chez un huissier de justice et peut être demandé gratuitement sur demande écrite à l'adresse suivante : Cognix Systems, 65 avenue Aristide Briand, 35000 Rennes.

Présentation des lots
12 sites Internet sont à gagner, basés sur un Pack WebGazelle CMS 2.0 « Essentiel » (valeur d'environ 2500 € TTC). Ces sites Internet comprennent : Les services compris dans un pack WebGazelle « Essentiel », un espace d'hébergement et un nom de domaine rattaché au site remporté, pour une durée de un an.

Webgazelle.net, une marque de COGNIX systems