JOOMLA! | WEB 2.0 | ACTIONSCRIPT | FLASH | POO | APACHE

Le plus grand magazine sur PHP au monde

phpsolutions

Nouvelles technologies et solutions pour les développeurs PHP

PHP N° 8/2010 (44) ISSN 1731-4593

PHP ET SÉCURITÉ
PROTÉGEZ VOS APPLICATIONS !

PROGRAMMATION ORIENTÉE OBJET
MANIPULEZ LES SESSIONS SOUS FORME D'OBJETS PHP

CONTENT MANAGEMENT SYSTEM
CRÉEZ ET INTÉGREZ VOTRE COMPOSANT JOOMLA!

.HTACCESS ET .HTPASSWD
SÉCURISEZ VOTRE DOSSIER GRÂCE À APACHE

POUR LES DÉBUTANTS

WORLD WIDE WEB
DÉCOUVREZ LES APPLICATIONS WEB 2.0

PROJETS

ACTIONSCRIPT 2 ET FLASHVARS
FAIRE COMMUNIQUER FLASH ET PHP

N'ayez plus honte de faire votre site web...

REFLEXION**FAITE** - © Crédits photos : Image Source.

Avec les packs WebSite,
créez un site web élégant et performant en quelques clics.

DÉCOUVREZ UNE SOLUTION SIMPLE, RAPIDE ET INTUITIVE POUR CRÉER UN SUPERBE SITE WEB : LE VÔTRE.

- 250 modèles de sites de grande qualité
- Bibliothèque de 1000 images haute résolution
- 14 jours d'essai gratuits sans engagement
- Un seul pack qui couvre tous vos besoins (nom de domaine, hébergement, email...)
- Amen, un interlocuteur unique

DOMAINE · EMAIL · HÉBERGERMENT · SERVEUR · PUBLICITÉ · PROTECTION DE MARQUE · E-COMMERCE · CRÉATION DE SITE

Amen
A DADA COMPANY

0 892 55 66 77 (0,34 €/mn) | www.amen.fr

aditik
agence web

Webdesign

Référencement

Formation

www.aditik.com

Agence web spécialisée dans la création de **sites dynamiques**, l'**e-commerce**, site Internet clés en main.

bannières pub - jeu concours - quizz...

Organisme de formation agréé

Expertise Joomla

- Formation tous niveaux
- Création / adaptation de templates
- Réalisation et maintenance de sites

Intégration de sites E-commerce

- Magento
- Prestashop
- Virtuemart

Parmi nos références

www.rungis-fleurs.com
www.allintubes.com
www.vinespagnol.fr
www.rungis-boutiques.com
www.nbmglobalknowledge.com
www.wecontrol.ch
www.surveycopter.fr

Formation - Audit - Conseil - Réalisation - Webmastering

aditik formation

Des **professionnels** vous transmettent leur **savoir-faire**

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

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

Directrice de la publication :
Ewa Łozowicka

Dépôt légal :
à parution
ISSN : 1731-4593

Rédacteur en chef :
Łukasz Bartoszewicz

Couverture :
Sławomir Sobczyk

DTP :
Sławomir Sobczyk *Studio2W@gmail.com*

Composition :
Sławomir Sobczyk

Correction :
Valérie Viel, Thierry Borel, Barbara Bourdelles

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

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

Publicité :
publicite@software.com.pl

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

TABLE DES MATIÈRES

VARIA

PROJETS

DOSSIER

PRATIQUE

FICHE TECHNIQUE

POUR LES DÉBUTANTS

PHP Compta 5.2

PHP Compta est un logiciel distribué sous licence Open Source et destiné aux indépendants, les associations et les PME. Il est compatible avec la gestion comptable française et belge. Il va vous permettre de tenir les journaux comptable, de banque, une balance et rapprocher les écritures.
http://www.phpcompta.be/

Dolibarr 2.9

La nouvelle version de Dolibarr vient de sortir. Cette nouvelle version apporte de nombreuses fonctionnalités (document ODT, filtres, tableau de Gantt, TVA...) et propose de nombreuses améliorations (comme les frais de déplacements). Par ailleurs, ce logiciel ERP/CRM est plus rapide grâce à une amélioration du moteur.
http://www.dolibarr.fr

Tine 2.0

Il s'agit d'une plateforme collaborative associé à un CRM. Il va vous permettre de regrouper et de rassembler les informations importantes sur un même espace. Vous aurez à disposition un carnet d'adresses, une gestion de tables, un suivi de dossier, des exportations PDF, une version pour support mobile et une gestion administrative de la VOIP. L'ensemble de l'outil a été développé pour vous en faciliter l'utilisation.
http://www.tine20.org/

TokuDB 4.1

Il s'agit d'un moteur de stockage MySQL, qui va vous permettre d'améliorer les performances des bases de données volumineuses, en accélérant l'indexation par 10 à 50. Vous trouverez une interface simplifiée, une amélioration de la rapidité et une réponse parfaite aux transactions ACID.
http://tokutek.com/products/tokudb-for-mysql-v4/

Zend Framework 1.10.8

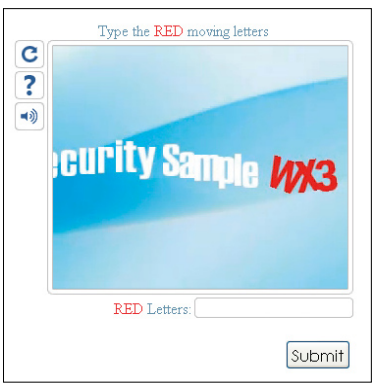
La nouvelle version de Zend Framework vient de sortir. Il s'agit d'une version mineure mais il est important d'effectuer la migration car elle comporte de nombreux correctifs importants mais aussi introduit les dernières modifications d'identification de Twitter.
http://devzone.zend.com/

MediaWiki 1.16.0

MediaWiki est un moteur de wiki pour le Web et vient de publier une nouvelle version avec de nombreuses nouveautés : les flux au format Atom, de nouvelles options (maintenance, préférences), accélérations avec la base de données. Par ailleurs, il est possible de bloquer l'envoi d'emails par les utilisateurs et le système de gestion ACL a évolué.
http://techblog.wikimedia.org/

NuCaptcha

NuCaptcha rejoint la famille des applications anti-spam mais il est complètement différent des captchas classiques. Ce nouveau projet a été lancé sur l'idée que les captchas deviennent de plus en plus complexes pour limiter le déchiffrage du code par les robots, et surtout plus ou moins lisibles pour un œil humain.
NuCaptcha se présente sous la forme d'une animation vidéo au format Flash, comme ceci lors de l'affichage de celui-ci dans un formulaire, vous êtes certain que la saisie effectuée est bien celle d'une personne et qu'il ne s'agit pas d'un robot.
L'affichage de l'animation se compose d'une image de fond et d'un texte en 2 couleurs, que vous aurez la possibilité de choisir parmi la liste proposée par le service du site.
La partie de validation concerne une partie du texte, représentée par des caractères rouges. Grâce à un affichage animé, le texte est complètement lisible par un œil, mais pas du tout par un robot.
Bien sûr, pour insérer ce captcha dans votre formulaire, l'opération s'effectue par l'intermédiaire d'une librairie et du langage PHP. Il se place n'importe où dans votre formulaire puisqu'une seule partie du texte est à saisir.
http://www.nucaptcha.com/



Forum PHP 2010

L'AFUP (*Association Française des Utilisateurs de PHP*) vient de dévoiler le programme du forum PHP 2010. Le forum PHP va se dérouler le 9 et 10 novembre prochain à la cité des sciences et de l'industrie à Paris pour la deuxième année consécutive.
Cette année n'est pas une année comme les autres car il s'agit d'un double anniversaire avec les 15 ans de PHP et les 10 ans de l'association. Et pour marquer l'évènement, de nombreux points vont vous faire venir à ce double anniversaire.
Tout d'abord, la présence du créateur de PHP : Rasmus Lerdoff, qui viendra nous parler de son langage favori : *Le PHP*. Ensuite, le contenu du programme est très diversifié autour du thème *PHP* comprenant de nombreux points qui vont vous intéresser comme :

- La découverte autour de PHP (Présentation, veille technologie, GIT / SVN).
- Architecture et Développement (Industrialisation, développement, base de données).
- Framework.
- Infrastructure (Hébergement, performance, sécurité).
- Qualité / Efficacité individuelle et collective.
- Applications métier : CMS et e-commerce.
- Le Web.

La totalité du programme, des conférences sont disponibles sur le site du Forum. Par ailleurs, vous trouverez toutes les informations pour réserver votre place :
http://www.afup.org/forumphp.

Rédaction des actualités :
Christophe Villeneuve

Création
d'un composant MVC Joomla!

Un composant se définit comme une application intégrée qui sert à étendre les fonctionnalités d'un système. Bien que certains composants tels que newsfeeds, banners, contact soient inclus lors de l'installation, Joomla! offre aux développeurs la possibilité de créer des composants personnalisés et adaptés à ses besoins.

Cet article explique :

- Cet article explique étape par étape comment s'y prendre pour créer et intégrer un composant Joomla! 1.6 Beta, respectant le modèle MVC.

Ce qu'il faut savoir :

- Connaissances du langage SQL.
- Bonne maîtrise du PHP et du HTML.
- Connaissance de Joomla!.

Le modèle MVC est une méthode de conception basée sur une architecture qui permet la séparation de l'application en trois couches : modèle, vue et contrôleur (voir Figure 1).

- Le modèle correspond aux données de l'application; il assure le traitement des données.
- La vue se charge d'afficher les données envoyées par le modèle et de recevoir les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc).
- Le contrôleur gère l'interaction et la synchronisation entre le modèle et la vue.

Interaction entre les trois couches

Le contrôleur se charge de prendre en charge les événements de l'utilisateur puis déclenche les actions pour synchroniser la vue et/ou le modèle. Si une action exige un changement des données, le contrôleur demande au modèle la modification des données ; le modèle avertit la vue que les données ont changé pour qu'elle se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande (voir Figure 2).

Les composants Joomla!

Les composants se répartissent en deux catégories :

- Les composants *Frontend* : ceux-ci constituent la partie publique c'est à dire visible par les visiteurs.

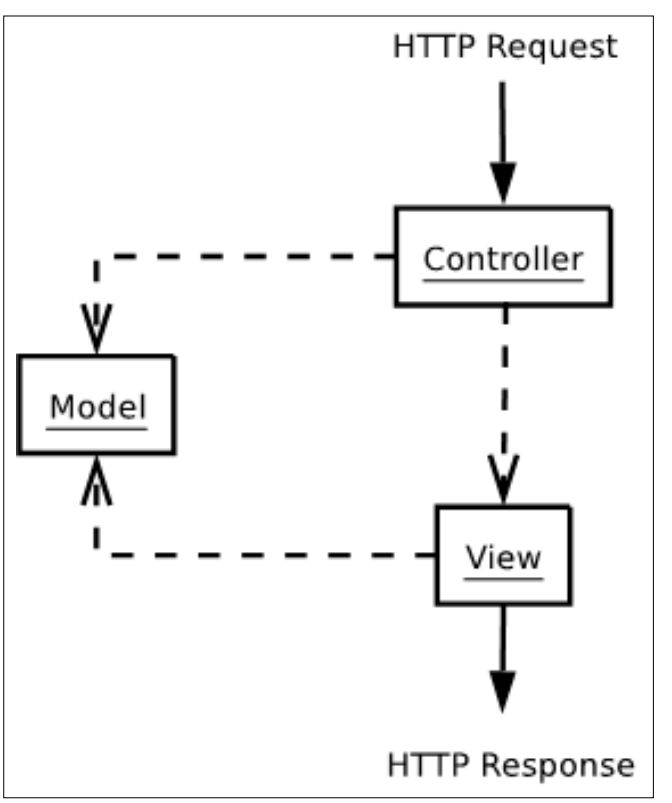


Figure 1. *Modèle MVC*

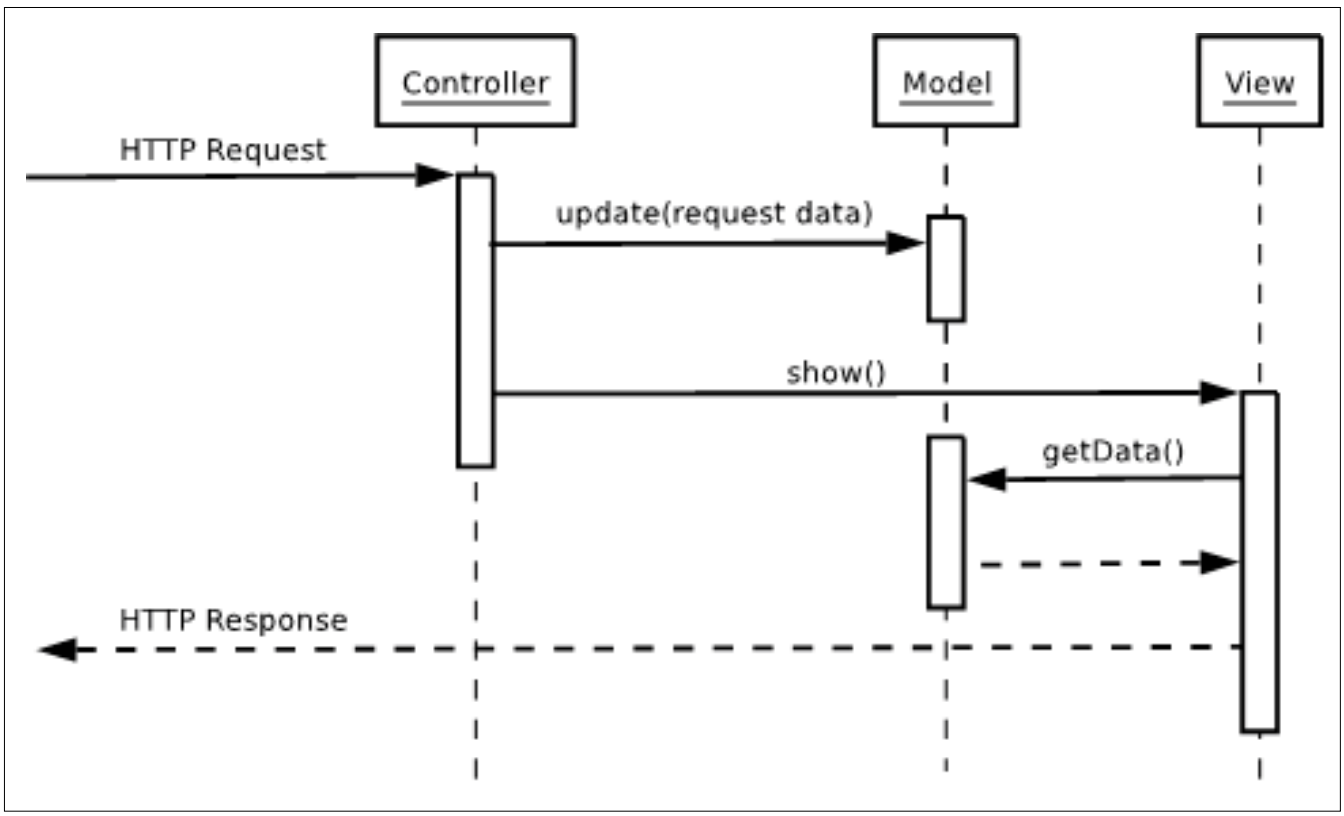


Figure 2. *Diagramme de séquence d'un modèle MVC*

- Les composants *Backend :* ils ne sont accessibles que par l'administrateur et servent à l'administration du site.

Il est possible de reconnaître les composants à partir de l'écriture de l'adresse URL :

Le lien *http://localhost/votre-site-Joomla/index.php?option=com_newsfeeds* fait appel au composant *Frontend com_newsfeeds*.

Le modèle MVC Joomla! est implémenté en utilisant trois classes fournies par le *Framework Joomla! : Jmodel, Jview et Jcontroller*.

Problématique

Nous voulons créer un module permettant de gérer des produits ; l'administrateur doit pouvoir ajouter, supprimer et modifier les produits et les utilisateurs auront la possibilité de consulter la liste des produits sans pouvoir les modifier. Nous allons donc concevoir un *Frontend* et un *Backend.*

Le Frontend

Comme il a été dit précédemment, le front est la partie visible aux visiteurs. Il est composé de plusieurs fichiers :

- */components/com_produit/frontend/produit.php ;*
- */components/com_produit/frontend/controller.php ;*
- */components/com_produit/frontend/views/produits/view.html.php ;*
- */components/com_produit/frontend/views/produits/tmpl/default.php ;*
- */components/com_produit/frontend/models/produit.php.*

Le point d'entrée : /frontend/produit.php

C'est le point d'entrée du composant, il doit obligatoirement porter le même nom que celui-ci. Ce fichier délègue les opérations effectuées par notre composant. C'est lui qui est exécuté en premier lorsqu'on entre l'adresse URL suivante :

http://localhost/votre-site-joomla/index.php?option=com_produits&view=produit. Examinons le contenu de ce fichier (voir Listing 1).

L'instruction `defined( '_JEXEC' ) or die( 'Accès restreint' );` vérifie si la demande d'exécution de ce fichier a été lancée par Joomla !. Si ce n'est pas le cas, la fonction `die( 'Accès restreint' )` interrompt automatiquement la demande.

La ligne `require_once( JPATH_COMPONENT.DS.'controller.php' );` désigne le chemin absolu à notre composant. Les variable `JPATH_COMPONENT_SITE` *ou* `JPATH_COMPONENT_ADMINISTRATOR` peuvent être utilisées si l'on veut accéder à des composants Frontend ou Backend. Le suffixe DS symbolise le séparateur de répertoire du système d'exploitation utilisé : / pour Windows et \ pour Unix. Ceci évite au développeur de créer des versions différentes pour chaque système d'exploitation.

Listing 1. *Code source du fichier /frontend/produit.php*

```
<?php

// Accès direct interdit
defined('_JEXEC') or die('Accès restreint');
// Chargement du contrôleur primaire de Joomla!
require_once (JPATH_COMPONENT.DS.'controller.php');
// Création d'un contrôleur spécifique
$controller = new ProduitController();
// Lecture de la tâche à réaliser
$controller->execute(JRequest::getCmd('task'));
// Redirection vers le contrôleur
$controller->redirect();
?>
```

Listing 2. *Code source du fichier /frontend/controller.php*

```
<?php

// Pas d'accès direct

defined( '_JEXEC' ) or die( 'Accès restreint' );
//importation de la classe JController
jimport('joomla.application.component.controller');

class ProduitController extends JController
{
    // affichage
    function display()
    {
        parent::display();
    }

}
```

Listing 3. *Code source du fichier /frontend/produits/view.html.php*

```
<?php

// Pas d'accès direct

defined( '_JEXEC' ) or die( 'Accès restreint' );
//importation de la classe JView
jimport( 'joomla.application.component.view');
  class ProduitViewProduit extends JView{
        //fonction affichage
    function display($tpl = null){
    //création d'une instance du modèle
      $model = &$this->getModel();
      //génération d'une liste des produits
      $rows = $model->getProduitList();
      $this->assignRef('rows' , $rows);
      parent::display($tpl);
    }
  }
?>
```

Listing 4. *Code source du fichier /frontend/produits/tmpl/default.php*

```
<?php

// Contrôle pour s'assurer que le fichier
//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');
?>

<h1><?php echo "Liste des produits"; ?></h1>
<ul>
<?php
// Lecture des enregistrements vers un tableau
foreach ($this->rows as $row) {
        ?>
        <li><?php echo $row->designation; ?></li>
        <li><?php echo $row->description; ?></li>
        <?php
}
?>
</ul>
```

`JRequest::getCmd( 'task' )` sert à vérifier si l'URL saisie contient des paramètres. Ces paramètres peuvent correspondre à une demande d'affichage, un ajout, une modification ou une suppression. `$controller->redirect();` permet de faire une redirection, généralement après une modification (ajout, suppression, changement de valeur).

Création du Contrôleur : /frontend/controller.php

Le contrôleur est le composant principal de l'architecture MVC. Il interagit avec le modèle, la vue ainsi qu'avec d'autres contrôleurs. Le contrôleur capte et analyse les requêtes de l'utilisateur ; il accède ensuite au modèle pour enfin présenter la vue. Examinons le Listing 2 pour analyser le code du contrôleur. La fonction `display()` est la fonction par défaut exécutée par le constructeur de la classe Jcontroller si aucune autre tâche n'est demandée. Cette fonction ne fait rien d'autre qu'appeler la fonction `display()` de la classe parente.

Création de la vue : /frontend/views/produits/view.html.php

La vue n'effectue aucun traitement ; sa tâche se résume à recevoir les actions de l'utilisateur et d'afficher les résultats renvoyés par le modèle. Les résultats d'un même modèle peuvent être affichés par plusieurs vues distinctes. Voyons un exemple de vue (Listing 3).

L'instruction `$model = &$this->getModel();` nous a permis de créer une instance du modèle nommée $model ; la méthode `getProduitList()` ($rows `$model->getProduitList();`) quant à elle, renvoie un tableau (*$rows*) qui contient les enregistrements des produits. Ce résultat est ensuite transmis au template que nous allons voir dans le paragraphe suivant.

Création du template : /frontend/views/produits/tmpl/default.php

Les *templates* sont des fichiers utilisés pour l'agencement des données en provenance de la vue. Nous allons construire un template simple contenant du code html, php (Listing 4).

Étant donné que la mise en forme est maintenant gérable de manière autonome depuis Joomla! 1.5, la présentation des templates peut être enrichie en utilisant des outils et des éditeurs appropriés comme *Dreamweaver, MooTools*.

Création du modèle : frontend/models/produit.php

Le modèle encapsule l'accès à la source de données. Il est totalement indépendant du reste; il ne connaît ni les contrôleurs, ni les vues car les liens vont des contrôleurs vers le modèle et parfois des vues vers le modèle mais jamais l'inverse. Jetons un coup d'œil sur le Listing 5 pour voir le contenu de ce fichier.

C'est la méthode `_getProduitQuery()` qui est chargée d'accéder aux données et c'est aussi elle qui contient la requête SQL. La méthode `getProduitList` se charge de contrôler l'accès et récupérer le résultat. Ce qui nous renvoie à la ligne `$rows = $model->getProduitList();` du code du fichier *views.html.php* (Listing 3).

Le Backend

Cette partie est réservée à l'administrateur du site ; c'est ici que s'opèrent l'ajout, la suppression, la modification et la publication des enregistrements. Plusieurs fichiers sont utilisés dans cette partie. En voici la liste :

- */components/com_produit/backend/admin.produit.php ;*
- */components/com_produit/backend/controller.php;*
- */components/com_produit/backend/controllers/produit.php;*
- */components/com_produit/backend/views/produits/view.html.php ;*
- */components/com_produit/backend/views/produits/tmpl/default.php ;*
- */components/com_produit/backend/views/produit/view.html.php ;*
- */components/com_produit/backend/views/produit/tmpl/form.php ;*
- */components/com_produit/backend/tables/produit.php ;*
- */components/com_produit/backend/install.sql ;*
- */components/com_produit/backend/uninstall.sql.*

Le point d'entrée : backend/admin.produit.php

Tout comme pour le *frontend*, il faut également créer un point d'entrée pour le backend. La seule différence entre ces deux points d'entrée est que celui du backend possède une condition qui vérifie s'il y a des contrôleurs spécifiques après le contrôleur de base. Examinons le contenu de ce fichier (Listing 6).

Le contrôleur primaire: backend/controller.php

Le code contrôleur primaire du backend est identique à celui du frontend. Nous allons donc faire une copie du contrôleur du frontend (Listing 2) étant donné qu'aucune modification n'intervient à ce niveau.

Contrôleur supplémentaire : backend/controllers/produit.php

Ce contrôleur contient plusieurs méthodes : *éditer, sauvegarder, supprimer, publier et annuler*. Chacune de ces méthodes instancie le modèle à travers l'instruction `$model = $this->getModel('produit');` . Les classes `JText` et `JError` interagissent avec l'utilisateur pour le renseigner sur le déroulement des opérations (Listing 7).

Listing 5. *Code source du fichier /frontend/models/produit.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');

//Importation du modèle JModel
jimport('joomla.application.component.model');

class ProduitModelProduit extends JModel
{
        //Préparation de la requête (sélection des
produits publiés)
        function _getProduitQuery( &$options )
        {

                $db                     =
JFactory::getDBO();
                $id                     =
@$options['id'];

                $select = 'p.*';
                $from    = '#__produit AS p';

                $wheres[] = 'p.publication = 1';

                $query = "SELECT " . $select .
                                "\n FROM " . $from
.
                                "\n WHERE " .
implode( "\n  AND ", $wheres );

                return $query;
        }
        //Affichage des résultats (liste des produits
publiés)
        function getProduitList( $options=array() )
        {
                $query  = $this->_getProduitQuery(
$options );
                $result = $this->_getList( $query );
                return @$result;
        }
}
?>
```

Listing 6. *Code source du fichier backend/admin.produit.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');

//Importation du contrôleur de base

require_once (JPATH_COMPONENT.DS.'controller.php');

// Importation contrôleur spécifique (sur demande)

if($controller = JRequest::getVar('controller')) {
        require_once (JPATH_COMPONENT.
DS.'controllers'.DS.$controller.'.php');
}

// Création du contrôleur
$classname      = 'ProduitsController'.$controller;
$controller = new $classname( );

// Exécution de la tâche demandée
$controller->execute(JRequest::getCmd('task'));

// Redirection
$controller->redirect();

?>
```

Listing 7. *Code source du fichier backend/controllers/produit.php*

```
<?php

// Contrôle pour s'assurer que le fichier
//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');

class ProduitsControllerProduit extends
ProduitsController
{
        //Constructeur
        function __construct()
        {
                parent::__construct();

                // Recensement des tâches
                $this->registerTask( 'add'  ,
'edit' );
                $this->registerTask( 'unpublish',
'publish');
        }

        //Affichage du formulaire d'édition
        function edit()
        {
                JRequest::setVar( 'view', 'produit' );
                JRequest::setVar( 'layout', 'form'  );
                JRequest::setVar('hidemainmenu', 1);

                parent::display();
        }

        //Enregistrement
        function save()
        {
                $model = $this->getModel('produit');

                if ($model->store($post)) {
                        $msg = JText::_( 'Produit
enregistré!' );
                } else {
                        $msg = JText::_( 'Erreur lors
de l enregistrement du produit' );
                }

                $link = 'index.php?option=com_
produits';
                $this->setRedirect($link, $msg);
        }

        //Suppression
        function remove()
        {
                $model = $this->getModel('produit');
                if(!$model->delete()) {
                        $msg = JText::_( 'Erreur:
suppression impossible' );
                } else {
                        $msg = JText::_( 'Produit(s)
supprimé(s)' );
                }

                $this->setRedirect( 'index.
php?option=com_produits', $msg );
        }

        //Publications
        function publish()
        {
                $this->setRedirect( 'index.
php?option=com_produits' );

                // Initialisation des variables
                $db                     =& JFactory::
getDBO();
                $user           =& JFactory::
getUser();
                $cid            = JRequest::getVar(
'cid', array(), 'post', 'array' );
                $task           = JRequest::getCmd(
'task' );
                $publish = ($task == 'publish');
                $n                      = count( $cid
);

                if (empty( $cid )) {
                        return JError::raiseWarning(
500, JText::_( 'Aucun produit séléctionné' ) );
                }

                JArrayHelper::toInteger( $cid );
                $cids = implode( ',', $cid );

                $query = 'UPDATE #__produit'
                . ' SET published = ' . (int) $publish
                . ' WHERE id IN ( '. $cids .' )'
                ;
                $db->setQuery( $query );
                if (!$db->query()) {
                        return JError::raiseWarning(
500, $row->getError() );
                }
                $this->setMessage( JText::sprintf(
$publish ? 'Produit publié' : 'Produit non publié', $n )
);

        }

        //Annulation
        function cancel()
        {
                $msg = JText::_( 'Operation annulée'
);
                $this->setRedirect( 'index.
php?option=com_produits', $msg );
        }

}

?>
```

Listing 8. *Code source du fichier backend/views/produits/view.html.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');

//Importation de la vue JView
jimport( 'joomla.application.component.view' );

class ProduitsViewProduits extends JView
{
        //Affichage des boutons standards de Joomla!
        function display($tpl = null)
        {
                JToolBarHelper::title(   JText::_(
'Gestionnaire des produits' ), 'generic.png' );
                JToolBarHelper::publishList();
                JToolBarHelper::unpublishList();
                JToolBarHelper::deleteList();
                JToolBarHelper::editListX();
                JToolBarHelper::addNewX();

                // Récupération des données en
provenance du modèle
                $items          = & $this->get(
'Data');

                $this->assignRef('items',
$items);

                parent::display($tpl);

        }
}
```

HOSTING NEXT LEVEL

IPv6 RUNNING

intel CORE i7 inside

HETZNER DEDICATED ROOT SERVER EQ 4

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

42,- € par mois

HETZNER DEDICATED ROOT SERVER EQ 6

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

58,- € par mois

HETZNER DEDICATED ROOT SERVER EQ 8

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

75,- € par mois

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

2 En tant que nouveau client, vous pouvez bénéficier de 8 € de réduction sur n'importe quel produit présenté ici. Utilisez, s'il vous plait, le code promo **011909** en remplissant votre bon de commande. Cette offre expire le 01.10.2010.

Tous les tarifs s'entendent HT et sont soumis aux conditions de vente de Hetzner Online AG. Les prix sont susceptibles d'être changés. Tous droits réservés par les fabricants respectifs.

Intel, le logo Intel, Intel Core et Core Inside sont des marques déposées par Intel Corporation aux U.S.A. et dans d'autres pays.

HETZNER ONLINE

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

www.hetzner.info

info@hetzner.com

Listing 9. *Code source du fichier backend/views/produits/tmpl/default.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!

defined('_JEXEC') or die('Accès restreint');
 ?>

<form action="index.php" method="post"
name="adminForm">

<div id="editcell">
        <table class="adminlist">
        <thead>
                <tr>
                <tr>
                        <th width="5">
                                <?php echo JText::_(
'NUM' ); ?>
                        </th>
                        <th width="20">
                                <input
type="checkbox" name="toggle" value=""
onclick="checkAll(<?php echo count( $this->items ); ?>);"
/>

                        </th>
                        <th  class="title">
                                <?php echo
JHTML::_('grid.sort',   'Produit', 'a.description',
@$lists['order_Dir'], @$this->lists['order'] ); ?>
                        </th>
                        <th width="5%" align="center">
                                <?php echo JHTML::_
('grid.sort',   'Publication', 'a.publication', @$this-
>lists['order_Dir'], @$this->lists['order'] ); ?>
                        </th>
                        <th width="1%"
nowrap="nowrap">
                                <?php echo JHTML::_
('grid.sort',   'ID', 'a.id', @$this->lists['order_Dir'],
@$this->lists['order'] ); ?>
                        </th>
                </tr>

        </thead>
        <?php
        $k = 0;
        for ($i=0, $n=count( $this->items ); $i < $n;
$i++)
        {
                $row = &$this->items[$i];

                $published                = JHTML::_
('grid.publication', $row, $i );
                $checked          = JHTML::_('grid.id',
$i, $row->id );
                $link             = JRoute::_( 'index.
php?option=com_produits&controller=produit&task=edit&c
id[]='. $row->id );

                ?>
                <tr class="<?php echo "row$k"; ?>">
                        <td>
                                <?php echo $row->id;
?>
                        </td>
                        <td>
                                <?php echo $checked;
?>
                        </td>
                        <td>
                                <a href="<?php echo
$link; ?>"><?php echo $row->description; ?></a>
                        </td>
                                                </
td>
                        <td align="center">
                                <?php echo
$published;?>
                        </td>
                        <td align="center">
                                <?php echo $row->id;
?>
                        </td>
                </tr>
                <?php
                $k = 1 - $k;
        }

        ?>

        </table>

</div>

<input type="hidden" name="option" value="com_
produits" />

<input type="hidden" name="task" value="" />

<input type="hidden" name="boxchecked" value="0" />

<input type="hidden" name="controller" value="produit"
/>

</form>
```

Listing 10. *Code source du fichier /backend/views/produit/view.html.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!

defined('_JEXEC') or die('Accès restreint');

//Importation de la vue JView

jimport( 'joomla.application.component.view' );

class ProduitsViewProduit extends JView
{
        //Affichage
        function display($tpl = null)
        {
                $produit          =& $this->get('Data');
                $isNew            = ($produit->id < 1);

                $text = $isNew ? JText::_( 'New' ) :
JText::_( 'Edit' );
                JToolBarHelper::title(   JText::_(
'Produit' ).': <small>[ ' . $text.' ]</small>' );
                JToolBarHelper::save();
                // La variable isNew détermine si le
produit existe ou pas
                // Si le produit n'existe pas dans la
BD, il sera créé
                if ($isNew)  {
                        JToolBarHelper::cancel();
                } else {
                        // Pour les produits existant
le bouton 'cancel' sera remplacé par le bouton 'close'
                        JToolBarHelper::cancel(
'cancel', 'Close' );
                }

        //La méthode assignRef() permet de transmettre
le contenu de la variable et son nom au template.
                $this->assignRef('produit',
$produit);

                parent::display($tpl);

        }
}
```

Listing 11. *Code source du fichier /backend/views/produit/tmpl/form.php*

```
<?php defined('_JEXEC') or die('Restricted access'); ?>

<script language="javascript" type="text/javascript">
        function submitbutton(pressbutton) {
                var form = document.adminForm;
                if (pressbutton == 'cancel') {
                        submitform( pressbutton );
                        return;
                }
                // Validation des champs
                if (form.text.value == "") {
                        alert( "<?php echo JText::_( 'Ce champ doit être renseigné', true ); ?>" );
                } else {
                        submitform( pressbutton );
                }
        }
</script>

<form action="index.php" method="post" name="adminForm" id="adminForm">
<div>
        <fieldset class="adminform">
        <legend><?php echo JText::_( 'Details' ); ?></legend>
        <table class="admintable">
                <tr>
                        <td width="110" class="key">
                                <label for="title">
                                        <?php echo JText::_( 'Désignation' ); ?>:
                                </label>
                        </td>
                        <td>
                                <input class="inputbox" type="text" name="designation" id="designation" size="60" value="<?php echo $this->produit->designation; ?>" />
                        </td>
                </tr>
                <tr>
                        <td width="110" class="key">
                                <label for="alias">
                                        <?php echo JText::_( 'Description' ); ?>:
                                </label>
                        </td>
                        <td>
                                <input class="inputbox" type="text" name="description" id="description" size="60" value="<?php echo $this->produit->description; ?>" />
                        </td>
                </tr>

                <tr>
                        <td width="120" class="key">
                                <?php echo JText::_( 'Publication' ); ?>:
                        </td>
                        <td>
                                <?php echo JHTML::_( 'select.booleanlist',  'publication', 'class="inputbox"', $this->produit->publication ); ?>

                        </td>

                </tr>
        </table>

        </fieldset>
</div>
<div class="clr"></div>
<div class="clr"></div>

<input type="hidden" name="option" value="com_produits" />

<input type="hidden" name="id" value="<?php echo $this->produit->id; ?>" />

<input type="hidden" name="task" value="" />

<input type="hidden" name="controller" value="produit" />

</form>
```

Listing 12. *Code source du fichier /backend/tables/produit.php*

```
<?php

// Contrôle pour s'assurer que le fichier

//est inclus dans Joomla!
defined('_JEXEC') or die('Accès restreint');

class TableProduit extends JTable
{
        //Initialisation des variables
        /** @var int Primary key */
        var $id                                 = 0;
        /** @var string */
        var $description                        = '';
        /** @var string */
        var $designation                = '';
        /** @var string */
        var $publication                = 0;
        /** @var int */

        // Mapping objet relationnel
        function TableProduit(& $db) {
                parent::__construct('#__produit', 'id', $db);
        }
}

?>
```

Listing 13. *Code source du fichier backend/install.sql*

```
DROP TABLE IF EXISTS '#__produit';
CREATE TABLE '#__produit' (
  'id' int(11) NOT NULL auto_increment,
  'description' text,
  'designation' varchar(100)  NOT NULL,
  'publication' tinyint(1) NOT NULL,
  PRIMARY KEY  ('id')
);

INSERT INTO #__produit VALUES (2, 'Telephone', 'SmartPhone', 1);

INSERT INTO #__produit VALUES (4, 'Ordinateur', 'MacBook', 1);
```

Listing 14. *Code source du fichier backend/uninstall.sql*

```
DROP TABLE IF EXISTS '#__produit';
```

La vue : backend/views/produits/view.html.php

Cette vue concerne la barre d'outils administrative ; elle fait intervenir la classe `JtoolBarHelper` chargée d'afficher les contrôles administratifs tels que : *publier, dépublier, éditer, supprimer, nouveau*. Analysons le code source de ce fichier (Listing 8).

Le template : backend/views/produits/tmpl/default.php

Ce template se charge de la mise en forme de l'affichage de la liste de nos produits ; il contient du code html, php et Javascript (Listing 9).

Listing 15. *Code source du fichier /backend/produit.xml*

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

<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">
<install type="component" version="1.5.0">

        <!-- Informations sur le composant -->
        <name>Gestionnaire de Produits</name>
        <creationDate>16/08/2010</creationDate>
        <author>Admin</author>
        <authorEmail>admin@admin.com</authorEmail>
        <authorUrl>www.admin.com</authorUrl>
        <copyright>Copyright</copyright>
        <license>Open source</license>
        <version>1.0.0</version>
        <description>Composant de gestion des produits</description>

        <!-- Structure des dossiers et des fichiers de la partie frontend-->
        <files folder="frontend">
                <filename>index.html</filename>
                <filename>produit.php</filename>
                <filename>controller.php</filename>
                <filename>views/index.html</filename>
                <filename>views/produit/index.html</filename>
                <filename>views/produit/view.html.php</filename>
                <filename>views/produit/tmpl/index.html</filename>
                <filename>views/produit/tmpl/default.php</filename>
                <filename>models/produit.php</filename>
        </files>

        <!-- Informations sur la création de la table du composant lors de l'installation-->
        <install>
                <sql>
                        <file charset="utf8" driver="mysql">install.sql</file>
                </sql>
        </install>

        <!-- Informations sur la suppréssion de la table du composant lors de la désinstallation-->
        <uninstall>
                <sql>
                        <file charset="utf8" driver="mysql">uninstall.sql</file>
                </sql>
        </uninstall>

        <!-- Structure des dossiers et des fichiers de la partie frontend-->
        <administration>
                <menu>Produit</menu>
                <files folder="backend">
                        <filename>index.html</filename>
                        <filename>admin.produit.php</filename>
                        <filename>controller.php</filename>
                        <filename>controllers/produit.php</filename>
                        <filename>controllers/index.html</filename>
                        <filename>models/produit.php</filename>
                        <filename>models/produits.php</filename>
                        <filename>models/index.html</filename>
                        <filename>views/produits/view.html.php</filename>
                        <filename>views/produits/index.html</filename>
                        <filename>views/produits/tmpl/default.php</filename>
                        <filename>views/produits/tmpl/index.html</filename>
                        <filename>views/produit/view.html.php</filename>
                        <filename>views/produit/tmpl/form.php</filename>
                        <filename>views/produit/index.html</filename>
                        <filename>views/produit/tmpl/index.html</filename>
                        <filename>tables/produit.php</filename>
                        <filename>tables/index.html</filename>
                        <filename>install.sql</filename>
                        <filename>uninstall.sql</filename>
                </files>
        </administration>
</install>
```

La vue formulaire : /backend/views/produit/view.html.php

Celle-ci concerne l'affichage de la fiche d'un produit ; nous allons également construire une barre d'outils pour cette vue afin de donner à la possibilité à l'administrateur de pouvoir modifier le produit sélectionné (Listing 10).

Le formulaire du template : /backend/views/produit/tmpl/form.php

Ce formulaire permet de modifier le produit affiché par la vue /backend/produit/view.html.php. Consultons le contenu de son code source (Listing 11).

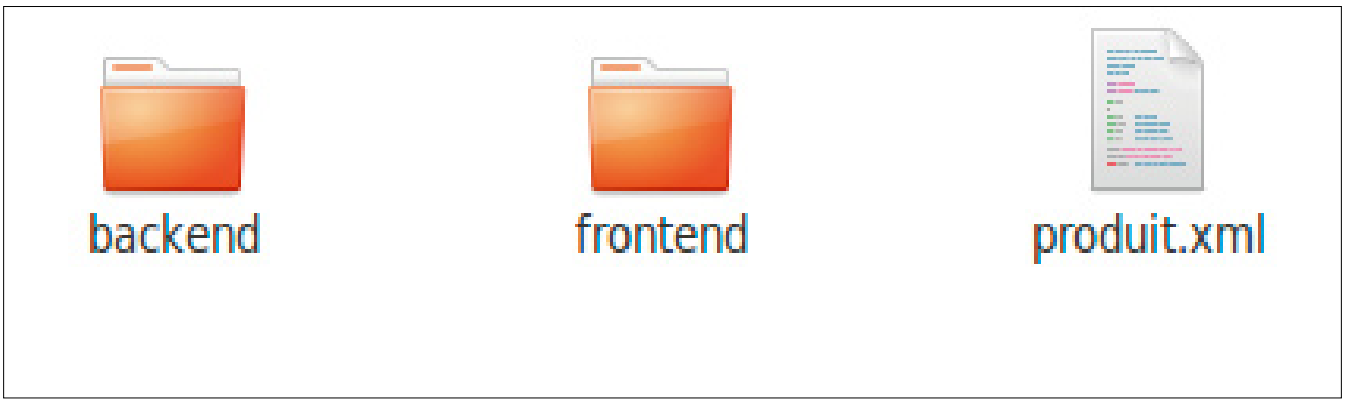


Figure 3. Structure du paquetage final

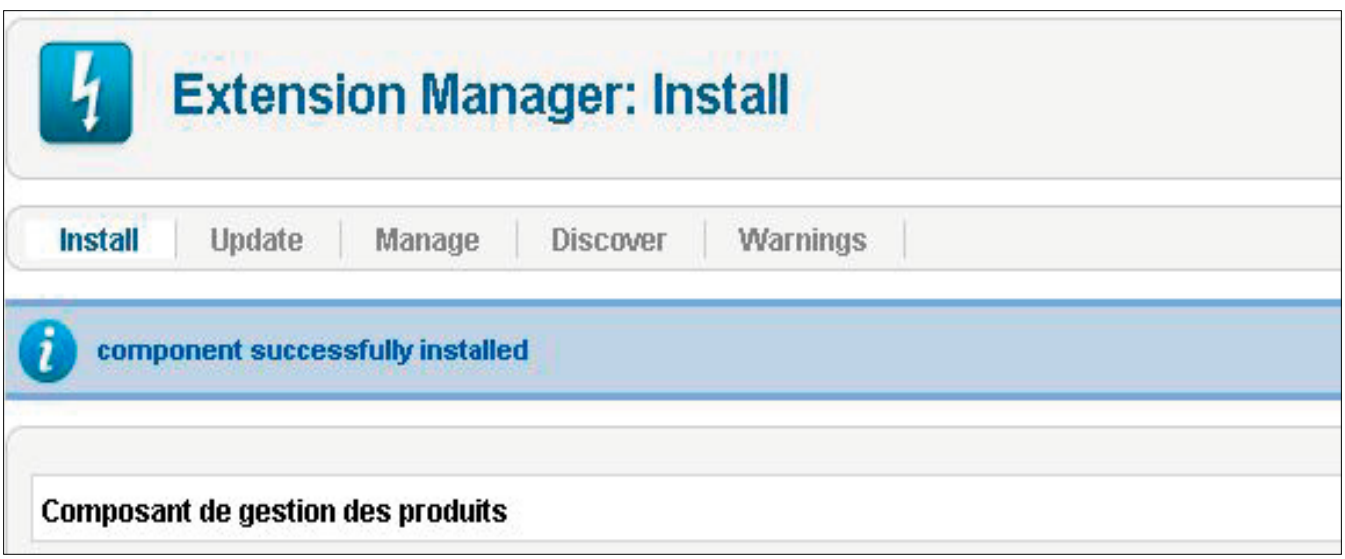


Figure 4. *Installation réussie*

Figure 5. *Composant produit*

La classe produit : /backend/tables/produit.php

Cette classe indique au modèle les données qu'il doit manipuler. Elle implémente l'interface JTable qui lui confère la simplicité d'accéder aux données et d'utiliser ses méthodes. Le nom de la table ainsi que la clé primaire sont fixés dans le constructeur de la classe (Listing 12).

Le fichier d'installation : backend/install.sql

Ce fichier contient des instructions SQL afin de donner à Joomla! les informations nécessaires à la création de la table du composant au moment de l'installation (Listing 13).

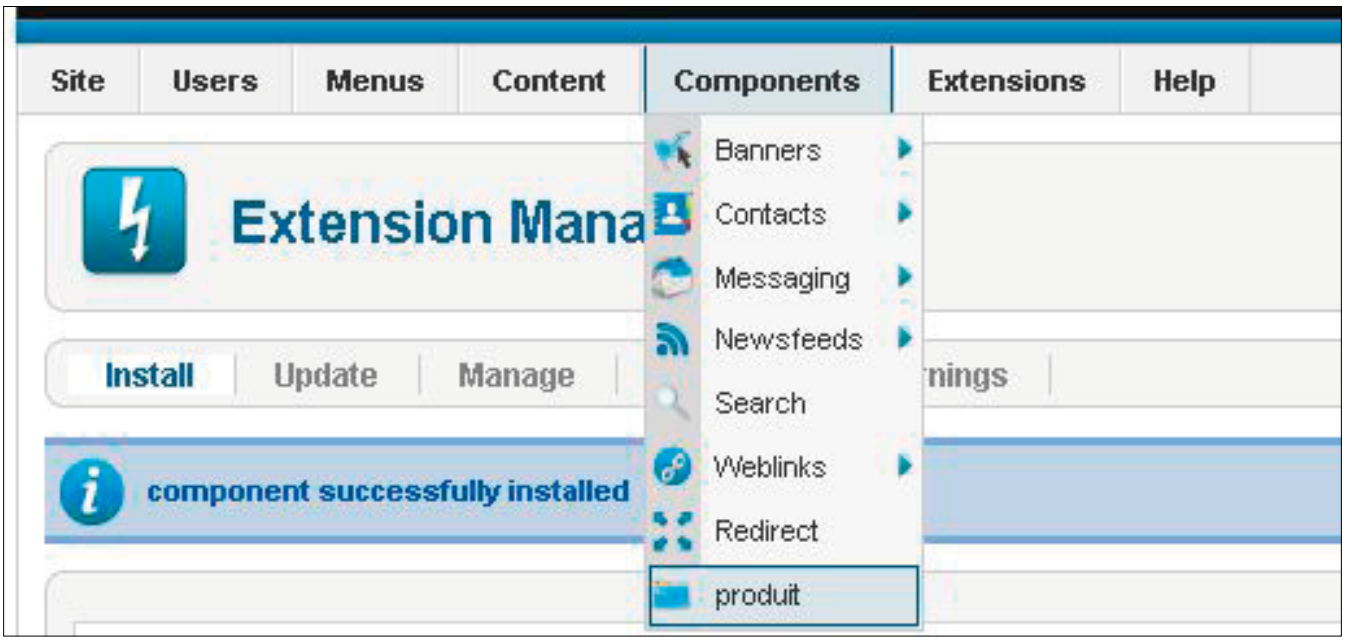


Figure 6. *Menu composant*

Le fichier de désinstallation : backend/uninstall.sql

Celui-ci intervient à la désinstallation du composant pour la suppression de la table du composant; il ne contient qu'une seule instruction comme nous allons le voir dans le Listing 14.

Le fichier de configuration : backend/produit.xml

C'est ce fichier qui décrit le nouveau composant ; les informations contenues dans le fichier de configuration seront utilisées durant l'installation. Il doit contenir les noms et les emplacements de tous les fichiers qui constituent le composant, c'est grâce à ça que Joomla! parvient à créer la table dans la base de données, les sous-dossiers, y placer les fichiers (Listing 15).

Paquetage d'installation

C'est la dernière étape du travail ; nous allons assembler les différentes parties de notre composant au sein d'un même dossier que nous nommerons : *com_produit*. Notre paquetage final aura une structure identique à celle présentée sur la Figure 3. Nous allons compresser notre dossier *com_produit* afin d'obtenir un dossier compressé : *com_produit.zip* qui servira à l'installation.

Après l'installation du composant , un message de confirmation du bon déroulement de l'opération vous sera affiché (Figure 4). Vous pourrez vous rendre dans le gestionnaire des extensions pour voir si notre composant est bien présent (Figure 5) ou jeter un coup d'œil dans le menu composant (Figure 6).

Environnement de developpement

Cet exemple a été réalisé dans un environnement constitué de :

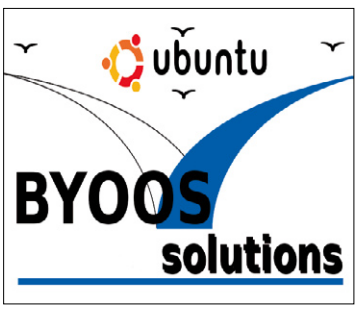
- Joomla! 1.6 Beta ;
- XAMPP for Linux 1.7.3a ;
- Ubuntu Lucid 10 LTS ;
- Firefox.

Sur Internet

- *http://api.joomla.org/Joomla-Framework/Application/JController.html* – API Joomla!,
- *http://extensions.joomla.org/extensions/miscellaneous/development/1509/details – Projet communautaire Joomla !, site communautaire des développeurs Joomla!,*
- *http://cocoate.com/fr/node/1870/conception-de-composants-de-modules-et-de-plugins/un-exemple-complet-auto* – Un exemple sur la conception de composant, de module et de plugin.

DONY CHIQUEL

Dony Chiquel est un ingénieur développeur spécialisé en technologies J2EE, .NET et en conception objet avec UML. Il travaille actuellement en tant que consultant en nouvelles technologies pour le compte de BUGOVER et PROSOLUCE.



BYOOS solutions

partenaire du développement DURABLE !

Le logiciel OPEN SOURCE DJAFOREST au service de la protection de l'environnement.

HISTORIQUE

DJAFOREST est un applicatif WEB de travail collaboratif qui vise à montrer les méfaits de la surconsommation du papier à l'heure où l'écologie et la protection de l'environnement sont des faits marquants de notre société moderne. Au sein du groupe de développement nous avons décidé d'offrir une application PHP qui, tout en étant orientée *développeur* sera simple à mettre en œuvre et fera la promotion de la forêt équatoriale du DJA située au sud du Cameroun, Afrique Central.

Cet article aussi modeste soit-il doit montrer les possibilités qu'offre le langage PHP dans les projets de développement RAD (Rapid – Application – Development) développement rapide d'application en ouvrant le choix des librairies aux applicatifs codés en PHP ayant fait leurs preuves. Les études ont commencé en Février 2010 puis après avoir collecté et testé moult librairies, notre développement a débuté fin juin 2010. Le projet sera disponible début Septembre 2010 afin que chaque codeur PHP intéressé puisse intégrer le projet en le dotant de tous les outils nécessaires à sa publication en version 1.0. Par exemple : blog, forum, download, e-commerce, GED, agenda, facturation, fichier clients…

Le choix de codeigniter et la P.O.O (Programmation Orientée Objet)

CODEIGNITER (CI en abrégé) ou le chemin du futur… Je me permets cette affirmation car après plusieurs mois passés en veille technologique à la recherche d'un FrameWork PHP pour nos projets BYOOS solutions, je suis arrivé au triste constat qu'il n'y avait pas sur le marché un applicatif qui répondait à nos critères. Nous avons été séduis par la simplicité de mise en œuvre de CI, la possibilité de coder en PHP4 et/ou PHP5 puis par la puissance de ses librairies, sa riche documentation et depuis peu ses ressources en Français. Le modèle conceptuel de CI est conforme à l'application MVC (Models, Views, Controlers), la racine de votre projet peut comporter à minima 3 répertoires et être pleinement fonctionnelle.

www-root
system
application
modules

chaque module ou application est découpé en fonction :

config
controlers
errors
helpers
hooks
language
libraries
models
views

ce qui permet une lecture claire des scripts de contrôle puis de retrouver simplement les variables d'affichage au niveau des vues (*./views/*).

Cet article n'est pas un cours sur la mise en œuvre de CI mais vous montre la clarté du modèle MVC qui facilitera d'autant la maintenance des applications, pour plus d'informations allez visiter le très bon site de CI France à l'adresse *http://www.codeigniter.fr.*

Le projet DJAFOREST est encore au stade de la version bêta mais déjà utilisable avec son installateur automatique venu de PyroCMS et permet de coder un site WEB multi-pages, multi-news. Le Back Office SiteManagr est doté de l'éditeur Javascript Ckeditor ce qui offre une grande souplesse dans la composition des pages et articles. Les éléments complémentaires comme les images, plugins et upload Photos sont localisés dans le répertoire */assets/*.

LA FEUILLE DE ROUTE de DjaForest par NDI TANYU Jones stagiaire byoos.fr

Le projet de **DjaForest** est une application informatique de type Web ciblé montrant au public le riche héritage naturel de la réserve de la forêt du Dja dans la région du sud de la République du Cameroun en particulier, des forêts équatoriales en général et également d'autre part

à décourager la population globale à abattre les arbres de nos forêts.

On avait pensé que l'arrivée de l'ordinateur réduirait considérablement la quantité de copies papiers reproduites par document. L'inverse s'est cependant avéré être vrai, autant de documents journaliers sont imprimés. L'impression signifie que l'usage des papiers et de leur production est remercié grâce à l'abattage des arbres. Quand un arbre est coupé, il va vers le bas avec les autres et dans certains cas l'écosystème est considérablement affecté. Ceci crée parfois les effets négatifs qui vont jusqu'à affecter la vie de l'habitant des zones de forêt comme les gens de Baka. Ceci et les activités semblables ne peuvent plus continuer en tant que tel sans être contrôlées et sont basées sur le proto djaforest qui a été conçu, dans son propre chemin de manière à sensibiliser le public sur les dangers du déboisement.

DÉFINITION DES BESOINS

DjaForest est une application Web et par conséquent en la mettant en place, son utilisation ne sera seulement rendue possible que si les conditions suivantes sont remplies : ordinateurs (PC pour les utilisateurs et un serveur pour accueillir l'application) et une connexion Internet ou Intranet

MISE EN APPLICATION

Un certain nombre de ressources sont nécessaires pour l'exécution de **DjaForest**. Les logiciels CodeIgniter, Matchbox, Sitemanagr et PyroCMS installer frameworks servent de base au codage à venir. Le framework sitemanagr est employé pour la zone d'administration et Pyrocms installer permet l'installation du système ; CodeIgniter permet de créer des thèmes d'affichage puis Matchbox appliqué à la gestion modulaire.

Mr. Gabriel Bobard Directeur de BYOOS Solutions a mis le CMS en place par intégration du framework CodeIgniter permettant à tout développeur Web d'avoir une plate-forme ou une base de travail sur laquelle il peut coder. Ce CMS offrira une page, un menu et une gestion de modules. Il a également été préparé pour assurer une excellente réécriture URL (url rewriting). Pour la gestion de modules Matchbox, cela signifie que beaucoup d'autres applications peuvent être développées et ajoutées au même système (galerie images, Facturation Invoice, module de e-commerce …). Le Back Office sera chargé de la gestion de l'administration et l'installation du programme. Il est prévu dans la partie de Ndi Jones, étudiant de l'internat de Byoos de construire le Front Office avant et de développer des modules (par exemple blog et forums ...) pour l'application.

La conception du Front Office est basée sur le modèle du Web 2.0. Ceci signifie que les boutons seront au format du Web 2.0, le choix de couleurs pour le thème ou du modèle est réduit aux quelques couleurs de base. *Couleurs pétantes* fortes à éviter autant que possible. La largeur des pages à considérer doit être moindre que celle de l'écran de l'ordinateur.

Parler sur la façon dont ceci sera réalisé est également très important. Premièrement les langages de programmation à employer sont :

- PHP (script du côté serveur) .
- HTML, CSS (codage du Front Office ou du côté client) .
- Javascript (codage dynamique de côté client).

Il est également intéressant de mentionner les logiciels à employer qui sont :

Pour Windows (c)
- notepad++
- wamp server
- navigateurs Web (Firefox, opera, Google chrome, Netscape et Explorer)
- photoshop (pour le traitement des images)
- boutons libres

Pour Linux ubuntu 10.04
- geany éditeur
- apache2
- rapidSVN
- Gimp

Avec ces ressources, les codes sont dactylographiés en utilisant les éditeurs de texte notepad++ ou Geany LINUX et exécuté sur les serveurs WAMPserver ou Apache 2 en utilisant les navigateurs (FireFox, Opera, Chrome, IE …) ceci permet une meilleure appréciation de la présentation de l'application sur le Web.

Auteur Gabriel BOBARD
I.T Manager BYOOS solutions
contact@byoos.fr
http://www.djaforest.com
http://www.byoos.fr
+237 22 06 58 13 (Cameroun)
+33 645 25 48 13 (France)

ANNEXE 1

application dja forest 08/02/2010 version 0.1.0

Objectifs

Fournir une application informatique NTIC visant à réduire la consommation de papier et par là même limiter la coupe de FORÊTS. Permettre au Directeur Informatique (D.S.I) de dotter les services Administratifs ou Opérationnels d'un couteau SUISSE virtuel.

1) framework
2) le back office
3) la gestion des modules

4) la gestion des templates
5) le front office

Développeur(s)
KOUAMOU Franck
Ndi TANYU Jones
BOBARD Gabriel

Gestion du projet
BOBARD Doris

Traducteur(s)
MANCEAU Jacques

Promoteur
BYOOS solutions *http://byoos.fr* *http://djaforest.com* *contact@byoos.fr*

Ressources

Codeigniter 1.7.2	*http://codeigniter.fr*
Sitemanager 0.34	*http://designelemental.net/sitemanagr*
Matchbox 1.0RC2	*http://codeigniter.com/wiki/Matchbox/*
Templates 1.4.1	*http://williamsconcepts.com/ci/codeigniter/libraries/template/reference.html*
CKeditor 3.1	*http://ckeditor.com*
PyroCMS	*http://pyrocms.com*

INSTALL
décompresser l'archive dans un répertoire de votre site *www-root/djaforest*,
placer les droits d'accès aux fichiers, généralement 644
activer le module apache mod_rewrite.c (url rewriting),
lancer l'application.

1) Si le logiciel DjaForest n'est pas encore installé sur votre machine, le programme d'installation va alors se lancer automatiquement: *www-root/djaforest/installer*,

   entrez le nom serveur, généralement 'localhost',
   entrez le nom de l'administrateur de la bases de données,
   entrez le mot de passe de l'administrateur de la base de données et VALIDER.

   A ce stade la création de la base de données se fait de façon automatique. Ensuite entrez votre nom d'utilisateur puis votre email courant et votre mot de passe VALIDEZ.

   entrez le nom d'utilisateur 'admin' puis le mot de passe 'admin'.

2) Si l'application est déjà installée, le controleur principal 'Pages' prend le contrôle du code,
   documentation complète à venir.

3) Toutes les ressources qui ont servi au développement sont placées dans le répertoire www-root/djaforest/ressources.

10/08/2010 0.1.1b15
intégration de la librairie MATCHBOX rc2 au projet SiteManagr,
intégration de l'installateur ionizecms0.94,
modification de la location des bibliotèques externes, javascript, flash, css, images ./assets.

16/08/2010
intégration du moteur de templates codigniter 1.7.x,
supprimer les lignes de codes siteManager faisant appel aux lib de Google et rendant l'application inutilisable hors ligne (???),
lib concernées , prototype.js, scriptalous.js.

22/08/2010 version 0.1.1b23
intégration de CKeditor aux NEWS et PAGES afin de permettre une mise en page aisée,
intégration des liens dans la région 'sidebar'.

24/08/2010 version 0.1.2b25
intégration du lecteur multimédia JW MediaPlayer,
ajout de la librairie flvplayer,
suppression de l'installateur de ionize-094 et remplace par l'installer de PyroCMS (nettement plus simple à intégrer !!!).

Formation WEBMASTER - LINUX en e-learning sur la plateforme de *elearning.byoos.fr* PHP5 programmation P.O.O & administration LINUX Debian - Ubuntu


BOBARD Gabriel Julien Pierre
I.T Manager BYOOS

PRÉPAREZ VOTRE RENTRÉE ! EN PUISSANCE ET À PETIT PRIX

LA GAMME **CRAZY FISH**

LES SERVEURS DÉDIÉS MALINS À PRIX IKOULA

GREEN FISH

- Atom Dual Core 510
- 2x2 Go DDR2
- 250 Go
- BP 100 Mbs full duplex

29,99€ HT/MOIS

SMART FISH

- AMD opteron Eight Core
- 16 Go RAM DDR3
- 2x750 Go SATA
- BP 100 Mbs full duplex

189,00€ HT/MOIS

MEGA FISH

- 4 Xeon 7550 8 coeurs
- 128 Go DDR3
- 16x500 Go sas Raid Hard
- BP 100 Mbs full duplex

SUR DEVIS

DÉCOUVREZ LA NOUVELLE GAMME DE SERVEURS DÉDIÉS DONT

IKOULA COREi3

DELL

- CORE i3 530
- 2x2 Go DDR3
- 2x160 Go sata
- BP 100 Mbs full duplex

79,99€ HT/MOIS

IKOULA 3430

DELL

- Xéon 3430
- 4x2 Go DDR3
- 2x500 Go sata raid1
- BP 100 Mbs full duplex

99,00€ HT/MOIS

IKOULA 3440

DELL

- Xéon 3440 HT
- 4x2 Go DDR3
- 2x1 To sata raid1
- BP 100 Mbs full duplex

139,99€ HT/MOIS

La gamme complète sur notre site web www.ikoula.com

Pour tout engagement de 36 mois, bénéficiez de 15% de remise sur ces tarifs

ikoula Hosting Services

0 890 710 712

WWW.IKOULA.COM

HEBERGEMENT depuis 12ANS HAUTE QUALITÉ

NOM DE DOMAINE | EMAIL | HÉBERGEMENT | CERTIFICATS SSL | SERVEURS PRIVÉS | SERVEUR DÉDIÉS

Sécurité des sessions PHP

Les sessions reposent sur un jeton unique pour reconnaître l'utilisateur au cours de sa navigation. Ceci expose les applications à des attaques où le pirate usurpe l'identité d'un utilisateur légitime afin d'accéder au système d'information. Dans cet article vous apprendrez à sécuriser les sessions.

Cet article explique :

- Les vulnérabilités liées à l'utilisation de sessions.
- Comment protéger les applications reposant sur des sessions.

Ce qu'il faut savoir :

- Vous devez connaître les bases du langage PHP, les cookies et les sessions.

Les applications web sont la cible d'attaques diverses. Celles-ci visent le système d'information ou la prise de contrôle du serveur hébergeant l'application afin d'attaquer d'autres sites ou d'installer des services. Les attaques d'un système d'information attentent à :

- l'intégrité des données, la personne malveillante modifie les données publiées sur un site web (défiguration) ou modifie des informations de l'application ;
- la confidentialité des données, le pirate obtient des données confidentielles sur les utilisateurs (numéro de sécurité sociale, numéro de carte bancaire, coordonnées, …), sur les visiteurs du serveur web (accès aux logs) ou sur le serveur (accès aux fichiers de mots de passe et de configuration) ;
- la disponibilité des données en bloquant l'accès à un service (déni de service sur le site web ou la base de données) ou à un utilisateur en particulier.

Les attaques liées aux sessions visent à accroître le niveau de privilège dans une application web protégée. Elles peuvent être le fait d'un utilisateur malveillant authentifié dans l'application ou d'un utilisateur non authentifié qui vole la session d'un utilisateur authentifié. Une fois que le pirate a augmenté son niveau de privilège, il a accès aux mêmes données que l'utilisateur victime du détournement de session. En fonction du niveau de privilège de l'utilisateur, la personne malveillante peut porter atteinte à l'intégrité, à la confidentialité et à la disponibilité des données.

Cet article présente le principe du détournement de session ainsi que les protections à mettre en oeuvre contre les attaques dédiées à l'obtention d'un jeton de session valide (prédiction, force brute, vol et fixation).

Un exemple minimal de gestion de session illustre cet article. Son but est de montrer les réglages de sécurité. Il comporte 4 scripts :

- *outils_session.php* (Listing 1) définit les fonctions de manipulation de session (ouverture, destruction, vérification de la validité) ;
- *deconnexion.php* (Listing 2) réalise la déconnexion de l'application ;
- *auth.php* (Listing 3) gère l'authentification de l'utilisateur (affichage du formulaire, contrôle des identifiants, mise en session des informations) ;
- *liste.php* (Listing 4) donne un exemple de script de l'application, il affiche un message et fournit

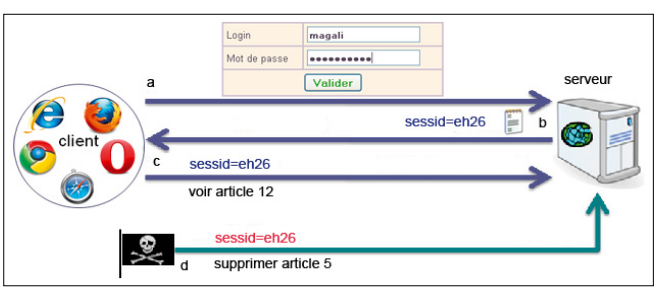


Figure 1. *Principe du détournement de session.*

Listing 1. *Outils_session.php*

```
<?php
// chemin de validité de la session (chaîne)
define('CHEMIN', '/appli/');
// communication cryptée (booleen)
define('HTTPS', true);
// durée de validité d'une session en secondes (entier)
define('DUREE', 1200);
// URL des scripts principaux
define('URL_APPLI', 'https://'.$_SERVER['SERVER_NAME'].
CHEMIN);
define('URL_AUTH', URL_APPLI.'auth.php');
define('URL_INDEX', URL_APPLI.'index.php');
define('URL_ERROR', URL_APPLI.'error.php');
/**
 * Règle les options de session et démarre la session.
 */
function initSession(){
        // utiliser les cookies uniquement
        ini_set("session.use_cookies", 1);
        ini_set("session.use_only_cookies", 1);
        ini_set('session.use_trans_sid', 0);
        /* fixer les proprietes du cookie :
        arg 1 = detruire le cookie quand le navigateur
est quitte,
        arg 2 = chemin de validite,
        arg 3 = garder le domaine par defaut,
        arg 4 = interdire la transmission en clair
(uniquement si https),
        arg 5 = ajouter la propriete HttpOnly */
        session_set_cookie_params(0, CHEMIN, '',
HTTPS, true);
        // envoyer le cookie de session PHP, ou
restaurer la session
        session_start();
}
/**
 * Efface les informations de la session.
 */
function detruireSession(){
        // supprimer le cookie de session
        setcookie(session_name(), '', time()-3600,
CHEMIN, '', HTTPS, true);
        // supprimer les variables de session
        $_SESSION = array();
        // fermer la session
        session_destroy();
}
/**
 * La session est valide si la personne s'est
authentifiée et si l'heure est
 * inférieure à celle en session.
 * @return vrai si la session est valide, faux sinon.
 */
function isSessionValide(){
        return (isset($_SESSION['login']) &&
                isset($_SESSION['timeout']) &&
(time() < $_SESSION['timeout']));
}
/**
 * Vérifie si le groupe a le droit d'accéder a la
ressource demandée.
 * @param $ressource string ressource demandée
 * @param $groupe string groupe de l'utilisateur connecté
 * @return vrai si l'accès est autorisé, faux sinon.
 */
function isAutorise($ressource, $groupe){
        // vérification des privilèges dans la base
}
/**
 * Démarre ou restaure la session et vérifie sa
validité, en cas de problème
 * redirige vers le formulaire d'authentification.
Vérifie ensuite que le
 * groupe a le droit d'accéder à la ressource demandée,
si problème redirige
 * vers une page d'erreur.
 */
function verifieAutorisation(){
        // initier ou restaurer la session
        initSession();
        // verifier la validite de la session
        if (!isSessionValide()){
                // supprimer la session
                detruireSession();
                // rediriger vers la page
d'authentification
                header('Location:'.URL_AUTH);
                exit;
        } else if (!isAutorise(basename($_
SERVER["SCRIPT_NAME"]), $_SESSION['groupe'])) {
                // rediriger vers une page d'erreur
                header('Location:'.URL_ERROR);
                exit;
        }
}
?>
```

un menu comportant un lien de déconnexion. Si la session n'est pas valide, l'utilisateur est redirigé automatiquement vers le script d'authentification.

Principe du détournement de session

Les sessions reposent sur un identifiant unique envoyé par l'application web à l'utilisateur. Ce jeton de session est ensuite transmis à l'application par le navigateur lors de chaque requête HTTP. L'identifiant peut être placé dans la partie requête de l'URL ou dans un cookie de session. Les cookies et le mécanisme des sessions ont été présentés dans deux numéros de ce même magazine ; pour en savoir plus veuillez vous reporter aux articles intitulés *Manipuler les cookies avec PHP* (PHP Solutions 4/2010) et *Manipuler les sessions avec PHP* (PHP Solutions 6/2010).

Le mécanisme des sessions reposant sur un jeton, le pirate doit fournir un jeton de session valide pour détourner une session d'un utilisateur authentifié (*session hijacking*). Dans la Figure 1, un utilisateur légitime s'authentifie sur le serveur (a), l'application lui envoie le jeton de session dont le nom est sessid et la valeur `eh26` (b) ainsi que la page demandée. Le navigateur retourne automatiquement le jeton de session au cours de la navigation (c). Lors de chaque requête l'application vérifie que l'utilisateur s'est préalablement authentifié et a des privilèges sur les ressources demandées grâce au jeton de session. Par exemple, si l'application est un CMS et que l'utilisateur connecté a des privilèges d'édition, le fait d'envoyer le jeton de session lui permet d'éditer, de supprimer ou de créer des articles. Si un pirate obtient la valeur du jeton de session et qu'il envoie ce jeton ainsi qu'une requête au serveur (d), alors il pourra légitimement supprimer ou modifier des articles.

La transmission du jeton de session peut être réalisée dans l'URL ou dans un cookie, en fonction des propriétés définies pour la session. Dans le premier cas il suffit que le pirate modifie la valeur du jeton dans la chaîne de requête pour envoyer la valeur souhaitée au serveur. Dans le second cas, il faut que le pirate intercepte la requête avant l'envoi, afin d'ajouter ou modifier le cookie, en utilisant un proxy ou un plugin dédié dans

Listing 2. *deconnexion.php*

```
<?php
// inclure les fonctions
require_once 'outils_session.php';
// restaurer la session
initSession();
// détruire la session courante
detruireSession();
// message de déconnexion
echo 'Deconnexion OK';
?>
```

Listing 3. *auth.php*

```
<?php
// ouvrir une session
require_once 'outils_session.php';
initSession();
/**
 * Filtre les données.
 * @param $var string données à filtrer
 * @return string donnée filtrée
 */
function filterInput($var){
        // opération de filtrage à implémenter
        return $var;
}
/**
 * Vérifie le login et le mot de passe, récupère le
groupe de l'utilisateur
 * @param $login string login de l'utilisateur
 * @param $pwd string mot de passe de l'utilisateur
 * @param $groupe string groupe de l'utilisateur
 * @return vrai si les identifiants sont corrects, faux
sinon
 */
function verifLogin($login, $pwd, &$groupe){
        // opération de vérification à implémenter
}
/**
 * Affiche le formulaire d'authentification
 */
function afficheFormLogin(){
        echo "<form action='auth.php' method='POST'>
                <input type='text' name='login'>
login<br>
                <input type='password' name='pwd'>
password<br>
                <input type='submit'
value='controler'>
                </form>";
}
// Si on a reçu des données du formulaire, faire
l'authentification
if (isset($_POST['login']) && isset($_POST['pwd'])){
        // filtrer les entrées utilisateur
        $p_login = filterInput($_POST['login']);
        $p_pwd = filterInput($_POST['pwd']);
        $groupe = null;
        // vérifier le login et récuperer le groupe
        if (verifLogin($p_login, $p_pwd, $groupe)){
                // regénérer le jeton de session
                session_regenerate_id();
                // fixer l'heure d'expiration
                $_SESSION['timeout'] = time() + DUREE;
                // stocker le login et le groupe
                $_SESSION['login'] = $p_login;
                $_SESSION['groupe'] = $groupe;
                // rediriger vers la page principale
de l'application
                header('Location:'.URL_INDEX);
                exit;
        } else {
                echo '<div class="erreur">identifiants
incorrects</div>';
                afficheFormLogin();
        }
} else {
        afficheFormLogin();
}
?>
```

un navigateur. Il est parfois également possible d'exploiter une faille XSS de l'application pour la forcer à placer le cookie souhaité dans le navigateur du pirate.

Utilisez les jetons PHP

Par défaut PHP génère automatiquement un jeton de session aléatoire comportant un grand nombre de caractères. Afin de protéger les applications des attaques par prédiction d'identifiant de session et des attaques par force brute, il est conseillé de ne jamais fixer la valeur du jeton de session dans l'application et de déléguer cette responsabilité à PHP. Aucune application web PHP ne devrait reposer sur des jetons de session dont les valeurs sont définies par le développeur, il ne faut donc jamais fixer la valeur du jeton de session avec la fonction PHP `session_id`.

Les attaques par prédiction reposent sur la prédiction d'un jeton de session valide, c'est-à-dire que le pirate devine l'identifiant de session à utiliser. C'est le cas lorsque la valeur du jeton est un nombre entier incrémenté à chaque ouverture de session. Par exemple, l'application attribue au premier utilisateur authentifié l'entier 101, au second utilisateur l'entier 102, au troisième l'entier 103, etc. Un utilisateur légitime de l'application, par exemple l'utilisateur qui a le jeton 103, peut ainsi deviner le système de numérotation et détourner la session d'un autre utilisateur en transmettant le jeton 101 ou 102. Pour se protéger de ces attaques, il faut que l'identifiant de session soit aléatoire, ce qui est le cas des jetons générés automatiquement par PHP.

Les attaques par force brute consistent à utiliser un processus automatique pour trouver la valeur d'un jeton de session valide. Un programme d'attaque par force brute envoie un grand nombre de requêtes HTTP par secondes. Il peut ainsi tester des millions de combinaisons et forger un jeton de session valide si le jeton ne comporte pas assez de bits. Pour se protéger de ces attaques, il faut que le jeton de session soit aléatoire et qu'il comporte suffisamment de caractères. La meilleure protection est donc d'utiliser les jetons PHP.

Dans cet article, le nom et la valeur du jeton de session ont été abrégés pour simplifier les exemples. Seuls les quatre premiers caractères du jeton de session sont affichés. Le nom du jeton de session utilisé est `sessid`, par défaut PHP génère un jeton de session `PHPSESSID` (ce nom dépend de la valeur de la directive `session.name` du fichier *php.ini*).

Transmettez le jeton dans un cookie

PHP accepte deux mécanismes de transmission du jeton de session. Le jeton peut être communiqué dans un cookie ou dans la chaîne de requête de l'URL. L'utilisation de l'URL pour transmettre le jeton pose des problèmes de sécurité importants car le jeton est stocké sur le client et sur un ou plusieurs serveurs.

Le jeton transmis par URL apparaît dans la barre d'URL du navigateur, derrière le caractère point d'interrogation.

Par exemple, si l'utilisateur demande la ressource *liste.php* située dans le répertoire *appli* du serveur web *www.monSiteWeb.fr*, l'URL contient dans la chaîne de requête le nom du jeton de session (*sessid*) et sa valeur (`eh26`) : *http://www.monSiteWeb.fr/appli/liste.php?sessid=eh26.* L'URL est stockée dans l'historique de navigation. Elle peut également être stockée dans un signet si l'utilisateur en crée un. Si l'ordinateur est utilisé par un tiers (ordinateur dans un cyber café, par exemple) et que la session n'a pas été détruite par une déconnexion, alors l'accès à la session sera possible à partir de l'historique ou du signet pour tout autre utilisateur de l'ordinateur.

Les serveurs web stockent dans un fichier de logs la première ligne de chaque requête HTTP reçue, la date et l'heure de la requête, ainsi que l'adresse IP de l'ordinateur qui a émis la requête. La première ligne d'une requête HTTP GET ou POST contient l'URL de la ressource demandée. Par exemple, lorsque la ressource *liste.php* est demandée, la requête HTTP envoyée au serveur est la suivante :

```
GET /appli/liste.php?sessid=eh26 HTTP/1.1
Host: www.monSiteWeb.fr
```

Dans cet exemple, le fichier de logs contiendra donc le jeton de session valide `sessid=eh26`. Il peut arriver qu'un pirate visualise le contenu du fichier de logs en exploitant une faille d'exécution de commande. Il peut également obtenir les URL demandées par le biais d'applications qui réalisent des statistiques de fréquentation. Si une telle application est installée sur le serveur web et que son accès n'est pas protégé, alors les jetons de session peuvent être obtenus facilement par un utilisateur malveillant en consultant les statistiques. Le jeton placé dans l'URL peut également être stocké dans les logs d'autres serveurs que celui qui héberge l'application. C'est le cas par exemple si un proxy est utilisé. Enfin, si l'application web contient un lien vers une page web externe, le jeton pourra être lu dans le `HTTP_REFERER` du lien externe (URL de la page à partir de laquelle l'utilisateur a accédé à la ressource courante).

Afin d'éviter toutes ces expositions du jeton de session, il faut imposer à PHP la transmission du jeton dans un cookie. Lorsque le jeton est transmis par cookie, l'URL pour obtenir la page *liste.php* ne contient plus le jeton : *http://www.monSiteWeb.fr/appli/liste.php.* La requête HTTP envoyée au serveur est la suivante :

```
GET /appli/liste.php HTTP/1.1
Host: www.monSiteWeb.fr
Cookie:sessid=eh26
```

Le fichier de log ne contient donc plus le jeton de session et ce jeton n'apparaît plus dans l'URL. Afin de forcer PHP à utiliser uniquement des cookies pour les sessions, il faut attribuer la valeur `1` aux directives `session.use_only_cookies` et `session.use_cookies` et la valeur `0` à la directive `session.use_trans_sid`. Ces directives ont été décrites dans l'article dédié à la manipulation des sessions ; veuillez-vous reporter à cet article pour plus d'explications. Les valeurs des directives peuvent être fixées dans le fichier *php.ini* de configuration PHP ou dans les scripts de l'application. Dans le premier cas, toutes les applications du serveur qui reposent sur les sessions seront alors concernées. Dans le second cas, le réglage sera local à l'application et devra intervenir avant l'ouverture de la session réalisée par l'instruction `session_start`.

La fonction `initSession` du Listing 1 fixe les valeurs des directives en utilisant la fonction PHP `ini_set`, puis elle démarre ou restaure la session. Elle est appelée par la fonction `verifieAutorisation` du même Listing. Cette dernière contrôle la validité de la session après sa restauration. Si la session n'est pas valide, la fonction redirige vers le script gérant l'authentification de l'utilisateur. Tous les scripts de l'application web incluent le Listing 1 et appelent la fonction `verifieAutorisation`, sauf les scripts d'authentification et de déconnexion qui n'ont pas à contrôler la validité de la session. Ces derniers appellent directement la fonction `initSession`. L'utilisation d'une fonction dédiée à l'ouverture de la session dans tous les scripts, permet de s'assurer que l'application fonctionnera uniquement avec des jetons transmis par cookie, quels que soient les réglages effectués par l'administrateur système du serveur web dans le fichier *php.ini*.

Ne transmettez pas le jeton en clair

Lorsque l'application web utilise le protocole HTTP, les communications ne sont pas cryptées. C'est-à-dire que si les échanges entre le client et le serveur sont interceptés par un tiers, les contenus des requêtes et des réponses apparaissent en clair. Les logins et mots de passe, les jetons de session, ainsi que les informations confidentielles (numéro de sécurité sociale, coordonnées, …) ne sont donc pas protégés. Pour éviter l'interception de jeton par écoute du réseau, il faut utiliser une connexion cryptée (SSL). Lorsque les données transmises sont sensibles (numéro de carte bancaire, numéro de sécurité sociale, jeton de session, …), les communications devraient toujours être cryptées. Si les échanges entre le client et le serveur sont effectués par le protocole HTTPS, c'est-à-dire si l'URL commence par *https://*, alors la communication est cryptée.

Si vous bénéficiez d'une communication sécurisée vous pouvez, pour plus de sécurité, régler la propriété de cookie `secure` afin d'interdire l'envoi de cookies lorsque la communication n'est pas établie en HTTPS. Si l'application est accessible à la fois par les protocoles HTTP et HTTPS, ce réglage évite que les cookies de session soient envoyés en clair. Le réglage peut

être effectué dans le fichier de configuration PHP ou dans les scripts de l'application. Dans le premier cas, il suffit de donner la valeur `on` à la directive `session.cookie_secure` du *php.ini*. Dans le second cas, il est possible d'effectuer le réglage en utilisant la fonction `ini_set`, ou en passant la valeur `true` à l'avant-dernier argument de la fonction `session_set_cookie_params`. Ces fonctions ont été décrites dans l'article sur les sessions ; veuillez vous reporter à cet article pour plus d'informations. La fonction `initSession` du Listing 1 règle les propriétés des cookies en utilisant la fonction `session_set_cookie_params`.

Protégez les cookies des attaques XSS

Bien que la transmission du jeton de session par cookie soit la méthode la plus sûre, le cookie de session peut lui-même être l'objet d'un vol. Il peut être obtenu par un pirate via une attaque XSS. Le XSS (*Cross Site Scripting*) consiste à injecter un contenu actif dans un document HTML. Par exemple l'injection du code JavaScript `alert(document.cookie)` provoquerait l'affichage dans le navigateur de la victime d'une fenêtre d'alerte contenant les cookies pour le domaine et le chemin de la page demandée. Généralement le pirate injecte un code XSS qui provoque l'envoi du cookie à un serveur distant, où le pirate pourra récupérer ultérieurement le jeton de session et l'URL de l'application web concernée par le jeton.

Toute donnée provenant d'un utilisateur est susceptible de contenir des chaînes de caractères malicieuses qui ont un sens pour un interpréteur JavaScript. La règle est de ne jamais faire confiance aux données envoyées par le client. Le filtrage des données sur le serveur est la première étape assurant le bon fonctionnement de l'application ainsi que la sécurisation de l'application et du système qui l'héberge. Il faut vérifier les données en entrée mais aussi traiter les données avant de les envoyer au client, notamment en utilisant la fonction `htmlentities`. Pour protéger les cookies de session des attaques XSS, il faut donc filtrer les entrées de l'application et protéger ses sorties.

Afin de lutter contre le vol des cookies de session, la propriété `HttpOnly` peut être ajoutée dans le champ `Set-Cookie` de l'en-tête de la réponse HTTP. Celle-ci indique au navigateur qu'il n'est autorisé à transmettre le cookie que dans les requêtes HTTP, le cookie ne peut pas être lu ou modifié depuis un script sur le client. Par exemple, la ligne ci-après de la réponse HTTP envoie un cookie de session `sessid` dont l'accès depuis JavaScript est interdit :

```
Set-Cookie: sessid=eh26; HttpOnly
```

Lorsque cette propriété est supportée par le navigateur, l'exécution du code JavaScript `alert(document.cookie)` ne montre plus le cookie de session. Depuis la version 5.2.0, PHP peut ajouter cette propriété automatiquement lors de l'envoi du cookie de session au navigateur. La présence de cette propriété dans le champ d'en-tête HTTP dépend de la valeur de la directive `session.cookie_httponly` du fichier de configuration *php.ini* (valeur `on` pour ajouter `httponly`). Il est également possible d'indiquer à PHP d'ajouter cette propriété en affectant le booléen `true` au dernier argument de la fonction `session_set_cookie_params` (cf. fonction `initSession` du Listing 1).

Bien qu'utile, cette propriété n'est pas suffisante pour garantir la protection des cookies de session. D'une part elle n'est pas implémentée dans tous les navigateurs, car elle n'est pas standard. D'autre part, des contre-mesures ont été trouvées par les pirates pour détourner des cookies comportant la propriété `HttpOnly` (attaques XST). Vous ne devez donc pas faire reposer la sécurité du cookie de session uniquement sur cette propriété.

Fixez le chemin de validité du cookie

Avant d'envoyer un cookie, le navigateur vérifie son chemin. Si la ressource demandée n'a pas le même chemin, alors le cookie n'est pas envoyé. Si le chemin est trop général, l'application est susceptible d'être attaquée par le biais d'autres applications sur le même serveur. Par exemple, si le chemin du cookie de session est défini sur la racine du serveur web, et non pas sur celle de l'application, le cookie sera envoyé par le navigateur à toutes les ressources demandées sur le serveur qui héberge l'application. Ceci augmente le risque de vol du cookie.

Par défaut, le chemin des cookies envoyé par PHP est la racine du serveur web (caractère `/`). Il est défini par la directive `session.cookie_path` du fichier *php.ini*. Pour restreindre la validité au chemin de l'application, définissez le chemin avec la fonction `session_set_cookie_params`. Ceci devra être fait avant l'appel à la fonction `session_start`.

Dans le Listing 1, le chemin de validité est celui du répertoire `appli` qui contient l'application web, il est défini par la constante `CHEMIN`. La propriété est fixée dans la fonction `initSession`. L'application est située à la racine du serveur web, le chemin est donc `/appli/`. Le caractère `/` après le nom du répertoire évite que le cookie soit envoyé à d'autres répertoires qui commenceraient par la sous-chaîne `appli`.

Gérez la déconnexion

Il est important de fournir à l'utilisateur un lien ou un bouton de déconnexion sur chaque page de l'application web. Pour être efficace, la procédure de déconnexion doit supprimer le fichier de session, effacer les variables de session, envoyer un cookie de session vide et dont la date limite est dépassée. La fonction `detruireSession` du Listing 1 réalise toutes ces opérations. Pour effacer

un cookie il faut en envoyer un de même nom, de même domaine et de même chemin. Dans cette fonction, le nom du jeton de session est obtenu grâce à la fonction `session_name`. Le cookie envoyé a une valeur vide et une date de validité antérieure d'une heure à l'heure courante. Le chemin de validité du cookie est celui du répertoire `appli`, il doit être identique au chemin donné dans la fonction `initSession`.

La destruction de la session courante ne peut être réalisée que si la session a été restaurée préalablement. Le script `deconnexion.php` du Listing 2 effectue la restauration puis la déconnexion. Ceci permet de détruire entièrement les informations de la session courante. Si l'utilisateur s'est déconnecté et que la déconnexion réalise les trois opérations décrites ci-avant, alors la session ne sera plus exploitable ultérieurement par un pirate qui a intercepté le jeton.

Limitez la durée de vie de la session

Un pirate peut voler un cookie de session par une attaque XSS, en exploitant un bug d'un navigateur, en écoutant le réseau ou dans le gestionnaire de session du navigateur lorsque l'ordinateur est partagé par plusieurs utilisateurs. Une fois le cookie obtenu, le détournement de session ne sera possible que si la session est encore active, c'est-à-dire si l'utilisateur ne s'est pas déconnecté ou si l'application ne gère pas correctement la procédure de déconnexion décrite dans la section précédente. Beaucoup d'attaques sont possibles car les données de sessions restent présentes sur le serveur web.

Plus la durée de validité d'une session est courte, plus il sera difficile pour un pirate de détourner la session d'un utilisateur. Afin de limiter l'utilisation d'un jeton intercepté, il est utile de mettre en place dans l'application web un contrôle de la durée de validité. Il est conseillé de stocker l'heure limite de validité de la session dans une variable de session et de vérifier à chaque requête de l'utilisateur que cette heure limite n'est pas dépassée. Afin de ne pas bloquer l'utilisateur légitime, vous pouvez actualiser l'heure limite au cours de la navigation. La durée de validité dépend du niveau de sécurité de l'application. L'OWASP (*Open Web Application Security Projec*t) recommande une durée de 20 minutes pour les applications non sensibles, et de 5 minutes pour les applications sensibles.

Le script auth.php du Listing 3 gère l'authentification de l'utilisateur. Lorsque les identifiants de connexion sont corrects, il crée une variable de session timeout qui stocke l'heure à laquelle la session ne sera plus valide. La durée de validité est fixée à 20 minutes par la constante DUREE du Listing 1. La fonction isSessionValide du Listing 1 est appelée par la fonction verifieAutorisation. Tous les scripts de l'application prennent donc les mêmes options de session, démarrent la session et contrôlent sa validité. Chaque page de l'application web est protégée, elle n'est accessible que si l'utilisateur a une session en cours. La fonction isSessionValide vérifie que l'utilisateur s'est préalablement authentifié (la variable de session login existe) et que l'heure d'expiration n'est pas atteinte. Dans les exemples de cet article la validité de la session est de vingt minutes, quelles que soient les opérations réalisées au cours de la navigation. En fonction du niveau de sécurité de l'application, vous pouvez actualiser l'heure de fin de validité à chaque fois que l'utilisateur réalise une opération, afin de lui éviter une déconnexion vingt minutes après la phase d'authentification.

Par ailleurs, il est conseillé de limiter la durée de vie des cookies de session dans le navigateur. Ceux-ci doivent être impérativement détruits lorsque le navigateur est quitté. Ceci est réalisé en attribuant la valeur `0` à la directive `session.cookie_lifetime` du fichier *php.ini* ou en affectant la valeur `0` au premier argument de la fonction `session_set_cookie_params` (cf. fonction `initSession` du Listing 1).

Régénérez le jeton de session

Plutôt que de voler le jeton de session ou de le deviner, le pirate peut imposer à un utilisateur d'utiliser un jeton. Une fois la victime authentifiée, le pirate utilise ce jeton pour usurper son identité.

Pour comprendre cette attaque, il faut tout d'abord savoir comment PHP gère une session. Lorsqu'un utilisateur demande une ressource d'une application web utilisant le mécanisme des sessions, PHP vérifie la présence d'un jeton de session. Si celui-ci est présent, il cherche les données pour cette session. S'il n'y en a pas, il crée un fichier de session pour le jeton. L'application vérifiant généralement la présence d'une variable de session pour déterminer si la personne est correctement authentifiée, l'utilisateur est redirigé vers un formulaire d'authentification. Une fois l'authentification effectuée, l'utilisateur conserve le jeton de session pendant la durée de sa navigation. L'acceptation par PHP d'un jeton de session avant l'authentification permet à un pirate d'usurper l'identité d'un utilisateur légitime, dès lors qu'il réussit à imposer ce jeton de session à la victime. Plusieurs méthodes peuvent lui permettre d'arriver à ses fins.

Les attaques par fixation du jeton de session sont facilitées par la transmission du jeton dans l'URL. Dans ce cas, le pirate peut facilement ajouter le jeton dans un lien sur lequel la victime cliquera. Il peut également exploiter une faille XSS pour ajouter un champ caché `sessid` dans le formulaire d'authentification de l'application. Le pirate peut aussi créer un formulaire de login contenant le jeton sur un autre serveur que celui de l'application et l'envoyer à l'application. Ceci est une raison supplémentaire de ne pas accepter les jetons de session par URL et de leur préférer les cookies (directive `session.use_cookies` = 1).

Quand la transmission par cookie est utilisée, PHP cherche tout d'abord si un jeton de session est présent dans les cookies. Si ce n'est pas le cas, il en cherche un dans l'URL. Un exemple typique d'attaque est le suivant. L'utilisateur clique sur un lien vers l'application web, celui-ci contient un jeton de session dont la valeur est `eh26`. Ce peut être un lien placé par le pirate dans un mail ou dans une page web. Le script d'authentification crée une session pour ce jeton et retourne un formulaire d'authentification. Ce dernier comporte le jeton de session dans un champ caché généré automatiquement, si la directive `session.use_trans_sid` a la valeur `1`. En PHP4, un cookie dont la valeur est `eh26` est retourné en même temps que le formulaire ; ce n'est plus le cas avec PHP5. Le formulaire est rempli par l'utilisateur, sa validité est vérifiée et les variables de session sont créées. Le pirate peut maintenant se connecter sur le site en utilisant l'identifiant qu'il avait fixé.

Afin d'interdire la prise en compte du jeton passé dans l'URL, il faut veiller à ce que la directive `session.use_only_cookies` soit activée. Il faut également désactiver la complétion automatique des URL avec la directive `session.use_trans_sid` (cf. Listing 1). Avec les deux réglages préconisés, le pirate ne peut plus imposer le jeton de session par le biais de l'URL.

Limiter la transmission des jetons uniquement par cookies n'est pas une protection suffisante contre les attaques par fixation. Un pirate peut en effet exploiter une faille de sécurité de l'application web (XSS, HTTP *Response Splitting*) pour qu'un serveur web place un cookie dans le navigateur de la personne dont il veut détourner la session. Ces attaques reposant sur un identifiant de session fixé par le pirate, il suffit de régénérer un nouvel identifiant de session lorsqu'il y a un changement de niveau de privilège (utilisateur authentifié, passage en mode administration, …). L'utilisation de la fonction PHP `session_regenerate_id` une fois l'authentification effectuée permet de protéger l'application. Cette fonction produit un nouveau jeton de session et l'envoie au navigateur. Dans la mesure où cette fonction envoie un cookie, il faut veiller à ne pas envoyer de données avec `echo`, `print`, … avant de l'appeler. Depuis PHP 5.1 la fonction accepte un argument booléen indiquant s'il faut supprimer (`true`) la vieille session. Par défaut elle est conservée. Le Listing 3 effectue la régénération du jeton de session une fois que l'utilisateur s'est authentifié.

Protégez les données de session

Afin de mémoriser les informations de session pendant toute la durée de navigation d'un internaute, celles-ci sont stockées dans un fichier temporaire sur le disque dur du serveur web. L'application web gérant simultanément plusieurs utilisateurs, un fichier est créé par session. Le nom du fichier de session contient la valeur du jeton de session, préfixé par `sess`. Ces fichiers sont généralement stockés dans un répertoire temporaire, dont l'emplacement est défini par la directive `session.save_path` du fichier *php.ini*.

Si un utilisateur malveillant a accès à ces fichiers de session, il peut obtenir les valeurs des jetons de session à partir des noms de fichiers, il peut également accéder aux données de session contenues dans ces fichiers. La consultation et l'écriture de fichiers de session est le plus souvent réalisée en interne par un utilisateur qui a accès au serveur web en *ssh* ou *sftp*. Cependant une faille d'injection (inclusion de fichier, exécution dans un *shell*) peut également permettre l'accès à ces fichiers à un pirate qui n'a aucune autorisation d'accès au serveur web. Il devient ainsi possible d'obtenir un identifiant de session valide ou de créer une session (création du fichier), mais également de voler des informations sensibles contenues dans le fichier de session. Si aucun utilisateur n'a accès au serveur web par *ssh* ou *sftp* en dehors de l'administrateur, il faut protéger les applications web des failles d'inclusion de fichier et d'exécution de *shell* pour éviter qu'un utilisateur malveillant obtienne des jetons de session valides à partir des noms des fichiers de session. Si possible, il faut interdire l'exécution de commandes telles que `system`, `exec` avec la directive `disable_functions` du *php.ini*.

La meilleure protection pour interdire l'accès aux données de session et l'exposition de la valeur du jeton de session, est de ne plus les stocker dans des fichiers. Deux solutions sont envisageables : stocker les informations dans une base de données ou en mémoire.

Dans le premier cas, l'application doit définir son gestionnaire de session, c'est-à-dire qu'il faut fournir les fonctions pour stocker et manipuler les données dans la base. Les fonctions à écrire seront appelées lors de l'ouverture et de la fermeture de la session, lors de la lecture et de l'écriture de données en session (variable `$_SESSION`), et lors de la fin d'exécution du script. La fonction `session_set_save_handler` permet d'indiquer les noms des fonctions du gestionnaire de session. Elle doit être appelée avant de démarrer la session.

La seconde solution est plus simple à mettre en place, car elle ne demande aucune création de fonction. Elle repose sur un serveur *memcached* et sur l'extension *memcache* (*memcache.so* ou *memcache.dll*). Celle-ci se charge de la mise en cache et de la récupération des données de session. L'administrateur système doit installer un serveur *memcached* et l'extension *memcache* pour PHP. Il suffit ensuite de définir les deux directives du *php.ini* :

```
session.save_handler = memcache
session.save_path = tcp://IP_serveur_memcached:
port_memcached?persistent=1
```

Il est également possible de combiner cette solution avec celle de l'utilisation d'une base de données afin d'avoir un gestionnaire de session tout le temps opérationnel,

même lorsque le serveur *memcached* ne répond pas. Pour ce faire il faut programmer son propre gestionnaire de session, c'est-à-dire définir les fonctions décrites précédemment et utiliser `session_set_save_handler`.

Utilisez un moyen d'identification secondaire

Le jeton de session est le premier moyen d'identification. Il peut être combiné à un autre moyen d'authentification pour les applications sensibles. Celui-ci ne peut pas reposer sur l'adresse IP car, d'une part, celle-ci peut varier au cours d'une session valide d'un utilisateur en fonction des choix de l'infrastructure réseau effectués par le fournisseur d'accès, et d'autre part lorsqu'un proxy est utilisé dans l'entreprise, plusieurs utilisateurs d'un même réseau peuvent avoir la même adresse IP (translation d'adresse). Dans le premier cas, invalider une session lorsque l'adresse IP change bloquerait la navigation d'un utilisateur légitime. Dans le second cas, des tentatives d'usurpation de session par un pirate situé sur le même réseau que la victime passeraient inaperçues.

Il est possible d'utiliser le champ `User-Agent` de l'en-tête HTTP comme moyen d'identification secondaire. Ce champ optionnel contient le nom et la version du navigateur utilisé par l'internaute. Si l'application reçoit un jeton de session mais que le navigateur qui l'a envoyé n'est pas celui qui a initié la session, alors il est fort probable que ce soit une tentative de détournement de session. La session étant liée à un cookie placé dans un navigateur, le jeton devrait toujours être transmis par le même navigateur pendant toute la durée de la session. Lors de la phase d'authentification, certains développeurs stockent le contenu du champ `User-Agent` dans une variable de session. Ce champ étant présent dans la requête HTTP, il est susceptible de contenir des données malicieuses, comme toute autre entrée utilisateur. Il est donc conseillé d'utiliser la fonction `md5` afin de simplifier les comparaisons ultérieures :

```
$_SESSION['agent'] = md5($_SERVER['HTTP_USER_
AGENT']);
```

Lors de chaque requête, l'application vérifie si la personne s'est authentifiée (présence d'une variable de session), si la session n'a pas expiré et si le MD5 du navigateur qui a émis la requête est identique à celui stocké en session. En cas de problème, l'application demande à l'utilisateur de s'authentifier. Il faut noter cependant que si le pirate utilise le même navigateur, il ne sera pas bloqué par cette mesure d'authentification secondaire. De plus, le nom et la version du navigateur peuvent être obtenus facilement en JavaScript. L'instruction JavaScript `alert(navigator.userAgent)` affiche le nom et la version du navigateur dans les principaux navigateurs utilisés actuellement (Firefox, Internet Explorer, Safari, Chrome, Opera). La chaîne retournée par cette instruction est identique à celle transmise par le navigateur dans la requête HTTP. Si le pirate a exploité une faille XSS pour obtenir le cookie, il peut facilement obtenir par ce biais les informations de version. Il lui suffit ensuite de remplacer la valeur du champ d'en-tête `User-Agent` dans la requête envoyée par son navigateur. Des outils permettent d'intercepter et modifier les requêtes très facilement. Utiliser l'information contenue dans le champ `User-Agent` n'est donc pas une mesure de protection supplémentaire suffisante.

Une autre approche consiste à propager une chaîne aléatoire pendant toute la session. Utilisée en conjonction avec une connexion cryptée, la chaîne aide à s'assurer que la personne qui vient d'envoyer le jeton de session est bien celle qui s'est authentifiée. La chaîne peut être transmise par cookie, mais dans ce cas elle risque d'être elle-même interceptée si le jeton a été obtenu par un XSS. Il est préférable de toujours transmettre le jeton de session dans un cookie et la chaîne aléatoire dans l'URL. Lors de l'authentification il faut générer et stocker en session l'identifiant aléatoire unique, après avoir régénéré le jeton de session :

```
$_SESSION['id'] = md5(uniqid(rand(), TRUE));
```

Listing 4. *liste.php*

```
<?php
// restaurer la session
require_once 'outils_session.php';
verifieAutorisation();
?>
<!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>Liste</title>
</head>
<body>
        <div>Une liste ...</div>
<div>
<a href='deconnexion.php'>deconnexion</a>
</div>
</body>
</html>
```

Listing 5. *Modification de la fonction isSessionValide*

```
/**
 * La session est valide si la personne s'est
authentifiée, si l'heure est
 * inférieure à celle en session, et si le jeton
aléatoire est dans l'URL
 * et est identique à celui en session.
 * @return vrai si la session est valide, faux sinon.
 */
function isSessionValide(){
        $g_id = (isset($_GET['id'])) ? trim($_
GET['id']) : null;
        return (isset($_SESSION['login']) &&
                        isset($_SESSION['timeout'])
&& (time() < $_SESSION['timeout']) &&
                        !empty($g_id) && !empty($_
SESSION['id']) &&
                        ($g_id == $_SESSION['id']));
}
```

Il suffit d'adapter le Listing 3 en ajoutant cette ligne de code, lorsque l'utilisateur a fourni des identifiants valides au formulaire d'authentification. Il faut ensuite ajouter l'identifiant dans tous les liens de l'application (liens hypertextes, attribut action des formulaires). Par exemple la redirection vers la page principale après authentification du Listing 3 devient :

```
header('Location:'.URL_INDEX.'?id='.$_
SESSION['id']);
```

Lorsque l'utilisateur émet une requête, l'application doit vérifier comme précédemment que l'utilisateur s'est authentifié préalablement (présence d'une variable de session), que l'heure d'expiration n'a pas été dépassée, et que la chaîne aléatoire a bien été envoyée et est identique à celle stockée dans la variable de session. Le code de la fonction `isSessionValide` du Listing 1 a été modifié pour prendre en compte cette vérification dans le Listing 5.

Vérifier les droits d'accès

Une application utilisant les sessions comporte en général plusieurs groupes d'utilisateurs avec différents niveaux de privilèges. Il est important de vérifier que l'utilisateur connecté a des droits sur chaque ressource demandée. Lors de l'authentification à l'application, le groupe auquel l'utilisateur appartient doit être stocké dans une variable de session. C'est ce qui est fait dans le Listing 3.

Dans le Listing 1, la fonction `verifieAutorisation` ouvre la session et vérifie que le groupe de l'utilisateur connecté a le droit d'accès à la page demandée. Si ce n'est pas le cas, l'utilisateur est redirigé vers une page affichant un message d'erreur. Cette fonction étant appelée au début de chaque script de l'application, toutes les pages sont ainsi protégées.

Afin de simplifier les listings, le code de la fonction `isAutorise`, qui effectue la vérification des privilèges, n'est pas donné dans cet article. En général les ressources accessibles pour chaque groupe sont stockées dans une base de données. La fonction compare les droits d'accès et retourne un booléen.

Conclusion

Vous savez à présent configurer les directives de sécurité des sessions et mettre en place des mécanismes pour protéger vos applications d'un détournement de session. Le niveau de protection à mettre en oeuvre dépend de l'application web. Les règles de base à appliquer, quel que soit le niveau de sécurité de l'application web, sont les suivantes :

- utiliser les jetons générés automatiquement par PHP,
- transmettre le jeton uniquement par cookie,
- fixer un chemin de validité pour le cookie,
- régénérer le cookie après l'authentification,
- protéger le cookie des attaques XSS,
- fournir un moyen de déconnexion,
- limiter la durée de vie de la session,
- vérifier les droits d'accès d'un groupe d'utilisateurs à une ressource.

Toutes ces opérations peuvent être réalisées dans le code de l'application. Elles ne dépendent donc pas du réglage du serveur web. Si possible, il faut également utiliser une communication cryptée, afin que le jeton et les données sensibles ne soient pas transmis en clair. Si l'application est hébergée sur un serveur web dont vous n'êtes pas l'administrateur, et que celui-ci ne supporte pas le protocole HTTPS, vous ne pourrez malheureusement pas utiliser cette protection.

Si l'application est sensible vous devez en plus :

- transmettre le jeton et les données par le protocole HTTPS,
- utiliser un moyen d'identification secondaire transmis par URL,
- protéger les données de session et le jeton en mettant en place un stockage des données dans une base de données ou en mémoire cache.

Sur Internet

- *http://www.php.net/manual/en/session.security.php* – Sécurité des sessions dans le manuel officiel de PHP,
- *http://www.owasp.org/index.php/Session_Management* – Gestion de la sécurité des sessions dans le guide de l'OWASP (Open Web Application Security Project),
- *http://www.php.net/manual/fr/book.memcache.php* – Manuel memcache/PHP,
- *http://www.memcached.org* – site officiel de Memcached,
- *http://www.owasp.org/index.php/XST* – Informations sur le XST dans le guide de l'OWASP,
- *http://www.owasp.org/index.php/XSS* – Informations sur le XSS dans le guide de l'OWASP.

CÉCILE ODERO
MAGALI CONTENSIN

Cécile Odero est spécialisée dans la conception et le développement d'applications web en PHP. Elle est développeur web freelance.
Contact : cecile.odero@gmail.com
Magali Contensin est chef de projet en développement d'applications au CNRS. Elle enseigne depuis plus de dix ans le développement d'applications web à l'université et est l'auteur de nombreux articles sur le développement web en PHP.
Contact : http://magali.contensin.online.fr

www.sos-hebergement.com

Chez nous pas de promesse, que des engagements.

SOS Hébergement

Packs Mutualisés

Configuration à partir de :
> Plesk : 1
> Espace disque : 2 Go
> Trafic mensuel : 50 Go
> Base de données : 1
> Alias de domaine : 1
> Backup hebdomadaire

Prix : 35€ /an*

Packs Privés

Configuration à partir de :
> Plesk : 10
> Ram : 1/2 Go
> Processeur : 1/4 Core**
> Espace disque jusqu'à : 10 Go***
> Trafic mensuel : 500 Go/mois

Prix : 12€ /mois*

Packs Dédiés

Configuration à partir de :
> Plesk : 30
> Ram : 2 Go
> Processeur : 1 Core**
> Espace disque jusqu'à : 50 Go***
> Bande passante garantie : 1 To
> Ressources modifiables séparément

Prix : 49€ /mois*

Construisez l'hébergement qui vous ressemble !

Toutes les ressources mises à disposition sont garanties et modifiables indépendamment pour s'adapter à vos besoins.

Options disponibles

Un large choix d'options est disponible en fonction des packs :
- Infogérance
- Monitoring
- Licence Plesk 1 à 300 domaines
- Disques haute qualité, 10Kt, 15Kt
- Ip supplémentaire
- Bande passante garantie 100Mbs
- Data center Paris ou Amsterdam
- Sauvegarde hebdomadaire ou mensuelle désactivable
- CDN , Load Balancing ...

SOS Hébergement
www.sos-hebergement.com

0 892 230 322
CODE : 89747
0,34€/MIN

contact@sos-hebergement.com

*Prix HT.
**1 Core = Un processeur X5570.
*** Espace maximum sans sauvegarde.

Sécurisation

d'un répertoire avec .htaccess et .htpasswd

Protégez les parties sensibles de votre site web, de votre webapp ou de votre intranet grâce à Apache. Commet limiter l'accès grâce à un couple login/pass.

Cet article explique :

- Comment sécuriser un dossier grâce à Apache et n'autoriser l'accès qu'aux seuls visiteurs munis d'un couple login/pass valide.

Ce qu'il faut savoir :

- Quelques notions de PHP.
- Configurer son logiciel FTP.
- Manipuler les fichiers txt.

Sur un site internet ou plus encore sur une webapp, il est souvent utile de sécuriser un dossier pour en limiter l'accès aux seuls utilisateurs autorisés. Apache fournit pour cela une solution très efficace : la protection par fichier *.htaccess* et *.htpasswd*. Par défaut, tant le répertoire super-utilisateur que `DocumentRoot` sont réglés pour permettre la prise en compte des instructions de fichiers *.htaccess*. La directive `AllowOverride` du fichier *httpd.conf* définit si des Options peuvent être invalidées par les instructions d'un fichier *.htaccess*.

Les fichiers *.htaccess* sont des fichiers de configuration du serveur Apache, ils permettent de définir des règles dans un répertoire. On peut les utiliser pour protéger un répertoire par mot de passe, pour changer le nom ou l'extension de la page index (page appelée par défaut ; exemple : *http://monsite.com/* appelle par défaut *http://monsite.com/idex.html*, on peut très bien changer en *http://monsite.com/accueil.html*), ou encore pour personnaliser les pages d'erreur. Le fichier *.htaccess* est placé dans le répertoire dans lequel il doit agir. Il agit ainsi sur les permissions du répertoire qui le contient **et de tous ses sous-répertoires**. Néanmoins, cette solution a un défaut majeure : les mots de passe étant cryptés, impossible de retrouver un mot de passe perdu. A défaut de le retrouver, à l'aide de PHP, nous allons pouvoir créer un module permettant de réinitialiser un mot de passe et de réécrire le *.htpasswd* à la volée. Supposons que votre site *www.monsite.com* dispose d'un espace réservé qui se situe dans le dossier *www.monsite.com/admin* . Autorisez les droits en écriture pour ce dossier (CHMOD 777) afin que nous puissions générer le fichier *.htpasswd* (qui sera lui en CHMOD 644).

.htaccess

Nous allons tout d'abord créer un fichier *.htaccess* pour protéger celui-ci. Pour cela, ouvrez un simple éditeur de texte (*Bloc Notes, Vim, Emacs, Gedit, TextWrangler*…), créez un nouveau fichier vide et copiez-y le code du Listing 1 (modifié selon vos besoins).

Vous allez devoir modifier trois de ces lignes :

- `ErrorDocument 401` : c'est le chemin absolu de la page vers laquelle l'utilisateur sera redirigé en cas d'erreur (ne pas mettre cette page dans le répertoire protégé).
- `AuthUserFile` : c'est le chemin absolu du fichier *.htpasswd* qui contient le mot de passe crypté

Listing 1. *.htaccess*

```
ErrorDocument 401 /var/www/vhosts/monsite.com/
httpdocs/401.php
AuthUserFile /var/www/vhosts/monsite.com/httpdocs/
admin/.htpasswd
AuthGroupFile /dev/nullAuthName "Authentification
requise"
AuthType Basic
<limit GET>
require valid-user
</Limit>
```

(mettez-le au même endroit que le .htaccess, c'est plus simple).

- `AuthName` : c'est le texte qui apparaîtra dans la boite de dialogue demandant le mot de passe.

Si vous ne connaissez pas le chemin absolu de votre fichier, faites ceci :

- Créez un fichier nommé *path.php.*
- Insérez ce code PHP : <?php echo realpath('path.php'); ?>.
- A l'aide de votre logiciel FTP, placez ce fichier dans le même répertoire que votre *.htaccess.*
- Exécutez ce fichier dans votre navigateur. Il vous donnera le chemin absolu (/var/www/vhosts/monsite/httpdocs/admin/path.php dans mon cas).
- Remplacez path.php par .htpasswd et vous obtenez le chemin du *.htpasswd.*
- Faites de même pour votre fichier *401.php* (à ne pas mettre dans le répertoire à protéger, bien sûr. Pour bien vous organiser vous pouvez créer un répertoire « error ou erreurs » à la racine de votre site pour y stocker vos pages personnalisées d'erreur).

Enregistrez ce fichier en *htaccess.txt* puis, avec votre logiciel FTP, placez-le dans le répertoire à protéger (vous devez transférer ce fichier en mode ASCII (voir documentation de votre client FTP pour assurer la conversion des caractères *fin de ligne*) et renommez-le en *.htaccess* .

Voilà pour notre premier fichier.

.htpasswd

Nous allons maintenant créer notre fichier *.htpasswd*. Pour ce faire, nous allons créer un script PHP (Listing 2) contenant un formulaire permettant d'ajouter et de modifier les *login/pass*.

Ce fichier fournit un simple formulaire à deux champs (*login* et *pass*). Une fois validé, deux cas se présentent : soit le login est inexistant, auquel cas il sera ajouté avec le mot de passe choisi (crypté), soit il existe déjà, auquel cas le mot de passe sera remplacé par le nouveau (crypté). Enregistrez ce fichier *creer.php* dans le répertoire parent du dossier à protéger. Ce script fonctionnera dans 99% des cas, mais il faut noter que selon la configuration de Apache, les mots de passe peuvent être cryptés différemment. PHP définit une constante appelée `CRYPT_SALT_LENGTH` permettant de vous indiquer la longueur du salt disponible pour le système de hachage utilisé. Dans ce cas, essayer de changer :

// fonction pour crypter le mot de passe
`$passcrypt=crypt($pass);` //nouveau mot de passe crypté
// fin de la fonction

Listing 2. *Générateur de fichier .htpasswd*

```
<html>
<head>
</head>
<body>
<?php

//récupération des données du formulaire
$subform=$_POST['subform'];
$login=$_POST['login'];
$pass=$_POST['pass'];
// fin de la récupération
// fonction pour crypter le mot de passe
$passcrypt=crypt($pass); //nouveau mot de passe crypté
// fin de la fonction
$new="";//nouveau contenu fichier .htpasswd

if($subform and $login != "" and $pass != ""){// si formulaire ok

    if (file_exists("admin/.htpasswd")) {// si le fichier .htaccess existe

        $handle = fopen("admin/.htpasswd", "r");
             while (!feof ($handle)) {

                $ligne = fgets($handle, 4096);// on lit le .htaccess existant ligne par ligne
                $element=explode(":",$ligne);// on découpe la ligne pour récupérer le login et le pass
                if($element[0]==$_POST['login']){// si le login existe déjà
                    $new.=$login.":".$passcrypt."\n";// on écrase avec le nouveau login
                    $deja=1;// le login existait déjà
                }
                else{
                    $new.=$ligne."\n";
                }
            }
        fclose( $handle );

        if($deja==0){$new.=$login.":".$passcrypt;}// si le login n'existait pas, on l'ajoute
    }

    else{// si le fichier .htaccess n'existe pas

         $new.=$login.":".$passcrypt."\n";

    }
    $handle = fopen("admin/.htpasswd", "w+");
    fwrite($handle,"$new");// on réécrit le fichier .htpasswd
    fclose( $handle);

    print("<strong>Mise a jour effectuee.</strong><br />");
}
?>
<form action="" method="post" name="form1" id="form1">
<table width="600" border="0">
  <tr>
    <td width="129" valign="middle">Login :</td>
    <td width="461" valign="middle"><input type="text" name="login" id="login" /></td>
  </tr>
  <tr>
    <td valign="middle">Pass :</td>
    <td valign="middle"><input type="text" name="pass" id="pass" /></td>
  </tr>
  <tr>
    <td valign="middle"> </td>
    <td valign="middle"><input type="submit" name="subform" id="subform" value="Valider" /></td>
  </tr>
</table>
</form>
</body>
</html>
```

Listing 3. *Document d'erreur 401*

```
<html>

<head>
<title>Acces restreint</title>
</head>

<body>
Vous n'avez pas l'autorisation d'accéder à cette
partie du site.
<br />
Si vous avez perdu votre mot de passe, merci de
contacter l'administrateur qui vous réinitialisera
celui-ci.
<br /><br />
Merci.
</body>

</html>
```

par

```
// fonction pour crypter le mot de passe
$cset = "abcdefghijklmnopqrstuvwxyzABCDEFGHI-
JKLMNOPQRSTUVWXYZ0123456789./";
$salt = "";
for ($i=0; $i<CRYPT_SALT_LENGTH; $i++)
$salt .= substr($cset, rand() & 63, 1);
$passcrypt=crypt($pass, $salt);//nouveau mot
de passe crypté
// fin de la fonction
```

L'argument optionnel salt sera utilisé comme base pour le chiffrement. De même, souvent, sur les serveurs Windows, les mots de passe ne sont pas cryptés. Dans ce cas, remplacez par :

```
// fonction mot de passe
$passcrypt= $pass; //nouveau mot de passe non
crypté
// fin de la fonction
```

Exécutez ce fichier. Vous pouvez créer autant de couples login/pass que vous le souhaitez. Si un login existe déjà, son mot de passe sera écrasé. Bien entendu, il n'est pas conseillé de laisser ce fichier *creer.php* sur le serveur, ou alors de modifier le code (les chemins du fichier *.htpasswd*) et de l'enregistrer dans un autre répertoire lui même protégé par un unique login/pass que seul l'administrateur connaîtra. Le plus simple pour un usage classique est de supprimer ce fichier et de le remettre si besoin est.

Sur Internet

- *http://httpd.apache.org/docs/2.2/howto/auth.html* – Documentation Apache,
- *http://aspirine.org/htaccess.html* – Aspirine : génération de fichier .htaccess.

Voilà, votre dossier est maintenant parfaitement protégé. En effet, il n'existe pas de fonction de déchiffrement des mots de passe, car la fonction `crypt` utilise un algorithme à un seul sens (injection). Il ne reste plus qu'à créer la page vers laquelle les utilisateurs ayant entré un mauvais mot de passe seront redirigés. La première ligne de notre *.htaccess* étant

```
ErrorDocument 401 /var/www/vhosts/monsite.com/
httpdocs/401.php
```

La page de redirection se nomme *401.php* et se trouve dans le répertoire parent de notre répertoire à protéger (elle ne peut en effet pas être dans le répertoire à protéger puisque les utilisateurs arrivant sur cette page sont ceux qui ont entré un mauvais couple login/pass). Vous pouvez de même ajouter des lignes dans votre *.htaccess* pour personnaliser les autres pages d'erreur (404…). Le fichier *401.php* peut ressembler au Listing 3 (modifié selon vos besoins).

Conclusion

Voilà. Nous avons vu ici comment protéger simplement et efficacement un répertoire. Vous pouvez aller encore plus loin dans la configuration du fichier *.htaccess*, par exemple en bloquant certaines adresses IP, ou en créant des groupes d'utilisateurs… Le fichier Apache *.htaccess* est très puissant et permet de faire de nombreuses choses.

FRANCK CANONNE

Développeur PHP freelance, Franck Canonne a réalisé de nombreux sites web dynamiques avant de se spécialiser dans les webapps et la gestion de bases de données MySQL. Touche à tout, il préfère aujourd'hui utiliser des solutions libres (Drupal, Wordpress…) pour tout ce qui est développement web, et réserve le développement PHP/MySQL AJAX pour les applications riches de gestion de base de données. Conscient des contraintes des webapps, il optimise celles-ci en amont par une configuration spécifique du serveur Apache hôte.

Communication Flash/PHP

Flash, PHP, est-il encore bien nécessaire de les présenter ? Si largement répandus que chaque internaute les connaît au moins de nom ; leur succès n'a d'égal que leur efficacité. Chacun sa spécialité, chacun son utilisation, réunis pour créer des applications alliant l'ergonomie et le dynamisme des interfaces Flash aux capacités d'exploitation serveur offertes par PHP.

Cet article explique :

- Faire communiquer une application flash AS2 et un script PHP.

Ce qu'il faut savoir :

- L'environnement Flash.
- ActionScript 2.
- Le traitement d'informations reçues par POST ou GET dans un script PHP.

Les animations Flash sont des applications clientes ; s'exécutant exclusivement sur le client, il faut, pour les faire interagir avec un serveur distant (MySQL par exemple), un *quelque chose* faisant le lien. Pour créer ce *quelque chose* il existe plusieurs solutions : une solution payante, le *Flash Media Server* (FMS) permettant notamment la communication par socket ; des solutions gratuites équivalentes au FMS ; et une autre solution gratuite : le PHP via le protocole HTTP. Nous verrons les deux sortes de communication possibles entre Flash et PHP : le passage à l'instanciation de l'animation, et le passage lors de l'exécution de l'animation par requête. Ces deux types d'envoi de données sont très différents, autant dans leur utilisation que dans les méthodes employées. À vous de voir en fonction de vos besoins dans vos applications.

Comment ? Quand surtout !

Il y a deux manières pour envoyer des données à une application Flash, et elles se distinguent principalement par le moment où vous voulez faire passer les informations. La première méthode consiste à envoyer les informations avant que l'animation soit lancée : c'est à son instanciation que l'application cliente va assimiler ces informations. Principal avantage de cette méthode : très simple à mettre en place. Principal défaut: le flux d'information est unidirectionnel : l'application pourra recevoir les informations mais ne pourra pas en envoyer au serveur par ce biais. Dans la seconde méthode, l'application Flash va envoyer une requête HTTP pour récupérer une page qu'elle va ensuite lire et analyser pour obtenir les informations contenues. L'avantage cette fois-ci est que l'application peut insérer dans la requête des informations que le serveur recevra.

Passage à l'instanciation : les FlashVars

Lorsque vous insérez une animation Flash dans une page html, vous l'insérez dans une certaine *configuration* que vous définissez. Pour cela vous passez un certain nombre de paramètres : le chemin du fichier swf que vous voulez afficher, la qualité des graphismes, la couleur de fond de l'animation, et ce qui nous intéresse dans cet article : les *FlashVars*. L'envoi de données par cette

Listing 1. *Code HTML d'intégration d'une application Flash avec FlashVars*

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-
444553540000" codebase= HYPERLINK "http://fpdownload.
macromedia.com/pub/shockwave/cabs/flash/swflash.cab" \
l "version=8,0,0,0"http://fpdownload.macromedia.com/
pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0
width="550" height="400" id="Sans nom-1" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="exemple.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="flashvars" value="parametre1=valeur1&magazi
ne=PHP%20Solution" />
<embed src="exemple.swf" quality="high"
bgcolor="#ffffff" width="550" height="400"
name="exemple" align="middle" allowScriptAccess=
"sameDomain" flashvars="prenom=Jérôme&nom=Forget"
type="application/x-shockwave-flash" pluginspage="http://
www.macromedia.com/go/getflashplayer" />
</object>
```

Listing 2. *Exemple simple de fonction PHP pour l'utilisation des FlashVars*

```
<?php

$flashvars = "";

function flashvarsAdd($var, $value){
    return urlencode($var)."=".urlencode($value)."&";
}

$flashvars .= flashvarsAdd("question","Etes-vous lecteur de
phpSolutions ?");
$flashvars .= flashvarsAdd("labelVote1", "oui");
$flashvars .= flashvarsAdd("labelVote2", "non");

?>

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase=http://fpdownload.macromedia.com/pub/shockwave/
cabs/flash/swflash.cab#version=8,0,0,0 width="550"
height="400" id="Sans nom-1" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="animation.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="flashvars" value="<?php echo $flashvars ?>" />
<embed src="animation.swf" quality="high"
bgcolor="#ffffff" width="550" height="400" name="exemple"
align="middle" allowScriptAccess="sameDomain"
 flashvars="<?php echo $flashvars; ?>" type="application/x-
shockwave-flash" pluginspage="http://www.macromedia.com/go/
getflashplayer" />
</object>
```

Listing 3. *Utilisation de la classe LoadVars*

```
<?
var sender = new LoadVars();
var loader = new LoadVars();

//enregistrement des informations à envoyer
sender.nomParametre1 = "valeur1";
sender.nomParametre2 = "valeur2";

loader.onLoad = function(success) {
        if (success) {
                //affichage des données reçues
                trace(this.nomReponse1);
                trace(this.nomReponse2);

                }
        else{
                        trace("la requête a échoué");
                }
        };

sender.sendAndLoad("com.php",loader,"POST");
```

Listing 4. *index.php, affichage de l'application Flash*

```
<?php

$flashvars = "";

function flashvarsAdd($var, $value){
    return urlencode($var)."=".urlencode($value)."&";
}

$flashvars .= flashvarsAdd("question","Etes-vous lecteur
de phpSolutions ?");
$flashvars .= flashvarsAdd("labelVote1", "oui");
$flashvars .= flashvarsAdd("labelVote2", "non");

?>

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-
444553540000" codebase=http://fpdownload.macromedia.
com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0
width="550" height="400" id="Sans nom-1" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="animation.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="flashvars" value="<?php echo $flashvars ?>"
/>
<embed src="animation.swf" quality="high"
bgcolor="#ffffff" width="550" height="400"
name="exemple" align="middle" allowScriptAccess=
"sameDomain" flashvars="<?php echo $flashvars; ?>"
type="application/x-shockwave-flash" pluginspage="http://
www.macromedia.com/go/getflashplayer" />
</object>
```

Listing 5. *com.php, le script php qui va nous permettre le lien entre la bdd et l'application flash*

```
<?php

// Connexion BDD
$_sqlHost = "localhost";
$_sqlUser = "root";
$_sqlPassword = "";
$_sqlDataBase = "phpSolution ";
mysql_connect($_sqlHost, $_sqlUser, $_sqlPassword);
mysql_selectdb($_sqlDataBase);

// Fonction facilitant l'envoi d'informations
function flashvarsAdd($var, $value){
    return urlencode($var)."=".urlencode($value)."&";
}

// L'id du vote est contenu dans idChoix, on incrémente
donc le compteur pour ce vote
mysql_query("UPDATE sondage SET compte = compte + 1
WHERE 'idChoix' = ".$_REQUEST['idChoix']);

// On affiche les résultats de façon lisible pour
l'application Flash,
// exemple de ce que va afficher ce bloc:
choix1=9&choix2=7&total=16&
$total = 0;
$sqlRequest = mysql_query("SELECT * FROM sondage WHERE
1;");
while($sqlResult=mysql_fetch_array($sqlRequest)){
    echo flashvarsAdd("choix".$sqlResult['idChoix'],
$sqlResult['compte']);
    $total += $sqlResult['compte'];
}

echo flashvarsAdd("total", $total);

?>
```

méthode se fait à travers un paramètre appelé *FlashVars* et ayant pour valeur une chaîne de caractère contenant toutes les données que vous souhaitez faire passer. Cette chaîne est une combinaison de *paramètre=valeur* séparés par des *&*, les caractères spéciaux et d'échappements étant codés sous la forme *%+2* caractères hexadécimaux (pour faire simple, il s'agit de l'encodage que l'on retrouve pour l'envoi de données d'un formulaire html à une autre page par méthode `GET` aussi appelé *url-encoding*). Le Listing 1 vous présente l'ensemble du code HTML permettant l'intégration d'une application flash avec passage de paramètres à l'instanciation. Comme vous pouvez le remarquer on répète le passage des arguments au niveau du paramètre movie ainsi que dans la balise *embed.* Ceci est dû au fait que les navigateurs n'interprètent pas tous le paramètre *FlashVars* de la même façon. Par souci de compatibilité, il est donc conseillé de procéder ainsi.

Enfin, pour envoyer des informations à une application Flash par cette méthode, il suffira côté serveur d'écrire une fonction générant dynamiquement les balises d'intégration de l'animation munies du paramètre *flashvars* (cf Listing 2), et de lire ces paramètres à travers des variables flash classiques de nom *_root.nomParamètre*.

Passage pendant l'exécution

Dans ce genre d'échange, c'est l'animation qui va être l'initiatrice de la requête. L'échange d'information se fera comme ceci : le client (l'animation Flash) envoie une requête contenant les données que l'on souhaite faire parvenir au serveur. Le serveur (script php) analyse la requête et en dégage une réponse sous forme de page HTML. Enfin le client reçoit la réponse à sa requête contenant les informations transmises par le serveur. Pour envoyer et recevoir les informations, Flash a mis à disposition des bibliothèques rendant la création et l'analyse des requêtes totalement transparentes.

Du coté Actionscript

En AS2, c'est la classe LoadVars qui va nous permettre de communiquer avec un serveur distant, et ce de façon totalement transparente : son utilisation est montrée dans le Listing 3. Avec cette classe, l'envoi et la réception de données se font très simplement au travers d'instances, et on peut distinguer deux types d'instances (pour l'envoi et la réception) créées à partir du même constructeur. Pour établir un échange d'information avec un serveur, vous devrez tout d'abord créer deux instances : une servant pour l'envoi et une servant pour la réception. Celle servant pour l'envoi doit contenir les données que vous souhaitez transmettre au serveur. Pour cela vous n'aurez qu'à lui appliquer des propriétés portant le nom des paramètres : *instanceEnvoi.nomParametre = valeur*. Une fois toutes vos propriétés définies, vous n'aurez qu'à appeler la méthode `sendAndLoad` de cette instance dans laquelle vous spécifierez l'instance gérant le retour des données, l'adresse du script CGI auquel vous souhaitez transmettre les informations, et enfin la méthode avec laquelle vous souhaitez envoyer ces informations (`POST` ou `GET`). Comme vous l'aurez compris, pour gérer l'analyse des données reçues, vous devrez créer une deuxième instance. La contrainte principale de cette deuxième instance est qu'elle doit définir une méthode portant un nom bien spécifique, qui sera appelée une fois que les informations seront reçues. Cette méthode portant le nom *onLoad* contiendra toutes les données reçues qui sont stockées dans les propriétés de cette instance. Vous pourrez donc les obtenir à l'intérieur de la méthode par *this.nomParamètre*. Cette méthode reçoit comme paramètre une variable booléenne déterminant si la requête à bien été envoyée puis reçue correctement. Nous l'avons appelé *sucess*, elle peut être fausse si par exemple le fichier ou l'adresse n'a pas été trouvé. NB : pour faciliter le *debuging* d'une application flash, vous pouvez également dans la méthode `onLoad` récupérer l'ensemble de la réponse contenu dans `this.toString()`.

Listing 6. *Code AS2 de l'application Flash*

```
// choixA et choixB sont les 2 instances des boutons
qui permettront le vote
choixA.texte = _root.labelVote1;
choixB.texte = _root.labelVote2;

choixA.onPress = function() {
        appelServeur(1);
}
choixB.onPress = function() {
        appelServeur(2);
}

// NB : le champs question visible sur la Figure
1 est un champs textuel dynamique ayant pour
valeur _root.question qui se trouve être attribuée
directement par les flashvars, aucun code Actionscript
n'est nécessaire pour ce champs.
// resultatA et resultatB sont les deux barres
traduisant les résultats du vote,
// non visible pour l'instant puisque l'utilisateur
n'a pas encore voté.
resultatA._visible = false;
resultatB._visible = false;

function appelServeur(choix:Number){
        var sender = new LoadVars();
        var loader = new LoadVars();

        // on prépare à l'envoi l'id du vote qui a été
choisi
        sender.idChoix = choix;

        loader.onLoad = function(success) {
                if (success) {
                        //on donne aux barres des
dimensions proportionnelles aux votes reçus
                        _root.resultatA._height =
Math.round(300*this.choix1/this.total);
                        _root.resultatB._height =
Math.round(300*this.choix2/this.total);
                }
                else{
                        trace("la requete a
échoué");
                }
        };

        // on envoi les informations
        sender.sendAndLoad("com.php",loader,"POST");
        // on desactive les boutons et on affiche les
barres (toujours invisible puisque hauteur nulle)
        choixA._visible = false;
        choixA.enabled = false;
        choixB._visible = false;
        choixB.enabled = false;

        resultatA._visible = true;
        resultatB._visible = true;

};
```

Du coté PHP

Afin que l'application flash puisse interpréter correctement les informations, les données envoyées par le serveur, il faut qu'elles soient mises sous la forme *url-encoding*. Pour cela il existe une fonction en php portant le même nom : *urlencode*. Cette fonction retourne la chaîne de caractère passée en paramètre en chaîne de caractère encodée sous la forme *url-encoding*.

Application, énoncé du problème

Pour mettre en application tout ce que nous venons de voir, nous allons prendre un exemple simple. Imaginons que nous voulions faire une application Flash permettant de créer des sondages (à choix binaire dans notre cas) à la volée. Posons comme cahier des charges que les intitulés de la question et des réponses possibles soient paramétrables et que l'application puisse répondre à deux besoins : répondre à la question posée et voir sous forme d'une animation Flash le résultat global du sondage. Dans ce cas les deux types d'échange de données seront utilisés. Nous emploierons le passage à l'instanciation pour paramétrer l'interface, c'est-à-dire définir les intitulés, et nous favoriserons le passage pendant l'exécution pour envoyer la réponse du sondé et récupérer les résultats globaux, avant de les utiliser pour les afficher dans l'interface.

Choix technologiques

Vu le cahier des charges que nous nous sommes imposé, quel que soit le contexte dans lequel l'animation Flash tournera, le libellé des questions et des réponses possibles restera le même. Dans ce cas, la communication des libellés des questions est unidirectionnelle ; nous utiliserons donc les *FlashVars*.

Ensuite, une fois que l'utilisateur aura fait son choix pour le sondage, l'animation Flash devra l'envoyer au serveur pour que le choix soit comptabilisé au sein de la base de données. Il faudra donc utiliser la classe LoadVars. Nous profiterons également de cet envoi pour récupérer les résultats globaux du sondage et les afficher sous forme d'animation. Un des procédé les plus simples et les plus efficaces graphiquement est la représentation sous forme de bâtons : pour chaque choix on prend un carré de couleur uniforme différente et on lui donne une hauteur représentant le rapport des voix qu'il a remporté. Vous trouverez l'ensemble du code ainsi que les explications en commentaires dans les listings **4, 5 et 6, ainsi que des illustrations dans les figures 1,2 et 3**

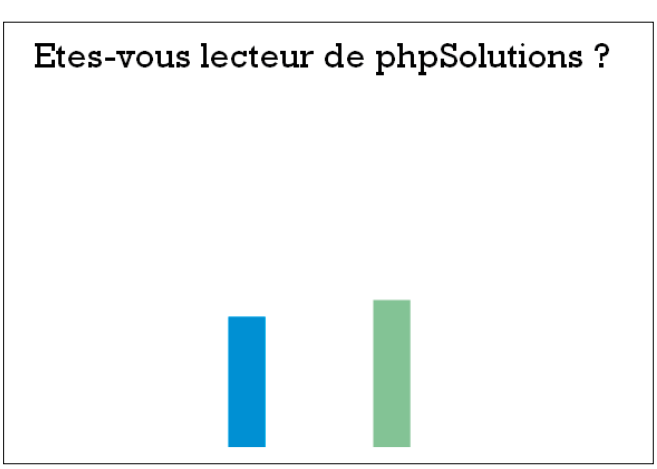


Figure 3. *Affichage du résultat*

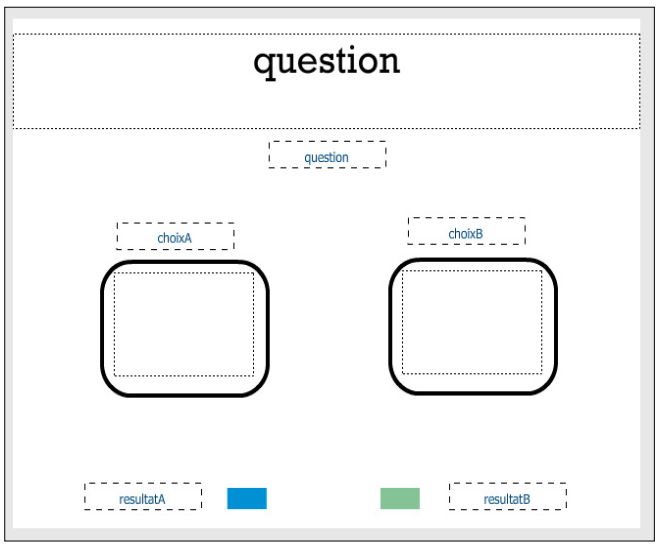


Figure 1. *Dans l'éditeur Flash*

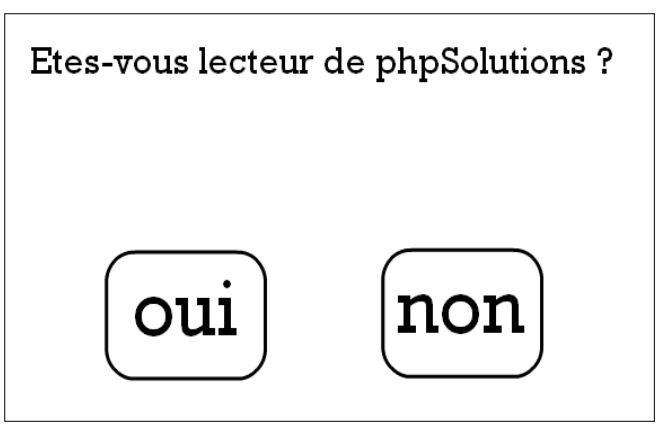


Figure 2. *Avant que l'utilisateur ne clique sur un des boutons*

Conclusion

Maintenant vous savez faire communiquer une application Flash et un serveur php. Vous allez pouvoir enrichir vos interfaces. Nombreuses sont les applications qui s'offrent à vous : envoi de mail, processus d'identification, rendu graphique de données…

Rappelons enfin que 98% des clients internet actuels possèdent le plugin permettant de lire les fameux fichiers *.swf*. Lancez vous !

JÉRÔME MANCHON

Jérôme Manchon poursuit actuellement des études d'ingénieur à l'ENSAM tout en menant en parallèle des projets personnels afin d'élargir et approfondir ses connaissances dans le milieu du PHP et de la programmation. Jerome.manchon@gmail.com

Usages avancés des sessions avec la POO

Aujourd'hui, la grande majorité des sites dynamiques écrits en PHP utilisent le mécanisme natif des sessions. Dans certaines situations, ce moyen de stockage des sessions n'est pas le plus approprié et peut se voir remplacé par un moteur de stockage différent, une base de données par exemple.

Cet article explique :

- Comment manipuler les sessions sous forme d'objets PHP.
- Comment déplacer le stockage des sessions vers une base de données MySQL.

Ce qu'il faut savoir :

- Comment fonctionne le mécanisme des sessions PHP.
- Avoir des bases de programmation orientée objet.

L'objectif de ce cours consiste dans un premier temps à développer une approche orientée objet des sessions natives de PHP afin de simplifier leur utilisation. Puis nous chercherons à étendre ce mécanisme en vue de stocker les sessions dans une base de données MySQL sans avoir à modifier l'API principale en profondeur.

Veuillez noter cependant que cet article ne couvre pas les bases du système natif des sessions. Le numéro du mois de juin 2010 de ce magazine dédie un article complet à ce sujet et nous ne pouvons que vous inviter à le lire attentivement avant de vous plonger dans les prochaines lignes.

Rappels sur les sessions PHP

Le mécanisme des sessions PHP a été instauré dans le langage dans le but de proposer un moyen de persister des informations entre les requêtes HTTP. En effet, le protocole HTTP est *sans état* (*stateless*) et fonctionne en mode *non connecté*. Cela signifie que deux requêtes HTTP successives émises depuis la même adresse IP sont complètement indépendantes l'une de l'autre.

La reconnaissance de l'utilisateur entre deux requêtes HTTP est réalisée au moyen d'un cookie de session contenant un jeton unique d'identification. Selon la configuration de PHP (`session.use_cookies`, `session.use_only_cookies` et `session.use_trans_id`), l'identifiant de session peut également être propagé par l'intermédiaire d'une variable complémentaire dans l'url (`GET`).

En terme d'API, PHP fournit une vingtaine de fonctions (*http://fr.php.net/manual/en/ref.session.php*) utiles à la configuration et la manipulation des sessions. A cette liste de fonctions s'ajoute le tableau superglobal `$_SESSION` chargé d'accueillir les données de session. Ce tableau est accessible en lecture et en écriture. Il est automatiquement initialisé au démarrage du script.

Le Listing 1, *page1.php*, donne un exemple d'usage simple de l'API des sessions. La première ligne appelle la fonction `session_name()` avec le paramètre phpsolutions qui fixe le nom de la session, et donc du cookie de session. Par défaut, le nom de la session est fixé avec la valeur *PHPSESSID*. Ensuite, la fonction `session_start()` démarre (ou restaure) la session courante tandis que la ligne suivante, `session_regenerate_id()`, force PHP à recréer un nouvel identifiant de session afin d'éviter une éventuelle fixation de la session. Enfin la dernière ligne déclare une variable de session persistante, *color*, dans le tableau `$_SESSION` et l'initialise avec la valeur *blue*.

Le Listing 2, *page2.php*, réalise les mêmes opérations que le morceau de code de la page 1. La différence se situe à la dernière ligne qui lit la valeur de la variable de session *color* et l'affiche sur la sortie standard. Bien sûr, les trois premières lignes communes aux deux fichiers devraient être mutualisées dans un fichier séparé ou dans une fonction utilisateur afin d'éviter la duplication du code.

La plupart des développeurs PHP interrompent leur usage des sessions PHP à ces quelques lignes de code.

Listing 1. *Démarrage d'une session et écriture d'une variable de session*

```
<?php

session_name('phpsolutions');
session_start();
session_regenerate_id();

$_SESSION['color'] = 'blue';
```

Listing 2. *Restauration d'une session et lecture d'une variable de session*

```
<?php

session_name('phpsolutions');
session_start();
session_regenerate_id();

echo $_SESSION['color'];
```

Listing 3. *Définition de la classe SessionStorage*

```
<?php

class SessionStorage
{
  static protected $isStarted = false;

  protected $name;
  protected $useCookies = true;
  protected $options = array();

  public function __construct($name, array $options =
array())
  {
    $this->name = $name;
    $this->useCookies = (boolean) ini_get('session.use_
cookies');

    $cookie = session_get_cookie_params();

    $this->options = array_merge(array(
      'session_auto_start'    => true,
      'session_cache_limiter' => 'nocache',
      'session_cache_expire' => 180,
      'session_save_path' => null,
      'session_id' => null,
      'session_cookie_lifetime' => $cookie['lifetime'],
      'session_cookie_path' => $cookie['path'],
      'session_cookie_domain' => $cookie['domain'],
      'session_cookie_secure' => $cookie['secure'],
      'session_cookie_httponly' => isset($cookie['httponl
y']) ? $cookie['httponly'] : false
    ), $options);

    if (!ini_get('session.auto_start'))
    {
      session_name($this->name);

      if ($this->options['session_id'])
      {
        session_id($this->options['session_id']);
      }

      if ($this->options['session_save_path'])
      {
        session_save_path($this->options['session_
save_path']);
      }

      if ($this->options['session_cache_limiter'])
      {
        session_cache_limiter($this->options['session_
cache_limiter']);
      }

      if ('nocache' !== $this->options['session_cache_
limiter']
          && $this->options['session_cache_expire'])
      {
        session_cache_expire($this->options['session_
cache_expire']);
      }

      if ($this->useCookies)
      {
        session_set_cookie_params(
          $this->options['session_cookie_lifetime'],
          $this->options['session_cookie_path'],
          $this->options['session_cookie_domain'],
          $this->options['session_cookie_secure'],
          $this->options['session_cookie_httponly']
        );
      }

      if ($this->options['session_auto_start'] &&
!self::$isStarted)
      {
        session_start();
        session_regenerate_id();
        self::$isStarted = true;
      }
    }
  }

  public function destroy()
  {
    $destroyed = session_destroy();

    if ($destroyed)
    {
      if ($this->useCookies)
      {
        setcookie($this->name, '', time() - 42000,
          $this->options['session_cookie_path'],
          $this->options['session_cookie_domain'],
          $this->options['session_cookie_secure'],
          $this->options['session_cookie_httponly']
        );
      }

      $this->clear();
    }

    return $destroyed;
  }

  public function write($key, $value)
  {
    $_SESSION[$key] = $value;
  }

  public function read($key)
  {
    return array_key_exists($key, $_SESSION) ? $_
SESSION[$key] : null;
  }

  public function remove($key)
  {
    if (null !== $value = $this->read($key))
    {
      unset($_SESSION[$key]);
    }

    return $value;
  }

}
```

Listing 4. *Code de configuration de l'hôte virtuel Apache*

```
<VirtualHost *:80>
  ServerName www.phpsolutions.local
  DocumentRoot "/Users/Hugo/Developpment/PHPSolutions/www"
  DirectoryIndex index.php
  <Directory "/Users/Hugo/Developpment/PHPSolutions/www">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

Listing 5. *Ligne à ajouter au fichier hosts de la machine*

```
127.0.0.1 www.phpsolutions.local
```

Listing 6. *Code principal du fichier www/index.php*

```
<?php

// listing 6
require _ _ DIR _ _ .'/../src/SessionStorage.php';

function isValidColor($color) {

  return in _ array($color, array(
    'rouge', 'vert',
    'bleu', 'violet',
    'orange', 'noir'
  ));
}

$session = new SessionStorage('phpsolmag', array(
  'session _ save _ path' => _ _ DIR _ _ .'/../tmp'
));

$color = filter _ input(INPUT _ POST, 'color');

if (isValidColor($color)) {

  $session->write('favorite _ color', $color);
  header('Location: http://www.phpsolutions.local/index.
php');
  exit;
}

$color = $session->read('favorite _ color');

?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
    <title>Bienvenue</title>
  </head>
  <body>
    <form action="index.php" method="post">
      <fieldset>
        <legend>Personnalisation</legend>
        <div>
          <label for="color">Couleur favorite</label>
          <select id="color" name="color">
            <option value="rouge">Rouge</option>
            <option value="vert">Vert</option>
            <option value="bleu">Bleu</option>
            <option value="violet">Violet</option>
            <option value="orange">Orange</option>
          </select>
          <button type="submit">Ok</button>
        </div>
      </fieldset>
    </form>
    <?php if (!$color) : ?>
    <p>
      Vous n'avez pas encore choisi de couleur.
    </p>
    <?php else: ?>
    <p>
      Votre couleur favorite est le <strong><?php echo
$color ?></strong>.
    </p>
    <?php endif ?>
  </body>
</html>
```

Ces dernières font le travail qu'on leur demande et le font bien. Néanmoins, cette approche procédurale du code peut s'avérer contraignante lorsque l'application se compose de plusieurs centaines de fichiers et des milliers de lignes de code.

En effet, comment s'assurer que le code fonctionne bien ? Comment assurer sa réutilisation dans un autre projet ? Comment déplacer le stockage des sessions vers un autre système comme une base de données relationnelle ? Pour répondre à toutes ces questions, il convient de transformer l'approche procédurale du code en approche orientée objet. En somme, il s'agit de percevoir la session comme un objet métier PHP sur lequel s'appliquent des actions.

L'objectif de la seconde partie de ce cours consiste à développer une classe PHP représentant une session PHP. Celle-ci encapsulera les actions nécessaires à la configuration de la session, la lecture de variable ainsi que la fixation de variable.

Vers une approche orientée objet

Comme son nom l'indique, l'approche orientée objet nécessite de faire appel à des objets. Par définition, on dit qu'un objet est une instance d'une classe. Dit autrement, une classe est une sorte de moule à partir duquel sont issus des objets. Il s'agit donc de créer une classe `SessionStorage` chargée d'encapsuler la définition de la session courante.

Une session se définit également par un jeu de propriétés et d'actions. Les propriétés sont les attributs d'un objet, c'est-à-dire les variables internes de l'objet. La classe `SessionStorage`, présentée au Listing 3, accueille trois propriétés :

- `$name`, une chaîne contenant le nom de la session (ex: *phpsolutions*),
- `$useCookies`, une valeur booléenne indiquant si la session courante transmet l'identifiant de session par un cookie,
- `$options`, un tableau d'options de configuration de la session.

La classe `SessionStorage` du Listing 3 encapsule également un certains nombre de méthodes répondant chacune un besoin particulier. Les paragraphes qui suivent présentent une à une ces méthodes. Notez que le code de la classe `SessionStorage` est largement inspiré de celui du framework Open Source Symfony.

La première méthode, `__construct()`, est le constructeur de la classe. Elle a pour double rôle de construire l'objet à l'appel du mot-clé *new*, mais surtout d'initialiser ce dernier. Dans le cadre de notre session, le constructeur réalise synthétiquement les trois étapes suivantes :

I. Configurer le nom de la session.
II. Configurer la session en fonction des valeurs transmises dans le tableau d'options passé en second paramètre.
III. Démarrer ou restaurer la session.

La méthode `read()`, comme son nom l'indique, sert à lire une variable de session depuis le tableau `$_SESSION`. La fonction `array_key_exists()` s'assure ici que le nom de la variable de session passée en paramètre de la méthode `read()` se trouve bien dans le tableau. Le test est important afin d'éviter de lever un avertissement au cas où l'on essaierait d'atteindre un index inexistant dans le tableau `$_SESSION`.

L'écriture de nouvelles valeurs dans la session est délégué à la méthode `write()` qui accepte deux paramètres : le nom de la variable de session et sa valeur. La méthode `remove()` offre la possibilité de supprimer une variable de session en lui passant en paramètre le nom de cette dernière.

Enfin, la méthode `destroy()` a la responsabilité de détruire la session. Cette méthode ne se limite pas seulement à l'appel de la fonction `session_destroy()`. En effet, elle vérifie également si l'identifiant de session est propagé par un cookie. Si c'est le cas, un cookie vide et expiré est envoyé au navigateur.

Mise en pratique

A présent la classe `SessionStorage` est prête. Il convient de la tester à l'aide d'un petit exemple pratique. Il s'agit d'une application web composée d'une page web, *index.php*, divisée en deux parties : un formulaire et un paragraphe de texte. Le formulaire demande à l'utilisateur de choisir sa couleur favorite tandis que le paragraphe se charge d'afficher cette dernière.

Commençons tout d'abord par créer un dossier *PHPSolutions/* sur la machine. Celui-ci contiendra la structure initiale de l'application décrite ci-dessous :

- Le dossier *src/* accueille la classe `SessionStorage`.
- Le dossier *tmp/* stocke les fichiers des sessions. Ce dossier doit donc être accessible en lecture comme en écriture par PHP.
- Le dossier *www/* abrite les tous fichiers accessibles depuis un navigateur web comme le script principal *index.php*.

Une fois l'architecture de l'application en place, il faut configurer le serveur web Apache afin d'empêcher l'accès aux dossiers *src/* et *tmp/* depuis un navigateur web. En résumé, il s'agit de définir le dossier *www/* comme étant la racine web du projet. C'est une bonne pratique de développement web de séparer les fichiers accessibles publiquement de ceux qui ne doivent pas l'être. Ne rendez publiques que les fichiers que vous souhaitez atteindre depuis votre navigateur web. Tous les autres doivent être placés au niveau supérieur afin de les protéger des utilisateurs mal intentionnés.

Pour y parvenir, il suffit de définir un nouvel *hôte virtuel* (*virtual host*) dans le fichier de configuration *httpd.conf* d'Apache. Dans les versions récentes d'Apache, il s'agit du fichier *httpd-vhosts.conf* dont l'inclusion est parfois mise en commentaire à la fin du fichier *httpd.conf*.

Listing 7. *Création d'une base de données depuis un client MySQL en ligne de commande*

```
$ mysql -u root -p
$ Enter password:

Welcome to the MySQL monitor.  Commands end with ; or \
g.
Your MySQL connection id is 9
Server version: 5.1.49 Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates.
All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This
is free software,
and you are welcome to modify and redistribute it under
the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the
current input statement.

mysql>create databse php_solutions;
```

Listing 8. *Code SQL de création de la table php_session*

```
DROP TABLE IF EXISTS 'php_session';
CREATE TABLE 'php_session' (
  'id' integer(11) NOT NULL AUTO_INCREMENT,
  'sess_id' varchar(50) NOT NULL UNIQUE,
  'sess_data' text NOT NULL,
  'sess_time' integer(14) UNSIGNED NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Le Listing 4 présente le code de l'hôte virtuel à placer à la fin du fichier *httpd.conf* avant de redémarrer le serveur Apache. Les chemins absolus de l'hôte virtuel sont bien évidemment à remplacer par ceux équivalents au système hébergeant l'application. La directive *ServerName* définit le nom de domaine à utiliser pour atteindre le fichier *index.php* du répertoire *www/*. Les directives *AllowOverride* et *Allow from all* indiquent respectivement que la configuration Apache du répertoire *www/* peut être surchargée à l'aide d'un fichier *.htaccess* et que ce dossier est accessible par tout le monde.

Le domaine *www.phpsolutions.local* est un domaine local, c'est à dire qu'il pointe sur l'adresse IP locale *127.0.0.1*. Il faut donc indiquer à la machine comment résoudre ce nom de domaine localement. Pour ce faire, il suffit d'ajouter la ligne du listing 5 au fichier *hosts*. Ce fichier se trouve dans le dossier */etc* sur les environnements Linux et dans *C:\Windows\System32\drivers\etc* sur les plateformes Microsoft.

Enfin, le Listing 6 donne le code du fichier *www/index.php*. Ce fichier se compose de deux parties : le code PHP qui traite la requête utilisateur et la session, puis le code HTML destiné à l'affichage des données.

La première ligne de ce fichier inclut la définition de la classe `SessionStorage` afin de pouvoir l'instancier quelques lignes plus bas. Le bloc de lignes suivantes déclare une fonction utilisateur `isValidColor()`

Listing 9. *Définition de la classe MySQLSessionStorage*

```
<?php

require _ _DIR_ _ .'/SessionStorage.php';

class MySQLSessionStorage extends SessionStorage
{
  protected $dbh;

  public function _ _construct($name, PDO $dbh, array
$options = array())
  {
    $this->dbh = $dbh;

    // Table that stores session data
    $options = array_merge(array(
      'db_table'    => 'php_session',
      'db_id_col'   => 'sess_id',
      'db_data_col' => 'sess_data',
      'db_time_col' => 'sess_time'
    ), $options);

    $options['session_auto_start'] = false;

    parent::_ _construct($name, $options);

    // Ask PHP to use this class to handle sessions
instead of
    // its native behavior
    session_set_save_handler(
      array($this, 'sessionOpen'),
      array($this, 'sessionClose'),
      array($this, 'sessionRead'),
      array($this, 'sessionWrite'),
      array($this, 'sessionDestroy'),
      array($this, 'sessionGC')
    );

    session_start();
  }

  public function sessionClose()
  {
    return true;
  }

  public function sessionOpen($path = null, $name = null)
  {
    return true;
  }

  public function sessionDestroy($id)
  {
    // get table/column
    $db_table  = $this->options['db_table'];
    $db_id_col = $this->options['db_id_col'];

    // delete the record associated with this id
    $sql = 'DELETE FROM '.$db_table.' WHERE '.$db_id_
col.'= ?';

    try
    {
      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $id, PDO::PARAM_STR);
      $stmt->execute();
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to destroy the
session. Message: %s', $e->getMessage()));
    }

    return true;
  }

  public function sessionGC($lifetime)
  {
    // get table/column
    $db_table    = $this->options['db_table'];
    $db_time_col = $this->options['db_time_col'];

    // delete the record associated with this id
    $sql = 'DELETE FROM '.$db_table.' WHERE '.$db_
time_col.' < '.(time() - $lifetime);

    try
    {
      $this->dbh->query($sql);
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to clean
expired sessions. Message: %s', $e->getMessage()));
    }

    return true;
  }

  public function sessionRead($id)
  {
    // get table/columns
    $db_table    = $this->options['db_table'];
    $db_data_col = $this->options['db_data_col'];
    $db_id_col   = $this->options['db_id_col'];
    $db_time_col = $this->options['db_time_col'];

    try
    {
      $sql = 'SELECT '.$db_data_col.' FROM '.$db_
table.' WHERE '.$db_id_col.'=?';

      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $id, PDO::PARAM_STR, 255);

      $stmt->execute();
      $sessionRows = $stmt->fetchAll(PDO::FETCH_NUM);
      if (1 === count($sessionRows))
      {
        return $sessionRows[0][0];
      }
      else
      {
        // session does not exist, create it
        $sql = 'INSERT INTO '.$db_table.'('.$db_id_
col.', '.$db_data_col.', '.$db_time_col.') VALUES (?,
?, ?)';

        $stmt = $this->dbh->prepare($sql);
        $stmt->bindParam(1, $id, PDO::PARAM_STR);
        $stmt->bindValue(2, '', PDO::PARAM_STR);
        $stmt->bindValue(3, time(), PDO::PARAM_INT);
        $stmt->execute();

        return '';
      }
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to read session
data. Message: %s', $e->getMessage()));
    }
  }

  public function sessionWrite($id, $data)
  {
    // get table/column
    $db_table    = $this->options['db_table'];
    $db_data_col = $this->options['db_data_col'];
    $db_id_col   = $this->options['db_id_col'];
    $db_time_col = $this->options['db_time_col'];

    $sql = 'UPDATE '.$db_table.' SET '.$db_data_col.' =
?, '.$db_time_col.' = '.time().' WHERE '.$db_id_col.'=
?';

    try
    {
      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $data, PDO::PARAM_STR);
      $stmt->bindParam(2, $id, PDO::PARAM_STR);
      $stmt->execute();
    }
    catch (PDOException $e)
```

Listing 9. *Définition de la classe MySQLSessionStorage – suite*

```
    {
      throw new Exception(sprintf('Unable to write
session data. Message: %s', $e->getMessage()));
    }

    return true;
  }
}
```

Listing 10. *Code principal du fichier www/index.php (version 2)*

```
<?php

require _ _ DIR _ _ .'/../src/MySQLSessionStorage.php';

function isValidColor($color) {

  return in _ array($color, array(
    'rouge', 'vert',
    'bleu', 'violet',
    'orange', 'noir'
  ));
}

$dbh = null;
try {

  $dbh = new PDO(
    'mysql:dbname=php _ solutions;host=127.0.0.1',
    'root',
    '',
    array(PDO::MYSQL _ ATTR _ INIT _ COMMAND => "SET NAMES
'UTF8'")
  );

  $dbh->setAttribute(PDO::ATTR _ ERRMODE, PDO::ERRMODE _
EXCEPTION);

} catch (PDOException $e) {
  echo 'Unable to connect to MySQL:<br/>';
  echo $e->getMessage();
  die;
}

$session = new MySQLSessionStorage('phpsolmag', $dbh);

$color = filter _ input(INPUT _ POST, 'color');

if (isValidColor($color)) {

  $session->write('favorite _ color', $color);
  header('Location: http://www.phpsolutions.local/index.
php');
  exit;
}

?>
<!-- suite du code ... -->
require _ _ DIR _ _ .'/SessionStorage.php';

class MySQLSessionStorage extends SessionStorage
{
  protected $dbh;

  public function _ _ construct($name, PDO $dbh, array
$options = array())
  {
    $this->dbh = $dbh;

    // Table that stores session data
    $options = array _ merge(array(
      'db _ table'    => 'php _ session',
      'db _ id _ col'   => 'sess _ id',
      'db _ data _ col' => 'sess _ data',
      'db _ time _ col' => 'sess _ time'
    ), $options);

    $options['session _ auto _ start'] = false;

    parent:: _ _ construct($name, $options);

    // Ask PHP to use this class to handle sessions
instead of
    // its native behavior
    session _ set _ save _ handler(
      array($this, 'sessionOpen'),
      array($this, 'sessionClose'),
      array($this, 'sessionRead'),
      array($this, 'sessionWrite'),
      array($this, 'sessionDestroy'),
      array($this, 'sessionGC')
    );

    session _ start();
  }

  public function sessionClose()
  {
    return true;
  }

  public function sessionOpen($path = null, $name = null)
  {
    return true;
  }

  public function sessionDestroy($id)
  {
    // get table/column
    $db _ table  = $this->options['db _ table'];
    $db _ id _ col = $this->options['db _ id _ col'];

    // delete the record associated with this id
    $sql = 'DELETE FROM '.$db _ table.' WHERE '.$db _ id _
col.'= ?';

    try
    {
      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $id, PDO::PARAM _ STR);
      $stmt->execute();
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to destroy the
session. Message: %s', $e->getMessage()));
    }

    return true;
  }

  public function sessionGC($lifetime)
  {
    // get table/column
    $db _ table    = $this->options['db _ table'];
    $db _ time _ col = $this->options['db _ time _ col'];

    // delete the record associated with this id
    $sql = 'DELETE FROM '.$db _ table.' WHERE '.$db _
time _ col.' < '.(time() - $lifetime);

    try
    {
      $this->dbh->query($sql);
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to clean
expired sessions. Message: %s', $e->getMessage()));
    }

    return true;
  }

  public function sessionRead($id)
  {
    // get table/columns
    $db _ table    = $this->options['db _ table'];
    $db _ data _ col = $this->options['db _ data _ col'];
    $db _ id _ col   = $this->options['db _ id _ col'];
    $db _ time _ col = $this->options['db _ time _ col'];

    try
```

Listing 10. *Code principal du fichier www/index.php (version 2) – suite*

```
    {
      $sql = 'SELECT '.$db_data_col.' FROM '.$db_
table.' WHERE '.$db_id_col.'=?';

      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $id, PDO::PARAM_STR, 255);

      $stmt->execute();
      $sessionRows = $stmt->fetchAll(PDO::FETCH_NUM);
      if (1 === count($sessionRows))
      {
        return $sessionRows[0][0];
      }
      else
      {
        // session does not exist, create it
        $sql = 'INSERT INTO '.$db_table.'('.$db_id_
col.', '.$db_data_col.', '.$db_time_col.') VALUES (?,
?, ?)';

        $stmt = $this->dbh->prepare($sql);
        $stmt->bindParam(1, $id, PDO::PARAM_STR);
        $stmt->bindValue(2, '', PDO::PARAM_STR);
        $stmt->bindValue(3, time(), PDO::PARAM_INT);
        $stmt->execute();

        return '';
      }
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to read
session data. Message: %s', $e->getMessage()));
    }
  }

  public function sessionWrite($id, $data)
  {
    // get table/column
    $db_table    = $this->options['db_table'];
    $db_data_col = $this->options['db_data_col'];
    $db_id_col   = $this->options['db_id_col'];
    $db_time_col = $this->options['db_time_col'];

    $sql = 'UPDATE '.$db_table.' SET '.$db_data_col.'
= ?, '.$db_time_col.' = '.time().' WHERE '.$db_id_
col.'= ?';

    try
    {
      $stmt = $this->dbh->prepare($sql);
      $stmt->bindParam(1, $data, PDO::PARAM_STR);
      $stmt->bindParam(2, $id, PDO::PARAM_STR);
      $stmt->execute();
    }
    catch (PDOException $e)
    {
      throw new Exception(sprintf('Unable to write
session data. Message: %s', $e->getMessage()));
    }

    return true;
  }
}
```

qui a pour rôle de s'assurer que la valeur passée en paramètre est une couleur attendue parmi la liste de valeurs possibles.

A partir de la ligne 9, la classe de session est instanciée afin de produire un objet `$session` dont le nom du cookie de session est *phpsolmag*. L'option `session_save_path` quant à elle définit le dossier dans lequel les fichiers des sessions seront écrites. Ici, il s'agit du répertoire *tmp/* de notre application.

Ensuite, la fonction `filter_input()` récupère la valeur de la variable `$_POST['color']` issue du formulaire. Si aucune valeur n'a été transmise, alors la variable `$color` contiendra la valeur *null* par défaut.

Puis, le dernier bloc conditionnel s'assure que la valeur de la couleur transmise en PHP est valide. Si c'est le cas, on fait appel à la session pour y stocker la valeur de la couleur dans la variable de session `favorite_color`. Une redirection est ensuite déclenchée à l'aide de la fonction `header()`.

Enfin, la dernière ligne du bloc de code PHP tente de récupérer la valeur de la variable de session `favorite_color`. Si la valeur existe alors elle est affichée dans la partie HTML, sinon un message demande à l'utilisateur de bien vouloir choisir sa couleur préférée parmi la sélection proposée.

Pour tester l'application, il suffit de se rendre dans un navigateur et de saisir l'url *http://www.phpsolutions.local*. La Figure 1 montre l'application à sa première exécution tandis que la seconde affiche le résultat après avoir choisi la couleur bleu dans la liste déroulante. Le répertoire *tmp/* contient à présent les fichiers de session dont le nom contient l'identifiant de session transmis dans le cookie.

Cette petite application prouve le fonctionnement de l'objet *SessionStorage*. Le mécanisme des sessions a été entièrement réécrit à l'aide d'une classe PHP. Mise à part la syntaxe orientée objet, cette nouvelle manière de percevoir la session sous forme d'un objet métier apporte de nombreux avantages.

Le premier avantage est sans aucun doute la réutilisabilité du code. En effet, il suffit de copier le fichier *SessionStorage.php* dans les différents projets afin de bénéficier de cette nouvelle API plus verbeuse et plus facile d'utilisation.

D'autre part, grâce à la programmation orientée objet, le code est désormais complètement testable grâce aux tests unitaires. Il aurait ainsi été possible d'écrire une série de tests unitaires dans *PHPUnit* pour s'assurer que les objets *SessionStorage* se comportent bien comme on s'y attend.

Enfin, l'implémentation technique (le tableau `$_SESSION` et toutes les fonctions `session_*()`) a été entièrement masquée par une API de plus haut niveau grâce à la classe et ses méthodes. Par conséquent, tous les comportements implémentés ici peuvent être redéfinis ou surchargés et c'est d'ailleurs tout le sujet de la partie suivante. En effet, il s'agira de translater le stockage des sessions vers une base de données MySQL sans nécessiter une importante mise à jour du code de l'application.

Etendre le mécanisme des sessions

Aujourd'hui, les développeurs web doivent faire face à de plus en plus de contraintes lorsqu'ils travaillent pour un client. Ils doivent en effet prendre en compte l'existant de ce dernier. De plus, avec la professionnalisation

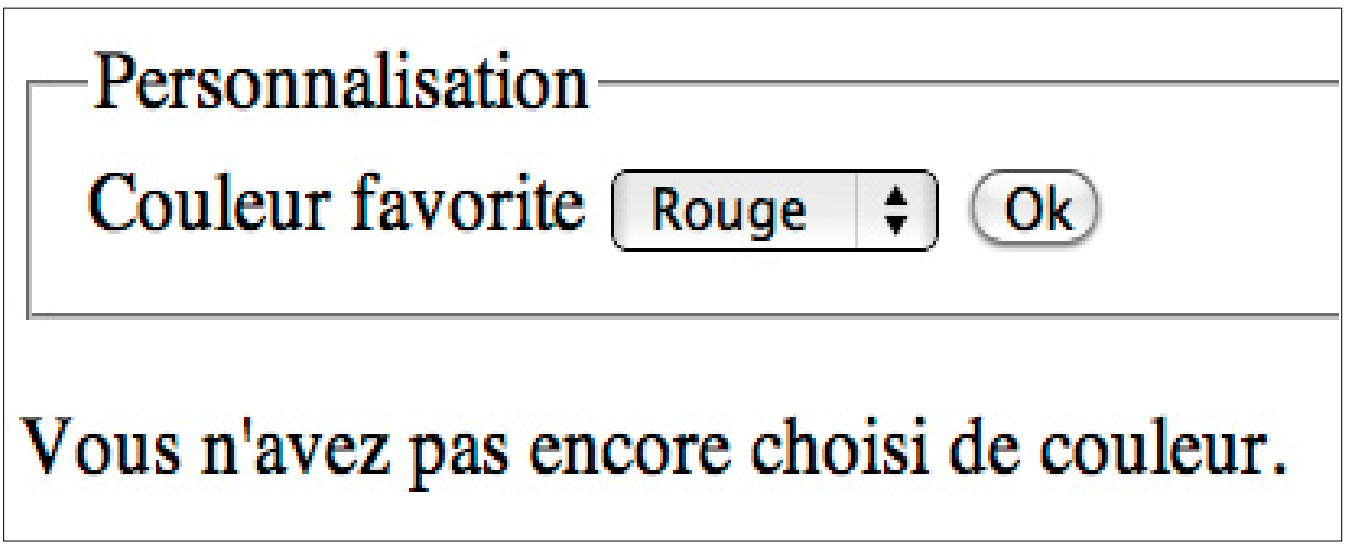


Figure 1. *A l'initialisation de l'application*

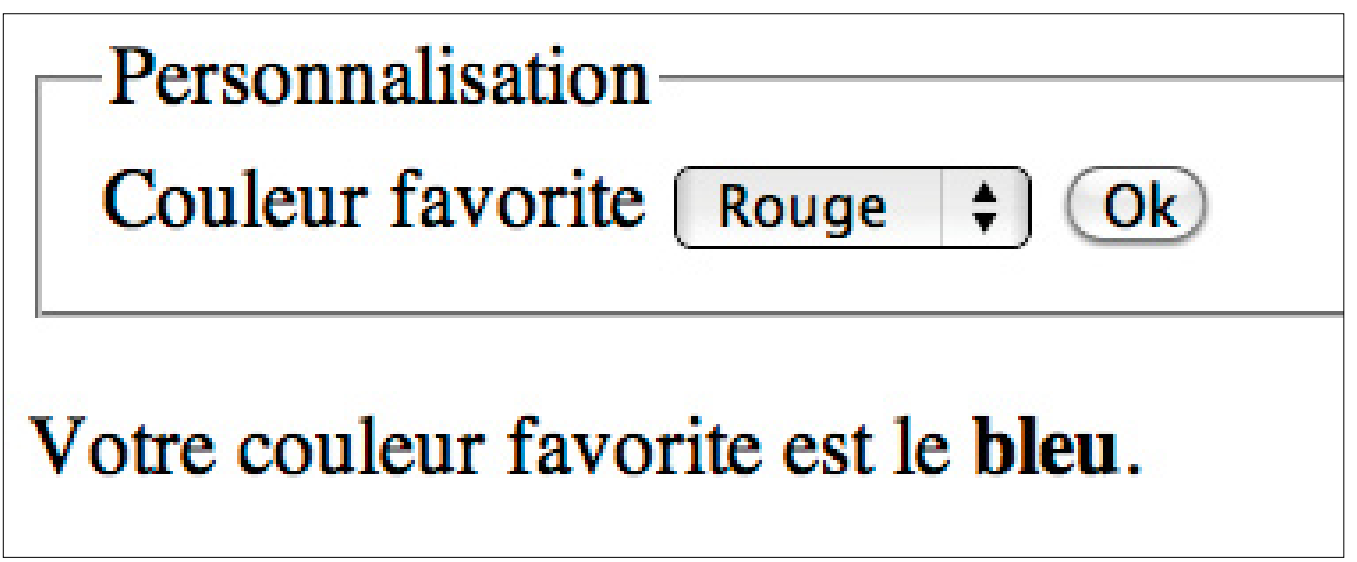


Figure 2. *Après enregistrement d'une variable de session*

et l'industrialisation des développements web, il n'est plus rare d'avoir à intégrer une application web dans un environnement technique existant et parfois très contraignant. Il convient donc d'adopter des stratégies au niveau du code source pour s'assurer que ce dernier s'adaptera lui même à l'environnement technique. Par exemple, écrire une application capable de se connecter à une base de données MySQL ou PostGreSQL sans avoir à toucher tout le reste du code.

Contraintes et limites des sessions natives

Le combat est aussi le même pour les sessions. Dans la majorité des cas, le mécanisme natif de gestion des sessions de PHP suffit à lui même car l'application web fonctionne sur un seul serveur web. Cependant, la donne change si l'on est amené à installer une seconde machine LAMP capable d'absorber une partie de la charge.

Imaginons simplement un répartiteur de charge qui redirige arbitrairement la première requête de l'utilisateur sur le premier serveur car à ce moment là, il est moins sollicité. La session de l'utilisateur est alors ouverte et le fichier de session sauvegardé sur le serveur 1. La seconde requête est déclenchée mais elle est cette fois-ci redirigée sur le serveur 2. Comment ce dernier peut-il récupérer les données de session alors qu'elles se trouvent sur le premier serveur ? C'est embêtant... Une solution simple consiste à monter un système de fichier de partage et de configurer la constante de PHP `session.save_path` des deux serveurs afin qu'elle pointe dans ce dernier.

A partir des contraintes soulevées, il semble tout à fait judicieux de réaliser un système capable d'adapter le stockage des sessions sans nécessiter de grosse intervention sur le code source de l'application. Pour y parvenir, la meilleure solution consiste à étendre et redéfinir le comportement original des sessions afin de stocker les sessions dans une base de données MySQL.

Préparer la base de données

Avant de s'attaquer au code, il convient tout d'abord de créer une nouvelle base de données *php_solutions*.

Listing 11. *Contenu de la table php_session à l'initialisation de l'application*

```
mysql> select id, sess_id, sess_data, sess_time from php_session;
+----+----------------------------+-----------+------------+
| id | sess_id                    | sess_data | sess_time  |
+----+----------------------------+-----------+------------+
|  1 | ljn1sbiuhr481829qk4jfcfoh0 |           | 1282167188 |
+----+----------------------------+-----------+------------+
1 row in set (0.01 sec)
```

Listing 12. *Contenu de la table php_session après enregistrement d'une variable de session*

```
mysql> select id, sess_id, sess_data from php_session;
+----+----------------------------+----------------------------+
| id | sess_id                    | sess_data                  |
+----+----------------------------+----------------------------+
|  1 | ljn1sbiuhr481829qk4jfcfoh0 | favorite_color|s:4:"vert"; |
+----+----------------------------+----------------------------+
1 row in set (0.00 sec)
```

Cette tâche peut être accomplie dans un gestionnaire de base de données MySQL graphique comme *PHPMyAdmin* ou bien *MySQL Query Browser*. Vous pouvez également utiliser la ligne de commande MySQL comme le montre le Listing 7.

Maintenant que la base de données est créée, l'étape suivante consiste à créer la table *php_session* destinée à accueillir les données de session. Cette table est constituée de quatre champs : *id*, *sess_id*, *sess_data* et *sess_time*. Le Listing 8 donne en langage SQL la définition complète de la table. C'est ce code que vous devez exécuter dans votre gestionnaire de base de données afin de créer physiquement la table dans la base de données.

Le champ *id* constitue la clé primaire de la table. Sa valeur est auto incrémentée et gérée par le moteur MySQL. La colonne *sess_id* est une chaîne de longueur variable (`varchar`) stockant l'identifiant de session. La valeur est obligatoire et doit être unique dans toute la table. Le champ *sess_data* est une chaîne multiligne accueillant la chaîne sérialisée des données de session. Enfin, le champ *sess_time* est un entier non signé obligatoire stockant la valeur de la dernière date de mise à jour des données de session sous la forme d'un *timestamp* Unix. La base de données est prête à recevoir les nouvelles sessions grâce à la classe `MySQLSessionStorage` dévoilée dans la partie suivante.

La classe MySQLSessionStorage

L'objectif à présent consiste à développer un nouvel adaptateur capable de translater le mécanisme natif des sessions vers une base de données MySQL. Pour y parvenir, il convient de réaliser une nouvelle classe `MySQLSessionStorage` qui étend la classe `SessionStorage` et redéfinit les comportements natifs. Le listing 9 présente l'intégralité de cette classe.

La nouvelle classe `MySQLSessionStorage` accueille à présent un constructeur et six nouvelles méthodes: `sessionClose()`, `sessionOpen()`, `sessionGC()`, `sessionRead()`, `sessionWrite()` et `sessionDestroy()`. Le constructeur accepte une instance de PDO en argument et configure diverses options de la session. Les six méthodes restantes sont appelées automatiquement par le gestionnaire des sessions PHP avant et après l'exécution du script. C'est la fonction `session_set_save_handler()` de PHP qui est responsable de redéfinir tous les comportements internes du gestionnaire de session.

Le gestionnaire de sessions

L'API des sessions de PHP offre une manière simple et intelligente de redéfinir entièrement les mécanismes internes de gestion des sessions. La fonction `session_set_save_handler()` (*http://fr.php.net/session-set-save-handler*) remplit parfaitement ce besoin. La documentation officielle indique que cette fonction accepte six paramètres obligatoires. Chacun de ces arguments est une fonction utilisateur de rappel ou bien un tableau contenant une instance et le nom de la méthode publique correspondante à appeler. On parle alors de *callable*.

Le premier paramètre définit la fonction d'ouverture de la session. Cette fonction reçoit toujours deux paramètres : le chemin vers le dossier de stockage des sessions ainsi que le nom de la session. Ici, c'est la méthode `sessionOpen()` qui est configurée. Elle renvoie toujours *true*.

Le second paramètre définit la fonction de fermeture de la session. Cette fonction agit comme un destructeur de classe, et est exécutée lorsque le script se termine. Dans la classe `MySQLSessionStorage`, c'est la méthode `sessionClose()` qui sera appelée. Elle renvoie toujours *true*.

Le troisième paramètre de la fonction `session_set_save_handler()` configure la fonction de lecture de la session. La fonction configurée accepte l'identifiant de session comme paramètre et doit être obligatoirement une chaîne contenant les données de session. Il s'agit d'une chaîne sérialisée servant à initialiser le tableau de session `$_SESSION` au démarrage de la session. Si

aucune donnée de session n'existe, alors la chaîne retournée doit être vide.

La méthode `sessionRead()` tente tout d'abord de récupérer la session en cours depuis la table *php_session* grâce à une requête SQL *SELECT* intégrant l'identifiant de session passé en paramètre. S'il existe un enregistrement, alors la méthode retourne le contenu de la colonne *sess_data* qui correspond à la chaîne sérialisée. Si aucun *tuple* (*ligne*) n'est rapatrié, alors la méthode initialise une session en base de données en créant un nouvel enregistrement dans la table. Remarquez ici l'utilisation de PDO et des requêtes préparées afin d'éviter tout risque d'injection SQL.

Le quatrième paramètre permet de définir la fonction de rappel appelée au moment de la sauvegarde des données de session. Cette fonction est appelée par PHP après l'exécution du script et reçoit deux paramètres : l'identifiant de session ainsi que la chaîne sérialisée du tableau de session. Ici, c'est la méthode `sessionWrite()` qui se charge de remplir ce besoin. Cette dernière met à jour l'enregistrement correspondant à la session dans la table *php_session* à l'aide d'une requête SQL *UPDATE*.

Le cinquième paramètre de la fonction `session_set_save_handler()` définit ensuite la fonction de destruction de la session. Cette fonction accepte l'identifiant de session en paramètre. Le comportement de destruction est automatiquement exécuté lorsque la fonction `session_destroy()` est invoquée. Dans notre exemple, la méthode `sessionDestroy()` sera appelée lorsque la méthode `destroy()` de l'objet de session est appelée. La méthode `sessionDestroy()` détruit la session en supprimant l'enregistrement correspondant dans la table *php_session*.

Enfin, le dernier paramètre définit la fonction de nettoyage des sessions expirées. Cette dernière accepte un unique paramètre qui est la durée de vie maximale d'une session, exprimée en secondes. La méthode `sessionGC()` de la classe `MySQLSessionStorage` se charge de supprimer de la table *php_session* tous les enregistrements dont la valeur de la colonne *sess_time* correspond à une date expirée.

La prochaine et dernière partie de cet article s'articule autour des tests de cette nouvelle classe avec notre précédente application.

Test de la classe MySQLSessionStorage

La dernière étape de ce cours avancé sur les sessions consiste à tester la nouvelle classe `MySQLSessionStorage` avec notre précédente application. Pour y parvenir, il convient de modifier les premières lignes du fichier *index.php* comme le montre le Listing 10.

Le bloc *try { } catch { }* ouvre une connexion sur la base de données MySQL *php_solutions* en instanciant un objet PDO. Le nom d'utilisateur MySQL est ici *root* et le mot de passe est vide. En cas d'erreur, une exception de type *PDOException* est interceptée et interrompt le cours normal du script. Si la connexion sur la base de données est validée, alors la classe `MySQLSessionStorage` est instanciée. Le reste du code de l'application reste exactement le même que précédemment.

Les Listings 11 et 12 prouvent respectivement le fonctionnement de l'application. A l'ouverture de la page *www.phpsolutions.local*, une nouvelle session est enregistrée en base de données avec l'identifiant *ljn1sbiuhr481829qk4jfcfoh0*. Comme aucune couleur n'a été sélectionnée, il n'existe pas encore de donnée de session dans la colonne *sess_data*. L'envoi du formulaire avec *vert* comme valeur de couleur favorite entraîne la mise à jour de la session dans la base de données. La colonne *sess_data* se voit attribuer la valeur présentée dans le Listing 12.

Conclusion

Cet article a été l'occasion de découvrir des usages avancés du mécanisme des sessions de PHP en s'appuyant sur une approche orientée objet. Cette dernière apporte un certain nombre de bienfaits au code source de l'application. En effet, le code de l'application est à présent plus facile à utiliser, à maintenir et à faire évoluer. De plus, le code écrit ici est à présent entièrement testable à l'aide d'un framework de tests unitaires comme PHPUnit mais il est aussi et surtout réutilisable. Enfin, le code s'adapte tout aussi aisément à un gestionnaire de session différent. Par conséquent, il sera désormais possible de basculer au choix sur des sessions natives ou bien vers une base de données MySQL.

En se basant sur les exemples présentés et grâce aux concepts de la programmation orientée objet (héritage, implémentation d'interfaces, classes abstraites...), il convient de créer de nouvelles classes capables d'adapter de nouveaux gestionnaires de session. En effet, la création des adaptateurs pour d'autres bases de données relationnelles (SQLite, PostGreSQL, Oracle...) devient trivial. Bien entendu, il ne s'agit pas que de se limiter à ces systèmes. Un stockage en mémoire vive avec Memcache ou bien dans des bases NoSQL (*CouchDB, MongoDB, Cassandra*...) sont des alternatives complètement envisageables.

HUGO HAMON

Hugo Hamon est responsable des formations chez Sensio Labs et passionné de développement web. Il est à l'origine du site Apprendre-PHP.com et coauteur d'un ouvrage sur le framework Symfony aux éditions Eyrolles. Sur son temps libre, Hugo Hamon participe à l'AFUP en tant que Secrétaire Général et publie des billets sur son blog http://www.hugohamon.com.

Les applications WEB 2.0

Web 2.0 est un terme à la mode parmi l'ensemble des sites internet. Dans cet article, nous étudierons le sens de ce terme ainsi que quelques applications Web qui véhiculent cette idée.

Cet article explique :

- Les principes des applications Web 2.0.
- Des exemples d'applications Web 2.0.

Ce qu'il faut savoir :

- Connaître une partie des sites les plus connus du web (Google, Wikipédia, Facebook...).
- Connaître les principaux langages du web : HTML, CSS, PHP, Ajax serait un plus.

Le web regorge de sites qu'on nomme parfois applications Web 2.0. Au travers de son histoire, le web a évolué pour aboutir à l'image qu'on connaît de lui aujourd'hui. Dans cet article nous verrons dans un premier temps ce qu'est le Web 2.0 puis nous illustrerons cette notion au travers de différentes applications plus ou moins célèbres.

Au commencement....

A ses débuts, le web consistait en une série de pages html (*HyperText Markup Language*) contenant du texte, des images ainsi que des liens permettant d'accéder à d'autres pages contenant des éléments du même type. Le web était purement statique, pas de formulaire permettant une inscription ou poster des commentaires. Il n'était donc pas possible d'interagir avec les sites tel qu'on le fait aujourd'hui en s'inscrivant, connectant, uploadant un fichier, postant un commentaire pour réagir à une news etc. Il n'y avait donc aucune interactivité possible.

Une première : CGI

Apparue dès les premiers pas du World Wide Web en 1993, CGI (pour *Common Gateway interface*) est une interface qui, au lieu d'envoyer un fichier (page, image...) permet d'exécuter un programme et retournera le résultat généré. CGI permet de passer des arguments au programme afin de modifier le résultat généré, pour un programme de recherche par exemple. Cependant, une certaine lourdeur est apparue et d'autres solutions ont vues le jour depuis. Ainsi des version *améliorées* de CGI sont sorties telles que *FastCgi*. CGI fut donc l'une des premières solutions (si ce n'est pas la première) à permettre un lien entre un client et un serveur.

Une étincelle : coté Serveur

Rasmus Lerdorf créa en 1994 une bibliothèque logicielle en Perl pour son site web afin de pouvoir mémoriser qui avait visité son CV. Plus le temps passait et plus il ajoutait de nouvelles fonctionnalités tant et si bien qu'il se décida à ré-implémenter ses fonctions en langage C capables de discuter avec une base de données. L'ensemble des fonctions créées prenaient alors pour nom PHP/FI (pour ***P**ersonal **H**ome **P**age Tools/**F**orm **I**nterpreter*). En 1998, le langage sortit dans une version 3 développée par 2 étudiants : Andi Gutmans et Zeev Suraski. Peu de temps après, une version 4 est apparue et le langage prit le nom de PHP : *Hypertext Preprocessor*. Avec cette nouvelle version, un modèle de développement à été introduit dans le développement de sites web : le modèle Objet. Même si la conception implémentée dans cette version 4 a fait débat sur *vrai objet* ou pas. La version 5 de PHP a permit de reprendre une grande partie des principales notions de la programmation orientée objet. Vous connaissez sûrement les évolutions suivantes ainsi que le succès rencontré par ce langage.

Une étincelle : coté client

Au cours de l'année 2001, *Microsoft* implémenta l'objet *XmlHttp* en tant qu'objet *ActivX*. *Mozilla* ajouta l'objet sous le nom *XMLHttpRequest* peu de temps

après dans son célèbre navigateur Firefox. Pour simplifier, lorsqu'on navigue sur un même site, on recharge toute la page ce qui inclut le design de la page : fichier CSS, images, fichiers Javascript, texte se retrouvant sur toutes les pages du site comme le menu... Cela n'est plus vrai aujourd'hui dans le sens ou les navigateurs ont implémenté un cache qui permet de stocker les fichiers CSS, les images et d'autres données. Cependant le contenu texte, contenu dans le code html, sera re-téléchargé. Cela impose donc une utilisation parfois superflue de la bande passante. La nouveauté que permet l'objet *XMLhttp/XMLHttpRequest* c'est simplement de recharger une partie de la page. Par exemple considérons 3 cases de type text dans un formulaire et un bouton *submit*. En tapant du texte dans les 2 premières puis en cliquant sur le bouton *envoyer*, une requête contenant les valeurs contenues dans les 2 premiers champs texte sera envoyée au serveur et, suite à un traitement, ce dernier enverra sa réponse. L'objet pourra enfin insérer cette réponse dans la page du navigateur client sans avoir à recharger l'ensemble de la page mais seulement la partie *formulaire*. Cet exemple, bien que réalisable en javascript pur (concaténation de 2 chaînes), démontre ici l'utilisation possible de la solution Ajax.

Grâce à cette solution, il est possible de concevoir des pages web qui ne se rechargent pas ou seulement en partie.

Un terme

C'est en 2003 que l'expression *Web 2.0* apparue. Elle désigne des sites reposant sur un ensemble de données plus ou moins fournies par les utilisateurs (administrateurs ou personnes inscrites), un effet de réseau (pas forcément social) invitant à la participation (*Soyez le premier à donner votre avis*), proposant parfois des systèmes de syndication (flux rss) ainsi que des web services.

D'un point de vue technologique, cela aboutit principalement à l'utilisation d'un site web dynamique utilisant bien entendu de l'Html et du Css mélangés assez souvent à du JavaScript voire de l'Ajax pour un résultat encore plus dynamique ainsi qu'un langage dynamique coté serveur tel que PHP. Pour résumer, on pourrait dire qu'un site Web 2.0 est un site reposant assez souvent sur les données fournies par les utilisateurs que l'on invite à participer activement de telle manière qu'il est placé au centre du système.

Quelques applications !

Aujourd'hui le réseau internet dispose de milliards de machines proposant parfois un site web. Certains d'entre eux n'offrent que la découverte d'une passion réalisé par un particulier. D'autres, réalisés par des entreprises, proposent du contenu parfois très différent ! On parle parfois d'applications. Nous allons en étudier quelques une par catégories.

Forums

Le style d'application le plus répandu sur le net est bien entendu les forums qui permettent à chaque inscrit de poster un sujet aussi appelé 'topic' sur lequel les autres utilisateurs pourront répondre pour répondre à la question postée par le premier utilisateur ou fournir des informations complémentaires. Parmi les services les plus utilisés en France nous retrouverons *forumactif.fr* et *xooit.com/fr/*. Toujours pour les fans du *je fais moi même*, ces deux services reposent sur les forums PHPBB : *forums.phpbb-fr.com*. Ce dernier site est la référence francophone du support phpBB et non son site officiel que l'on peut trouver à *www.phpbb.com*.

Blogs

Les blogs pourraient être eux aussi un des principaux styles d'applications les plus répandus sur le net. Aujourd'hui, ils en sont même devenus incontournables. Beaucoup d'internautes disposent d'un blog parfois personnel ou professionnel. Un blog, c'est un site ou le propriétaire peut y poster un 'billet' pour y exprimer une idée, parfois passagère mais aussi pour réagir sur l'actualité ou simplement poster des photos d'amis, de vacances etc. Les autres utilisateurs inscrits sur le blog peuvent réagir à leur tour en commentant le billet posté. Pour créer un blog, il existe des services très connus tels qu'*over-blog.com, skyblog, hautetfort.com* ou encore le service de Google blogger.com. Ces services proposent d'héberger eux même votre blog sur leur plate-forme et de le mettre à jour d'un point de vue technologique. Mais il est aussi possible pour les informaticiens en herbe de créer son propre blog, codé à partir de zéro ou pour plus de rapidité avec des CMS tels que *Dotclear, Worldpress* ou encore *Drupal.*

Partage multimédia

Outre le désir de vouloir partager une idée, il est aussi possible de partager une vidéo, un son ou plus simplement des images. Parmi les plus célèbre nous donnerons *FlickR http://www.flickr.com* pour le partage de photos, puis *Youtube (youtube.com)* et *Dailymotion (dailymotion.com)* pour le partage de vidéos.

Partage de connaissances

Parmi le top 10 des sites web les plus visités au monde, nous retrouverons bien évidemment Wikipedia, *http://fr.wikipedia.org* reposant lui aussi sur la technologie PHP. Le but de wikipédia est le partage d'informations, de connaissances afin de réaliser une grande encyclopédie en ligne qui peut être remplie par tout le monde. Il existe sur le web différents sites reposant sur ce qu'on appelle des Wikis qui permettent à tout utilisateur de contribuer à la rédaction d'informations sur un domaine précis comme le droit pour *Jurispedia http://fr.jurispedia.org/index.php/Accueil* ou pas.

Autres applications célèbres

Il existe bien d'autres applications très connues : *Google Maps* permet de retrouver un lieu géographique en donnant comme indication une adresse ou des mots clés. *Géoportail* permet lui aussi d'explorer la terre vue du ciel, du moins seulement le territoire français.

Il existe bien d'autres applications célèbres permettant la réalisation de document en ligne, de se créer un réseau social avec Facebook et compagnie, de créer des sites web sans connaissances informatique avec des CMS (*Content Management System*) tels que Joomla! D'ailleurs Facebook est un exemple des possibilités offertes par PHP couplé à ajax. Cependant, dû à une certaine lourdeur créée par le nombre d'utilisateurs, facebook a développé *HipHop*, qui permet de compiler les scripts PHP afin d'obtenir une version plus rapide qui fonctionne avec PHP 5.2.

Le principe mash up

Lors de la relecture de cet article, on m'a signalé que j'avais omis un sujet pourtant interessant : les mash up. Même si l'on ne peut pas tout traiter au sein de cette article, le système de mashs ups est lui aussi quasi incontournable. Pour ceux qui ne connaissent pas, prenons l'exemple simple de la video filmée dans la rue qu'on envoie sur *Youtube*. Une fois le video publiée, *Youtube* met à la disposition des visiteurs un code qui permet d'inserer la video sur son propre blog ou site web. La possibilité du mash up est de permettre d'avoir sur un site du contenu qui n'est pas hébergé sur le dit site. Il existe beaucoup de solutions de ce genre. De plus beaucoup de sites mettent à disposition des API permettant le développement de nouvelles solutions grâce à une autre solution. L'API qui doit être vraisemblablement la plus utilisée est celle de *Google Maps*. Un site peut alors proposer de naviger sur les cartes de google maps en y ajoutant des données supplémentaires.

Pour finir

Lister l'ensemble des applications serait un travail ardu mais avec l'ensemble des sites web que nous venons d'aborder vous devez maintenant voir l'interaction que doit permettre un site se disant *Web 2.0*. Le but principal est de fournir un service, de l'information et assez souvent, la possibilité de la compléter, ou demander plus de précision. On ne se limite plus à un dictionnaire qui vous donne juste une définition mais aussi la possibilité de comprendre plus et d'en savoir encore plus.

NICOLAS TURMEAU

Nicolas Turmeau est un jeune étudiant en 3ème année de licence. Il utilise le langage PHP tous les jours depuis bientôt deux ans. Il est le développeur du jeu en ligne Numénor Online. Contact : nturmeau@iia-laval.fr.

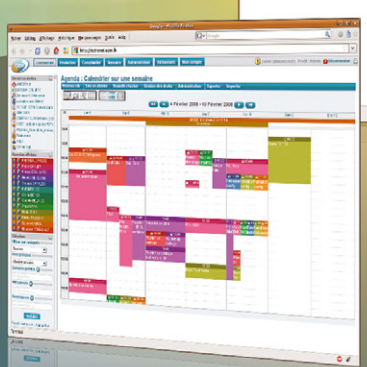
LINAGORA

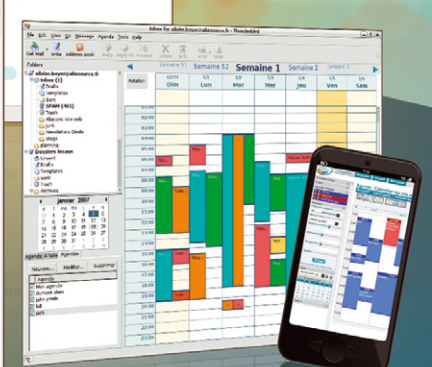
présente

OBM Online

LA MESSAGERIE COLLABORATIVE OPEN SOURCE HÉBERGÉE !

NEW





http://online.obm.org

WEB Gazelle

Génialement rapide

Votre site est-il vraiment rapide ?

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

En savoir plus : www.webgazelle.net

N°Azur 0 810 810 815
PRIX APPEL LOCAL

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

Lot d'une valeur de 2498.20 € TTC

Vous gagnez un site Internet qui comprend :

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

Extraits de règlement

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

Présentation des lots

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

Webgazelle.net, une marque de COGNIX systems