Sommaire

Au cœur de xoops

Xoops constitue un portail PHP évolutif et facile d'accès pour intégrer une application d'entreprise de type Intranet/Extranet. Cette possibilité d'extension se retrouve dans le « x » de xoops : eXtensible Object Oriented System. Prononcez-le à la mode anglo-saxonne : [zoo'ps] (zoups). De nombreux modules comme des forums ou des gestionnaires d'incidents, véritables applications dans l'application, couvrent déjà les fonctions les plus répandues du monde PHP. Il existe également plusieurs modules de gestion de contenus qui font entrer xoops dans la catégorie des CMS (Content Management System, soit Systèmes de Gestion de Contenu).

Par OryxVet

Côté utilisateur, xoops est simple et efficace. Il offre des services rassemblés dans une interface dédiée à la gestion du portail. Parmi eux, on trouve : un installeur de modules avec ses menus (block), une gestion d'utilisateurs et de groupes couplée avec un système de droits et de thèmes, un système d'alertes, un débogueur, un système de gestion de cache totalement intégré.

Côté programmeur, il est confortable de s'appuyer sur l'architecture solide de xoops : elle est constamment entretenue. Le modèle objet de xoops permet de s'appuyer sur une couche d'accès aux données persistantes. Diverses couches utilitaires facilitent le codage et rendent le cœur du programme particulièrement lisible. Livré en standard, on trouve les gabarits Smarty, un mini-cadre de développement (framework) pour construire un formulaire, un système de gestion des permissions, un utilitaire automatique pour envoi de messages électroniques, un gestionnaire d'images, un gestionnaire de téléchargement : cela constitue une trousse à outils à laquelle s'attend tout développeur PHP.

Une communauté très active et courtoise permet de rapidement avoir des réponses sur les forums à tous les niveaux d'utilisation. La documentation en français est complète, elle s'enrichit de jour en jour. La version anglaise est encore plus fournie.

Introduction

Nous cherchions un portail, une architecture, un cadre de développement qui puisse nous fournir un ensemble de services facilitant notre futur travail de développement. Nous recherchions au minimum une gestion d'utilisateurs permettant de se connecter de manière sécurisée et une gestion multilingue. Notre objectif était de développer et de mettre en place en 2 à 3 mois / homme une application destinée à un groupe de professionnels avec les 3 niveaux usuels : un extranet de type informatif, un intranet fournissant des services aux utilisateurs enregistrés et un accès administrateur. PHP/MySQL était une solution incontournable pour minimiser les frais fixes tout en permettant un hébergement de qualité à un coût abordable.

Parmi les architectures PHP proposées en Open Source, xoops nous est apparu le mieux adapté à nos besoins. L'abondance d'offre d'architecture PHP

est au premier abord un peu déroutante et après de nombreux tests difficiles, la facilité d'installation de xoops fit pencher la balance en sa faveur. xoops est opérationnel immédiatement après l'installation, sans toucher à une ligne de code. Cette facilité d'installation met en avant la simplicité et l'efficacité de xoops que l'on retrouve dans tous ses composants. Ceci fait défaut à beaucoup d'autres plates-formes PHP, où l'investissement initial en temps est plus important.

La communauté xoops est très réactive aux questions. Une moyenne de 150 personnes sont en permanence connectées sur le site xoops.org. Il existe dans le monde plus d'une vingtaine de sites de support, y compris en Chine et au Japon. Les langages multi-octets comme le chinois traditionnel ou le coréen sont pleinement supportés. La communauté française fait un travail remarquable sur www.frxoops.org . Le site reçoit 300 000 visites par mois et 13000 utilisateurs se sont enregistrés. Les développeurs du noyau sont très actifs et leur travail permet de bénéficier régulièrement de correctifs et d'évolutions. Une évolution majeure est livrée tous les 3 à 6 mois. Au moment de l'écriture de cet article, l'avant-dernière version intègre déjà la compatibilité avec PHP5 ; la dernière version apportait un système de sécurisation des formulaires contre les attaques CSRF[1]. Cette communauté de développeurs est structurée en mode projet et elle bénéficie d'une organisation de développements similaires aux plus grands éditeurs propriétaires.

Nous étions rassurés, xoops n'est pas un produit éphémère. La pérennité de xoops à moyen terme est assurée.

xoops : portail et système de gestion de contenu

Un des avantages de xoops est qu'il possède des extensions non-intrusives, qui ne remettent pas en cause son noyau. On les appelle les modules. Puisqu'il offre la possibilité d'intégrer facilement de nombreux modules, xoops se positionne comme un portail.

Les modules sont développés soit par l'équipe xoops, soit par les membres de la communauté. Chaque module ajoute des fonctionnalités au portail. Ils sont installés par l'administrateur. Plus de 200 modules téléchargeables sont disponibles en Open Source dont une trentaine de qualité professionnelle. Sans ordre, on peut citer :

- le forum newBB,
- le gestionnaire d'incidents xHelp
- le système d'envoi de courriel Xmail
- le système de gestion de téléchargements Downloads
- le gestionnaire de salle de réunion Resmanager
- plusieurs systèmes de FAQ faq
- les sondages xoopspoll
- les liens utiles mylinks etc...

La version 2.2, en cours de développement, livrera des paquets de différents modules afin de fournir un environnement dédié à chaque type d'utilisation. Il y aura un paquet de modules « Groupe de travail », un paquet « site communautaire », un pack « intranet » etc. [2]

Il existe plusieurs modules de type gestionnaire de contenu : le site http://www.xoops.fr en propose un comparatif. Ils placent xoops également dans la catégorie des CMS (Content Management System) au même titre que Typo3 ou Mambo. xoops fait régulièrement l'objet de comparatifs dans la presse du secteur. A noter que le module news livré en standard permet de gérer automatiquement un fil d'information au format RSS.

xoops est en perpétuelle évolution et développer un module au sein de cette architecture permet de bénéficier des avancés fonctionnelles et technologiques. La capacité à fournir des services simples et efficaces a fait le succès de xoops et l'attachement de sa communauté de développeurs.

Un autre avantage de xoops est l'utilisation du moteur de gabarits Smarty devenu incontournable en PHP. Il offre le découplage complet de la vue par rapport aux autres composants offrant ainsi une architecture souple et évolutive par l'utilisation de gabarits compatibles (X)HTML. La réalisation d'un thème, ensemble de gabarits, peut ainsi facilement être confié de manière indépendante à un graphiste puis intégré facilement pour être conforme à la chartre graphique d'une entreprise. Plus de 250 thèmes différents et personnalisables sont téléchargeables.

Notre besoin était de construire une application permettant de partager des services pour une communauté professionnelle. En nous appuyant sur l'architecture xoops, nous avons pu rajouter facilement un ensemble de modules afin d'établir une communication synchrone et dynamique entre nos membres.

Au cœur de xoops

Structure d'un module

Nous décrivons ici les différents fichiers et répertoires utiles pour la structure de base d'un module intégrable dans xoops.

xoops_version.php : c'est le fichier central du module. Il contient les informations nécessaires à l'installation du module : description du module, tables SQL, menus utilisés, blocks, gabarits, options de notification, etc. Il fait l'objet d'une documentation

claire [3]. Il est obligatoire.

header.php : il permet d'inclure les fichiers spécifiques du noyau du module. Il est facultatif.

Les répertoires :
/admin : c'est un répertoire contenant tous les fichiers relatifs à l'administration du module. Son accès est protégé. Seuls les utilisateurs de type administrateur peuvent y accéder. Au sein de ce répertoire, une structure particulière doit également être respectée.

/language : c'est est le répertoire contenant la valeur des constantes utilisées pour le support des langues. Il doit exister au moins un répertoire appelé 'english'. Le répertoire dédié au français s'appelle 'french'. Chaque type d'utilisation possède son propre fichier : main.php décrit les variables utilisées en général dans le module ; modinfo.php, celles utilisées lors de l'installation du module ; blocks.php celles utilisées dans les blocs.

/block : il contient les fichiers utilisés par les blocs. Nous reviendrons sur cette notion de bloc par la suite.

/sql : il contient le fichier de création des tables SQL. Un champ paramétrable à l'installation permet de suffixer le nom des tables.

/images : il contient les images utilisées par le module.

/cache : il contient les fichiers créés dynamiquement par xoops pour son cache.

/templates : il contient les gabarits du module.

Rien n'empêche le développeur d'ajouter des répertoires, pour structurer encore plus le module. Par exemple un répertoire appelé action, qui stockerait les actions métiers. Vous êtes libres de vos choix.

Structure de programme d'un module

Chaque programme doit être entouré d'un entête (header) et d'un pied de page (footer). Autrement dit, chaque programme doit obligatoirement commencer par le code suivant :

```
include '../../mainfile.php';
include XOOPS_ROOT_PATH."/header.php";
.....
```

et finir par le code suivant :

```
include XOOPS_ROOT_PATH."/footer.php";
break;
```

Le fichier mainfile.php situé à la racine comporte le chargement de variables globales telles que celles utilisées pour la connexion à la base de donnée (BD) ou les informations de l'utilisateur connecté.

XOOPS_ROOT_PATH."/header.php" charge les fichiers nécessaires au bon affichage de xoops. Il prépare les différents composants du portail en s'appuyant sur les paramètres personnalisés par l'administrateur.

XOOPS_ROOT_PATH."/footer.php"; finalise le travail du header en affichant tous les composants de l'IHM.

Le modèle objet de xoops

Le modèle objet de xoops est basé sur 2 classes XoopsObject et XoopsObjectHandler toutes les deux codées dans le fichier /kernel/object.php. Ces 2 classes forment la couche d'accès aux données persistantes tel qu'elle est décrite dans le pattern Data Acces Object (DAO). C'est en les spécialisant que l'on implémente le modèle pour une entité particulière, souvent associée à une seule table de base de données. Rien n'oblige le programmeur d'utiliser ces classes pour accéder au BD mais les raisons de les utiliser sont multiples :
- séparer la logique données de la logique métier,
- uniformiser la manière de programmer,
- augmenter la lisibilité du code et ainsi faciliter la maintenance,
- bénéficier d'un grand nombre de services de xoops, basés sur ce modèle.

On a souvent décrié PHP pour son manque de rigueur. En utilisant systématiquement cette couche d'accès aux données, on répond en partie à ce grief.

Nous décrivons ici sommairement les services rendus par ces 2 classes :

XoopsObject est la classe mère, qui permet de manipuler les attributs d'un objet donné. Pour cela, elle s'appuie sur le tableau $vars. $vars contient les attributs de l'objet sous la forme (clé, valeur). La clé désigne le nom de l'attribut, c'est-à-dire souvent le nom d'une colonne de la table de BD. XoopsObject offre les fonctions suivantes utilisables par héritage :

- initVar ($key, $data_type) permettant d'initialiser la définition d'un attribut.
- setVar($key, $value) qui met à jour l'attribut.
- getVar($kkey) qui restitue la valeur de l'attribut.
- cleanVars () nettoie les attributs des caractères spéciaux pour les stocker dans la BD.

Pour bénéficier de ces services, il suffit d'implémen-

ter le constructeur comme suit :

```
function XoopsPrivmessage() {
$this->XoopsObject();
$this->initVar('msg_id', XOBJ_DTYPE_INT, null,
false);
$this->initVar('msg_image', XOBJ_DTYPE_OTHER,
'icon1.gif', false, 100);
$this->initVar('subject', XOBJ_DTYPE_TXTBOX,
null, true, 255);
$this->initVar('from_userid', XOBJ_DTYPE_INT,
null, true);
$this->initVar('to_userid', XOBJ_DTYPE_INT,
null, true);
$this->initVar('msg_time', XOBJ_DTYPE_OTHER,
null, false);
$this->initVar('msg_text', XOBJ_DTYPE_TXTAREA,
null, true);
$this->initVar('read_msg', XOBJ_DTYPE_INT, 0,
false);
}
```

Listing 1

XoopsObjectHandler est une classe abstraite qu'il faut totalement implémenter. Elle permet de manipuler les objets d'une classe particulière (maclasse) étendant la classe XoopsObject. Ce gestionnaire (ou manipulateur) peut se récupérer à l'aide de la méthode xoops_gethandler(maClass) codée dans le fichier /include/function.php. Attention toutefois car cette méthode se base sur une convention de noms, il faut avoir nommé son gestionnaire xoopsmaclasseHandler.

Voici les principales méthodes proposées par ce gestionnaire qu'il s'agit d'implémenter :
- create : création d'un nouvel objet (référence au pattern fabrique abstraite).
- insert : insertion d'un objet dans la BD (requête insert ou update).
- delete : suppression d'un objet.
- get($id): accès direct à un objet à l'aide de son identifiant.
- GetObject : accès à une collection d'objets sélectionnée selon un critère.

La classe XoopsPrivmessage issue du noyau de xoops utilise ces deux classes. Voici quelques exemples d'utilisation :

Affichage du sujet et de la date de création des 'privmessage' destinés à l'utilisateur courant.

```
$criteria = new Criteria('to_userid',
                    $xoopsUser->getVar('uid'));
$pms =& $pm_handler->getObjects($criteria);
foreach ($pms as $pm ) {
echo $pm->getVar('subject').'-'.
                      $pm->getVar('msg_time');
}
```

La classe Criteria permet de définir le critère de sélection. Dans notre exemple le champs 'to_userid' doit être égal à l'identifiant de l'utilisateur courant. Mise à jour d'un objet dont l'identifiant est stocké dans la variable $idPm :

```
$pm_handler =& xoops_gethandler('privmessage');
// récupération du handler
$pm =& $pm_handler->get($idPm); // Récupération
de l'objet
$pm->setVars(($_post) ; // Mise à jour de l'ob-
jet à partir de la requête HTTP
$pm_handler ->insert($pm) ; // mise à jour phy-
sique dans la BD avec le handler
```

Voici un exemple d'utilisation de ces 2 classes au sein d'un programme 'index_client.php' associées à la table client. Nous ne décrivons pas le formulaire stocké dans un fichier du répertoire /form.

```
<?php
include_once XOOPS_ROOT_PATH."/header.php";
include   XOOPS_ROOT_PATH.'/modules/vxp/class/
client.php';
include  XOOPS_ROOT_PATH.'/modules/vxp/action/
store_client.php';

// request management
if (isset($_POST)) { extract($_POST); }
if (isset($_GET))  { extract($_GET); }
if (!(isset($op))) $op='';
if (!(isset($id))) $id='';

// creation du DAO vers la table client
if ($id) {
       $handler_client     =     &     xoops_
gethandler('client');
       $client = $handler_client->get($id);
       $client->unsetNew();
} else {
       $client = new client();
       $client->setNew();
    foreach ($client->getVars() as $k => $v){
              $$k='';
       }
}
// action management
if ($op) {
       if (isset($_POST)) { // chargement de
l'objet par la requete
              $client->setVars($_POST);
       }
       // execute directly the "op" fonction
       $retour=$op($client);
       $msg=$retour['msg'];
       redirect_header('index_client.php?id='.
$retour['id'],1,$msg);
}
foreach ($client->getVars() as $key => $value)
{
       if (isset($value['value'])) {
              $$key = $value['value'];
       }
}
include 'form/form_client.php';
include_once XOOPS_ROOT_PATH."/footer.php";
?>
```

Listing 2

L'action de sauvegarde « store » est stockée dans le répertoire /action :

```
function store (&$client)  {
$retour=array();
$client_Handler        =        &        xoops_
gethandler('client');
if (!$client_Handler->insert($client)) {
    $retour['msg'] = _MD_NO_DB_UPDATED;
} else {
    $retour['msg'] =  _MD_DB_UPDATED;
}
$retour['id']   =     $client->getVar('client_
id');
return $retour;;
}
```

Listing 3

La trousse à outils du développeur

Les développeurs de xoops ont développé un certain nombre de fonctions utilitaires qu'il est simple d'utiliser. Nous décrivons ici 3 fonctions permettant de développer une IHM conforme aux premiers critères ergonomiques de Scapin et Bastien [4] : conseiller, orienter, informer et conduire l'utilisateur lors de ses interactions avec le portail.

xoops_confirm : affiche pour l'utilisateur final une fenêtre de confirmation utilisée par exemple pour confirmer une suppression. Simple et efficace.

```
xoops_confirm(array('op' => 'confirm_delete', 'id'
=> $del_id), 'index_client.php', _ CONFIRMDEL);
```

La fonction prend en paramètre un tableau contenant les valeurs de la nouvelle requête, le programme vers lequel rediriger cette requête et le libellé de la question.

redirect_header : permet de rediriger la requête HTTP vers un autre programme PHP. Cette fonction est utilisée pour informer l'utilisateur de la bonne prise en compte d'une action, par exemple la bonne prise en compte de la création d'un élément.

XoopsPageNav permet d'effectuer facilement une pagination. Le premier paramètre indique le nombre d'occurrence, le second le nombre d'occurrence sur une page et le dernier le nom du champ permettant la communication entre les pages. Nous verrons par la suite un exemple d'utilisation avec gabarit.

Les principaux services de xoops

Pour présenter les services de xoops, il existe 3

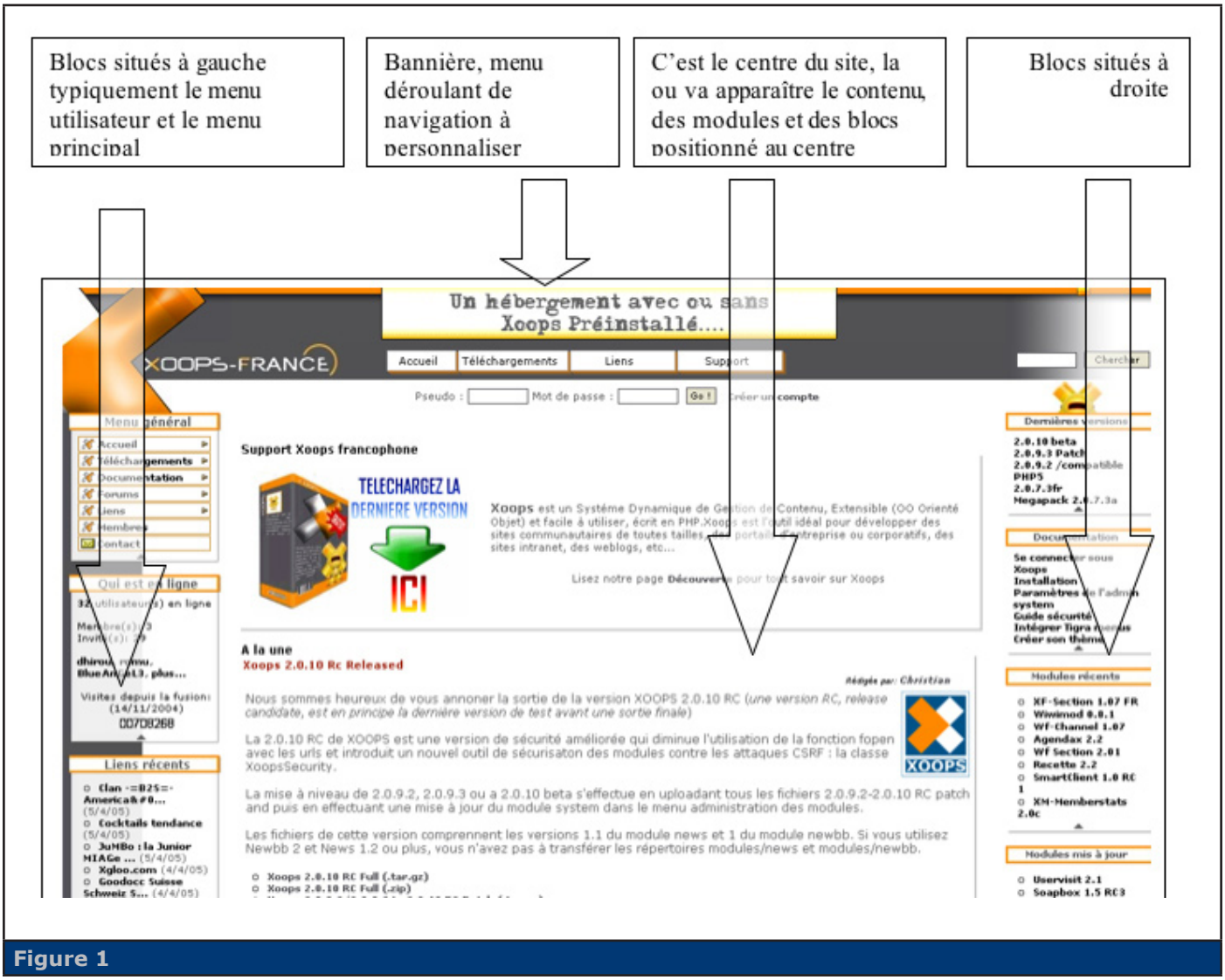

Figure 1

perspectives ou points de vue concernant :
- L'utilisateur final, c'est l'utilisateur enregistré ou le visiteur éphémère du portail. C'est lui qui fait vivre le portail.
- L'administrateur, aussi appelé webmestre, qui gère la configuration du portail.
- Le programmeur de module. Il est fournisseur à la fois de l'utilisateur final et de l'administrateur.

Les paragraphes suivants sont déclinés chaque fois pour chacune de ces 3 perspectives.

La notion de module et de block

utilisateur final

Pour l'utilisateur final, l'accès à un module s'effectue par le biais de blocs situés de part et d'autre du centre ou affichés dans les écrans du module courant. La figure 1 décrit les différentes parties du portail sur un exemple.

administrateur

Dans la partie dédiée à l'administration, l'administrateur a la possibilité d'installer les modules placés dans le répertoire /module. Une fois le module installé, un ou plusieurs blocs, menus relatifs au module, peuvent être positionnés sur le portail à droite, à gauche ou au centre. Chaque module possède ses propres menus d'administration nécessaires à son paramétrage [figure 2]. L'administrateur les personnalise selon les besoins des utilisateurs finaux. Cette approche est typique des progiciels, avec un paramétrage fort.

Lors de la première installation, aucun module n'est intégré par défaut et seuls 2 blocs sont affichés : le menu utilisateur, pour l'accès aux informations utilisateur et pour l'administrateur l'accès à l'administration du portail, et le menu général, qui sera enrichi par la suite des menus de chaque modules.

Programmeur de module

Le programmeur doit coder les programmes constituant le cœur de son module, les blocs permettant à l'utilisateur d'y accéder et les programmes relatifs à l'administration. Les informations nécessaires à l'intégration d'un module sont stockées dans le fichier xoops_version.php. On y code un nombre important de comportements communs à tous les modules. Paramétrages, menus, gabarits, alertes, commentaires sont personnalisés à l'aide de tableaux régis par des conventions de noms.

Nous décrivons ici un exemple de code du fichier xoops_version.php permettant d'ajouter un paramétrage utilisateur. Le principe est le même pour les autres fonctionnalités personnalisées. Un paramétre utilisateur est ajouté en définissant le tableau $modversion['config']. L'exemple suivant est tiré de la configuration utilisée par le module officiel « news », fourni avec le portail. Il offre la possibilité pour l'administrateur de personnaliser le nombre d'articles présent sur une page.

```
$modversion['config'][1]['name'] = 'storyhome';

$modversion['config'][1]['name'] = 'storyhome';
$modversion['config'][1]['title'] = '_MI_STORYHOME';
$modversion['config'][1]['description'] = '_MI_STO-
RYHOMEDSC';
$modversion['config'][1]['formtype'] = 'select';
$modversion['config'][1]['valuetype'] = 'int';
$modversion['config'][1]['default'] = 5;
 $modversion['config'][1]['options'] = array('5' =>
5, '10' => 10, '15' => 15, '20' => 20);
```

Listing 4

Le paramètre « 'storyhome'» est personnalisable par l'administrateur à l'aide du menu « préférence ». Un menu préférence est présent pour chaque module installé.

Dans le code, la valeur du paramètre 'storyhome' est accessible via la variable $xoopsModuleConfig'storyhome'.

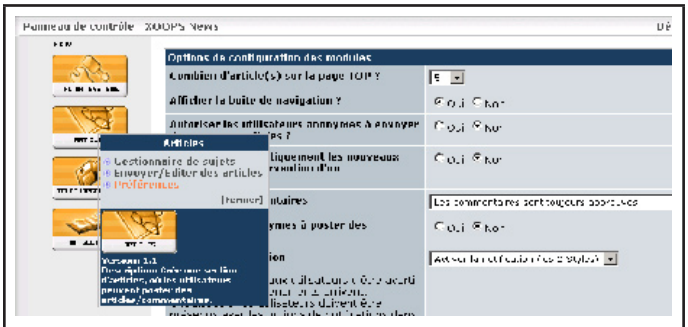

Figure 2

Lors de l'installation d'un module, les informations de xoops_version.php sont stockées en base de données et les requêtes de création de tables utiles au module sont exécutées.

Les utilisateurs, les groupes et leurs permissions

Xoops gère au minimum 3 groupes d'utilisateurs, créés par défaut à l'installation.
- Utilisateurs anonymes
- Utilisateurs enregistrés. Dès que quelqu'un s'enregistre, il est placé par défaut dans ce groupe
- Administrateurs ou Webmestres. Le groupe administrateur est un groupe particulier accédant à toute la partie d'administration du portail.

Utilisateur final

L'utilisateur final a accès à son identification. Il peut modifier à tout moment la plupart de ses informations publiques. Cette identification est très simple : elle comprend le nom d'utilisateur, son mot de

passe, l'adresse courriel et quelques informations permettant une communication synchrone telle que l'adresse IRC utilisée par les modules implémentant un chat.

L'utilisateur essayant de charger une page non permise tombe sur un écran signalant l'accès interdit.

Administrateur
Dans la partie administration, il gère les permissions des groupes. Chaque module ou bloc installé dans le portail possède des permissions par groupe. L'intérêt de ces groupes est évidente pour un intranet d'entreprise. Le groupe « Utilisateur enregistré » peut être divisé en plusieurs groupes donnant accès à tel ou tel service de l'entreprise.

La seule chose que l'on peut regretter ici est de ne pas avoir intégré le support LDAP directement dans le noyau. Il faut aujourd'hui utiliser une extension (hack). Cette lagune sera bientôt corrigée avec l'intégration de LDAP en standard [1].

Certains modules ont développé une couche de permission, l'administrateur aura alors à charge de gérer ces permissions. Par exemple, le module WFSection CMS avec workflow de validation, possède sa propre gestion d'accès pour chaque catégorie ou chaque article en fonction des groupes.

En parallèle avec les permissions, l'affichage de chaque bloc est personnalisable. L'administrateur décide sur quelle page le bloc doit apparaître. Ainsi, on peut permettre l'affichage sur tout le portail, uniquement la page d'accueil du portail ou bien sur un module particulier. Cette possibilité est puissante puisqu'elle permet de gérer l'affichage de tel ou tel bloc selon la situation, la position de l'utilisateur dans le portail. On parle d'IHM situationnelle.

Les administrateurs ont la possibilité de communiquer avec leurs membres à l'aide d'un système de mail permettant l'envoi par groupe, par sélection de critères ou individuellement. Il existe également une messagerie interne.

Programmeur de module
Le programmeur peut spécialiser le système de permission de son module comme le fait le module WFSection. Il doit alors développer dans la partie administration le programme permettant de gérer ses permissions et l'implémenter dans le corps du module. Il existe une classe utilitaire facilitant le travail du programmeur.

xoops, son aspect graphique et Smarty

Le moteur de gabarit Smarty est utilisé afin d'afficher l'ensemble des composants du portail indiqués sur la figure 1. Il détermine la disposition globale du portail. On parle alors de thème. Un thème définit l'aspect graphique du portail. La couleur, la disposition des logos et des bannières, la largeur de la colonne de droite et de gauche ou la présence d'entête et de pied de page sont des éléments personnalisables du thème. Un thème comporte un ensemble de fichiers Smarty mais aussi la feuille de style CSS. Les fichiers d'un thème particulier sont contenus dans un seul répertoire. Il devient alors directement utilisable. L'administrateur définit le thème utilisé par défaut. Une instance de xoops peut comporter plusieurs thèmes.

Smarty est également utilisable pour l'affichage individuel des éléments. Cela concerne aussi bien les blocs (menu principal, menu utilisateur,etc.) que le contenu d'un module.

Utilisateur final
Un bloc fourni en standard permet à l'utilisateur de personnaliser l'aspect graphique du portail en sélectionnant un thème particulier parmi ceux installés.

Administrateur
Chaque module possède sa propre collection de gabarits définis dans le fichier xoops_version. A l'installation ou à la réinstallation d'un module ses gabarits sont alors compilés et stockés en base de donnée. Chaque module possède sa propre collection de gabarits éditables et modifiables au sein d'un éditeur.

Programmeur de module
La réalisation d'une chartre graphique est souvent sous-traitée à un graphiste et le découplage complet du graphisme et des données le lui permet avec xoops de manière complètement autonome. xoops a spécialisé la classe maître de Smarty et utilise ses propres délimiteurs pour les variables Smarty <{ et }>.

Les gabarits utilisés dans un module sont définis dans le fichier xoops_version.php à l'aide de la variable $modversion['templates'].

Le simple fait d'affecter la variable $xoopsOption['template_main'] dans le corps du programme indique quel gabarit utiliser. Les variables Smarty utilisées dans les gabarits doivent être affectées à l'aide de la classe $xoopsTpl.

Voici un exemple couplant le modèle objet et l'utilisation de Smarty qui affiche l'ensemble des clients présents dans la BD. Voyez le listing 5 pour un exemple de programme utilisant un gabarit : liste des clients couplée avec le gabarit.

```
<?php
include ‘header.php’;
include XOOPS_ROOT_PATH.’/header.php’;
include   XOOPS_ROOT_PATH.’/modules/vxp/class/
client.php’;

$xoopsOption[‘template_main’] =
                          ‘liste_client.html’;
$xoopsTpl->assign(array(
“lang_nom”=> _FO_NOM,
“lang_prenom”=> _FO_PRENOM ));

$handler_client =
             & xoops_gethandler(‘client’);

$nb_clients = $handler_client->getCount();
$criteria    =    new    criteria();$criteria-
>setLimit(10);$criteria->setStart($_
GET[‘start’]);

$clients =
      $handler_client->getObjects($criteria);
$xoopsTpl->assign(‘clients’,$clients);

include_once XOOPS_ROOT_PATH.
                          ’/class/pagenav.php’;
$pagenav = new XoopsPageNav(
                $nb_clients, 10, 1, ‘start’);
$xoopsTpl->assign(‘pagenav’,
                       $pagenav->renderNav());
include_once XOOPS_ROOT_PATH.”/footer.php”;
?>
```

Listing 5

```
<div style=”text-align: left; margin: 10px;”>
<{if$pagenav}>Page <{$pagenav}><{/if}></div>
<table class=”outer” cellspacing=”1” cellpad-
ding=”4”>
  <tr>
     <th align=”center”><{$lang_nom}></th><th
align=”center”><{$lang_prenom}></th>
  </tr>
  <{section name=i loop=$clients}>
  <tr >
            <td   class=”even”><{$clients[i]-
>getVar(‘nom’)}></td>
           <td   class=”even>”><{$clients[i]-
>getVar(‘prenom’)}></td>
  </tr>
  <{/section}>
</table>
```

Listing 6

Listing 6 : gabarit utilisé par le listing 5 affichant la liste des clients dans une table HTML.

Cet exemple permet d’afficher la liste des clients. Nous avons utilisé la classe $Xoopspagenav qui permet de gérer une pagination. Dans notre exemple on pagine pour 10 clients.

Le système d’alertes

Le système de notification permet une communication asynchrone entre un utilisateur créant un événement et les personnes s’étant inscrites à cet événement. Cette fonction est très utilisée pour les forums ou les CMS où l’utilisateur peut s’inscrire pour être prévenu de l’ajout d’une réponse ou l’ajout d’un article dans une catégorie qui l’intéresse.

Utilisateur final

Sur chaque page d’un module implémentant le système d’alerte, il existe en bas de page une liste d’événements auxquels on peut souscrire. Par exemple « Alertez-moi quand un nouvel article est posté dans cette catégorie » ou « Alertez-moi de chaque nouvel envoi dans ce forum ». Pour souscrire à un événement, l’utilisateur coche l’événement et valide le formulaire. Il apparaît alors dans le menu utilisateur une ligne de menu « notification » qui permet d’ajouter l’événement parmi l’ensemble des événements auxquels l’utilisateur est inscrit.

D’une manière générale, l’utilisateur peut personnaliser son mode d’alerte en demandant une alerte par courriel ou une alerte par messagerie interne. Il peut aussi désactiver le mode d’alerte. Afin de ne pas être surchargé de messages, il peut aussi demander à être averti une seule fois, lors du premier événement qui survient.

Administrateur

Lorsqu’un module implémente les alertes, son menu « préférence » dans le panneau d’administration est enrichi de deux paramètres permettant de modifier le comportement des alertes au sein du module. Le premier permet de personnaliser le style d’alertes et le second de sélectionner les évènements auxquels les utilisateurs auront accès, parmi ceux définis par le programmeur.

Programmeur de module

Le fichier xoops_version.php contient les informations relatives aux alertes d’un module. Les évènements sont regroupés en catégorie, représentant un type d’objet utilisé par le module. Par exemple, la section et l’article sont des catégories pour le module news. Chaque catégorie est décrite dans le fichier ainsi que chaque événement [3]. Un évènement est associé à gabarit de message électronique, qu’il faut fournir.

Dans le code, on ajoute le support des alertes en ajoutant un gabarit au sein du gabarit maître du programme : $xoopsOption[‘template_main’].

```
include file=’db:system_notification_select.html’
```

Le programmeur peut également utiliser la classe Notification pour utiliser le système de notification explicitement :

```
$notification_handler =& xoops_gethandler('notification');
$notification_handler->triggerEvent ($category, $item_id, $event, $extra_tags=array(), $user_list=array(), $module_id=null, $omit_user_id=null);
```

Listing 7

Il y a de quoi faire un article complet avec toutes les fonctions de xoops. L'ensemble de classes gérant les formulaires, le mécanisme de recherche au sein des modules, le système de commentaires, le gestionnaire d'images sont d'autant de sujets que l'on pourrait détailler.

Bilan et perspective

Ce qui frappe chez xoops, c'est la facilité et l'efficacité de la plate-forme. xoops est un formidable outil pour intégrer des applications intranet. Face au populaire SPIP ou à Mambo, il est plus orienté portail et donc plus ouvert et personnalisable. Face à des architectures comme Typo3 ou ez Publish, xoops est beaucoup plus accessible pour une petite équipe informatique, qu'elle soit maîtrise d'œuvre ou maîtrise d'ouvrage. Sa prise en main par un administrateur est très rapide et ne nuit pas à ses possibilités d'extension. Le dynamisme de la communauté, sa fiabilité et le faible coût de possession en font un outil ayant un retour sur investissement élevé. Pour un développeur, le code du noyau est facilement lisible et de nombreux services présents en standard offrent une formidable efficacité pour développer de nouvelles applications.

Il existe de nombreux exemples d'utilisation de xoops dans le monde professionnel. Il est actuellement utilisé par le gouvernement du Québec [5], pour l'intranet de l'usine de pneu Goodyear à Valleyfield, pour les projets Open Source de Novell [6] ou simplement pour la dématérialisation des marchés publics, portail fournisseur de l'INRA ou de l'INRETS [7].

Une version majeure est en cours de développement. Elle est ambitieuse [8]. Elle apporte de nombreuses consolidations comme la prise en compte directement dans le noyau de LDAP et de la génération de PDF. De nombreuses évolutions vont être intégrées comme un système d'héritage de thème ou une gestion de version. Quand on voit l'enthousiasme de la communauté française et les statistiques croissantes de son utilisation, on se dit que xoops à de beaux jours devant lui....

A propos de OryxVet

OryxVet est également une compagnie experte sur xoops. Elle conseille et met en place des portails d'entreprise xoops et réalise des modules spécifiques.

Ressources

[1] : Voir l'article de direction|PHP en Novembre 2003 au sujet des attaques CSRF.
[2] : Interview de scalpa : http://www.frxoops.org/modules/news/article.php?storyid=680
[3] : Une description complète du code du fichier xoops_version.php:http://www.frxoops.org/modules/wfsection02/article.php?page=1&articleid=13
[4] : Critère d'ergonomie http://www.ergoweb.ca/criteres.html
[5] : Interview du directeur du CRISP : http://www.frxoops.org/modules/news/article.php?storyid=693
[7] : Site de Novell : http://forge.novell.com/modules/news/
[8] : Portail fournisseur de l'INRA http://www.inra.fr/fournisseurs/ et de l'INRETS : http://achats.inrets.fr/index.php ,
[9] : plan de développement de xoops : http://www.frxoops.org/modules/news/article.php?storyid=739

Pour discuter de cet article, et retrouver les réactions des lecteurs, rejoignez-nous sur le forum consacré [au numéro d'avril]

Edition

Abonnement en ligne :
C'est la solution la plus rapide. Rendez-vous sur le site de direction|PHP. Identifiez-vous, et suivez le lien "Magazine", puis la rubrique "s'abonner". Votre abonnement peut être effectif immédiatement, ou bien le mois prochain, ou encore en profitant des archives.

Abonnement par chèque ou
Abonnement avec bon de commande :
C'est la solution la plus formalisée. Rendez-vous sur le site de direction|PHP. Identifiez-vous, et suivez le lien "Magazine", puis la rubrique "s'abonner". Choisissez-un abonnement par chèque, et imprimez le document PDF qui vous sera remis. C'est un bon de commande complet. Retournez-nous votre règlement par chèque.

Achat au numéro
Direction|PHP est aussi disponible au numéro, aussi bien pour le numéro courant que pour les anciens numéros.

Anciens numéros
Tous les anciens numéros de Direction|PHP sont disponibles sur le site web [Y aller]. Il est possible de commander plusieurs archives de Direction|PHP en prenant un abonnement.

Ecrire dans Direction|PHP
Direction|PHP recherche en permanence les meilleurs articles sur la plate-forme PHP et MySQL
- Vous êtes un expert de PHP ou MySQL?
- Vous voulez partager votre cas d'utilisation?
- Vous avez une réalisation réussie ?
- Vous avez développé une technique originale?

Nous voulons vous entendre!
Contactez-nous pour nous détailler votre expérience ou votre idée d'article. Nous étudierons comment travailler ensemble .

D'autres informations, un guide pour écrire et des propositions de sujets qui nous intéressent sont disponibles sur le site : [Site en ligne]

Information sur le contenu
Direction|PHP est constitué de deux documents électroniques :
- Un document PDF personnalisé
- Une archive zip, contenant les codes sources, les tables de données, images ou tout autre document complémentaire aux articles.

Il arrive que la numérotation des listings, images ou autres encarts ne soit pas complet dans un article, car les listings font référence aux fichiers fournis dans l'archive. Ils ne sont pas forcément cités, mais sont livrés.

Direction|PHP Numéro 19
Avril 2005

Rédacteur en chef
Damien Séguy (damien.seguy@directionphp.biz)

Equipe éditoriale
Christophe Ballihaut
Frédéric Bordage
Cyril Pierre de Geyer
Guillaume Plessis
Ghislain Seguy

Auteurs
Marcus Baker, William Candillon, Ron Korving, Ed Lecky-Thompson, OryxVet, Damien Séguy, Chris Shiflett, José Pablo Ezequiel Fernández Silva, Lukas Smith.

Traduction
Maud Duminy.

Numéro ISSN : 1765-2634
Direction|PHP est un magazine mensuel,
publié par Nexen Services SA.
139 Bd Malesherbes
75017 Paris (France).

Direction|PHP contient des articles traduits du magazine original PHP|architect, publié par Marco Tabini Associates, Canada.

Contacts
Edition : edition@directionphp.biz
Abonnements: abonnements@directionphp.biz
Publicité : pub@directionphp.biz
Service à la clientèle : suppport@directionphp.biz