Un exemple d'utilisation des classes XoopsObject et XoopsObjectHandler

CatÃ©gorie : Fiches techniques

PubliÃ© par Christian le 17/04/2005

Un exemple d'utilisation des classes XoopsObject et XoopsObjectHandler par Oryxvet et article prÃ©sente une architecture minimale mais fonctionnelle de crÃ©ation / modification / liste des Ã©lÃ©ments d 'une table existante de BD. Il fait suite Ã¡ la publication d'un article prÃ©sentant les classes XoopsObject et XoopsObjectHandler. 1.1. Principe du MVC :L'architecture prÃ©sentÃ©e ici propose un architecture proche du fameux MVC (ModÃªle Vue Controlleur). Le MVC sert de trame Ã¡ la description d'un exemple d'utilisation du pattern DAO developpÃ© sous xoops. Le ModÃªle est reprÃ©sentÃ© par la class DAO (rÃ©pertoire Class) , la vue par le formulaire (rÃ©pertoire /form) et le contrÃ´leur par le programme index_Maclass.php mis Ã¡ la racine.
L'index gÃªre Ã¡ la fois l'affichage des donnÃ©es dans le formulaire et la rÃ©ception des requÃªtes http issues de ce mÃªme formulaire (Le formulaire affichÃ© par l'index est postÃ© vers l'index).1.2. Le ModÃªle :Prenons l'exemple d'une table Client que j'ai appelle vxp_client. Suffixer le nom des tables par le nom du module les utilisant est une bonne habitude car cela permet d'en faire un groupe homogÃ¨ne .
Pour avoir une class qui implante les classes XoopsObject et XoopsObjectHandler du pattern Data Access Objet on peux utiliser 3 maniÃªres :
 1/ On peux la coder Ã¡ la main 2/ On peux utiliser le concept de Mithrandir qui utilise une classe gÃ©nÃ©rique du handler http://dev.xoops.org/modules/xfsnippe ... ype=snippet&snippet_id=27 et ne code que le xoopsObjets 3/ On peux utiliser le module " class generator " afin de gÃ©nÃ©rer le fichier contenant ces 2 classes.
Nous utilisons ici class generator pour des raisons de simplicitÃ©. Un prÃ©alable Ã¡ l'utilisation du class generator est que le module utilisant cette table soit dÃ©jÃ¡ installÃ© sous Xoops (les modules proposÃ©s dans l'IHM sont ceux dÃ©jÃ¡ installÃ©s). Le module class generator est disponible sur dev.oryxvet.com et www.xoops.org . Il est en cours de validation sur www.frxoops.org. La classe gÃ©nÃ©rÃ©e par le class generator est la suivante : classÂ vxp_clientÂ extendsÂ XoopsObject {Â

```
Â Â Â Â varÂ $db;

Â Â Â Â functionÂ vxp_clientÂ ($id=null)
Â Â Â Â {
Â Â Â Â Â Â Â Â $this->dbÂ =&Â Database::getInstance();
Â Â Â Â Â Â Â Â $this->initVar("client_id",XOBJ_DTYPE_INT,null,false,10);
Â Â Â Â Â Â Â Â $this->initVar("nom",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar("prenom",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar("adresse_ligne1",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar("adresse_ligne2",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar("adresse_cp",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar("adresse_commune",XOBJ_DTYPE_TXTBOX,Â null,Â false);
Â Â Â Â Â Â Â Â ifÂ (Â !empty($id)Â )Â {
Â Â Â Â Â Â Â Â Â Â Â Â ifÂ (Â is_array($id)Â )Â {
Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â $this->assignVars($id);
Â Â Â Â Â Â Â Â Â Â Â Â }Â elseÂ {
Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â $this->load(intval($id));
```

Â Â Â Â Â Â Â Â Â Â Â Â }
Â Â Â Â Â Â Â Â }Â elseÂ {
Â Â Â Â Â Â Â Â Â Â Â Â $this->setNew();
Â Â Â Â Â Â Â Â }
Â Â Â Â Â Â Â Â
Â Â Â Â }

EtcÂ Ã¢â‚¬Â¦. Pour plus d'information sur la description de ces 2 classes voir les 2 url suivant du wiki xoops : http://dev.xoops.org/modules/phpwiki/index.php/XoopsObjectHandler http://dev.xoops.org/modules/phpwiki/ ... php/XoopsObjectAndHandler Notez que le nom des attributs est exactement la mÃªme que le nom des colonnes de la base de donnÃ©es.Nous avons donc notre ModÃªle du MVC via la class MonModule/class/vxp_client.php. Maintenant passons Ã¡ la prÃ©sentation de ce modÃªle dans une Vue. 1.3. La VueLa vue est tout simplement le formulaire. L'utilisation des classes xoopsform dÃ©diÃ©es Ã¡ la crÃ©ation de formulaire (rÃ©pertoire classxoopsform) est la maniÃªre la plus rapide de crÃ©er un formulaire. Nous allons coder ce formulaire dans un fichier form_vxp_client.php afin de bien distinguer la vue du contrÃ´leur. Nous utilisons la convention de nom selon laquelle le nom des variables stockant la valeur des attributs est exactement le mÃªme que le nom de la colonne, le mÃªme que celle des " Vars " du xoopsobject. Voici le fichier /form/form_vxp_client includeÂ XOOPS_ROOT_PATH.'/modules/vetexpert/language/' .$xoopsConfig['language'].'/form_vxp_client.php';
1Â Include_onceÂ XOOPS_ROOT_PATH.'/class/xoopsformloader.php';

$titre_formÂ .=_FO_VXP_CLIENT;
2Â $formÂ Â Â Â =Â newÂ GeneratorThemeForm($titre_form,Â 'form',Â 'index_vxp_client.php');

3Â $f_id_hiddenÂ Â =Â newÂ XoopsFormHidden("id",Â $client_id);
$form->addElement($f_id_hidden);

//Â ajoutÂ desÂ actions
4Â $op_hiddenÂ =Â newÂ XoopsFormHidden('op',Â 'store');
$form->addElement($op_hidden);
$action_buttonsÂ Â Â Â =Â newÂ XoopsFormElementTray("");
Â $submit_actionÂ Â Â Â =Â newÂ XoopsFormButton("",Â 'action',Â _AC_GEN_VALIDER,Â 'submit');
Â $action_buttons->addElement($submit_action);
Â $form->addElement($action_buttons);

//Â ajoutÂ desÂ champs
5Â $f_prenomÂ =Â newÂ XoopsFormText(_FO_PRENOM,Â 'prenom',50,150,$prenom);
$form->addElement($f_prenom,true);Â //Â obligatoire

$f_nomÂ =Â newÂ XoopsFormText(_FO_NOM,Â 'nom',50,150,$nom);
$form->addElement($f_nom,true);Â //Â obligatoire

$f_adresse_ligne1Â =Â newÂ XoopsFormText(_FO_ADRESSE_LIGNE1,Â 'adresse_ligne1',50,150, $adresse_ligne1);
$form->addElement($f_adresse_ligne1);

$f_adresse_ligne2Â =Â newÂ XoopsFormText(_FO_ADRESSE_LIGNE2,Â 'adresse_ligne2',50,150, $adresse_ligne2);

$form->addElement($f_adresse_ligne2);

$f_adresse_cpÂ =Â newÂ XoopsFormText(_FO_ADRESSE_CP,Â 'adresse_cp',10,150,$adresse_cp
);
$form->addElement($f_adresse_cp);

$f_adresse_communeÂ =Â newÂ XoopsFormText(_FO_ADRESSE_COMMUNE,Â
'adresse_commune',40,150,$adresse_commune);
$form->addElement($f_adresse_commune);

6Â $form->display();

?> Etudions le code : ->: Importation des classes xoopsform ->: CrÃ©ation du formulaire que l'on post vers l'index (index_vxp_client.php) li>: Un champ cachÃ© stockant l'id de l'enregistrement est mis dans le formulaire. Cet id est la rÃ©fÃ©rence, la clÃ© unique qui permet d'accÃ©der Ã¡ l'enregistrement de la BD lors de la soumission du formulaire vers l'index ->: Les actions possibles sur ce formulaire sont intÃ©grÃ©es. Ici nous n'en avons qu'une appelÃ© store. On utilise ici l'Ã©lÃ©ment XoopsFormElementTray qui permet d'ajouter plusieurs XoopsFormElement sur la mÃªme ligne -> : les champs de la class sont tour Ã¡ tour intÃ©grÃ©s ->: affichage du formulaire 1.4. Le contrÃ´leurLe contrÃ´leur est reprÃ©sentÃ© par l'index.php. Dans cette architecture, il y un contrÃ´leur par class, dans notre exemple ce sera index_vxp_client.php. Les principes suivants ont Ã©tÃ© retenus. Si l'objet est en modification l'identifiant est passÃ© dans la requÃªte get c'est a dire Ã¡ la fin de l'url sous la forme ?id=xxx. Dans le cas ou l'id n'est pas passÃ©, on considÃªre que l'on est en crÃ©ation. 1Â includeÂ 'header.php';
2Â includeÂ XOOPS_ROOT_PATH.'/header.php';
3Â includeÂ XOOPS_ROOT_PATH.'/modules/vetexpert/language/'.$xoopsConfig['language'].
'/main.php';
4Â includeÂ XOOPS_ROOT_PATH.'/modules/vetexpert/class/vxp_client.php';
5Â includeÂ XOOPS_ROOT_PATH.'/modules/vetexpert/action/action_manager_vxp_client.php';Â

//Â requestÂ management
//Â ----------
6Â ifÂ (isset($_POST))Â {
Â Â Â Â extract($_POST);
}
7Â ifÂ (isset($_GET))Â {
Â Â Â Â extract($_GET);
}
8Â ifÂ (!(isset($op)))Â $op='';
9Â ifÂ (!(isset($id)))Â $id='';

//Â creationÂ duÂ DAOÂ versÂ vxp_client
//Â ----------
10Â ifÂ ($id)Â {
Â Â Â Â $handler_vxp_clientÂ =Â &Â xoops_gethandler('vxp_client');
Â Â Â Â $vxp_clientÂ =Â $handler_vxp_client->get($id);
Â Â Â Â $vxp_client->unsetNew();
}Â elseÂ {
Â Â Â Â $vxp_clientÂ =Â newÂ vxp_client();

```
Â Â Â Â $vxp_client->setNew();
}
//Â actionÂ management
//Â ----------------
11Â ifÂ ($op)Â {
Â Â Â Â ifÂ (isset($_POST))Â {Â //Â chargementÂ deÂ l'objetÂ pasrÂ laÂ requete
Â Â Â Â Â Â Â Â $vxp_client->setVars($_POST);
Â Â Â Â }
Â Â Â Â //Â executeÂ theÂ "op"Â fonction
Â Â Â Â $retour=$op($vxp_client);
Â Â Â Â $msg=$retour['msg'];
Â Â Â Â redirect_header('index_vxp_client.php?id='.$retour['id'],1,$msg);
}


12Â foreachÂ ($vxp_client->getVars()Â asÂ $keyÂ =>Â $value)Â {
Â Â Â Â ifÂ (isset($value['value']))Â {
Â Â Â Â Â Â Â Â $$keyÂ =Â $value['value'];
Â Â Â Â }
}

13Â includeÂ 'form/form_vxp_client.php';

include_onceÂ XOOPS_ROOT_PATH."/footer.php";

?>
```

DÃ©crivons maintenant cet index :

6 et 7 : Chargement systÃ©matique des donnÃ©es issues du Get et des donnÃ©es issues du post en variable php. La fonction extract de PHP est ici bien utile, elle prend un tableau associatif en paramÃªtre et crÃ©e les variables dont les noms sont les index de ce tableau, et leur affecte la valeur associÃ©e.

8 et 9 : Ce code est mis afin de ne pas avoir de variable non dÃ©finie (car de la crÃ©ation par exemple)

10 : Chargement de l'objet. Si l'id n'est pas fournit on considÃªre qu'il s'agit d'un nouvel objet.

11 : Ce code gÃªre le retour suite Ã¡ validation du formulaire. On utilise la gestion des variables dynamique de PHP (Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable). Ici la valeur de la variable $op est le nom de la fonction. Dans notre exemple, cette variable op est reprÃ©sentÃ© par un champs cachÃ© aura la valeur " store ". La fonction store() sera donc exÃ©cutÃ©e.

12 : On dÃ©charge les valeurs des attributs de l'objet dans des variables afin qu'elle soit utilisÃ© dans le formulaire

13 : Chargement du formulaire. 1.5. Une liste de clients Ã¡ l'aide d'un template smartyIl est trÃªs facile d'utiliser smarty avec la classe DAO. Il suffit d'assigner le tableau des " vars " comme variable utilisable par smarty. Je dÃ©cris ci-dessous un exemple avec la liste de nos clients. Le programme liste_vxp_client.php utilise le template smarty liste_vxp_client.html.

```
includeÂ 'header.php';
includeÂ XOOPS_ROOT_PATH.'/header.php';
includeÂ XOOPS_ROOT_PATH.'/modules/vxp/class/vxp_client.php';

1Â $xoopsOption['template_main']Â =Â 'liste_vxp_client.html';
```

2Â $xoopsTpl->assign(array(
Â Â Â Â "lang_nom"Â =>Â _FO_VXP_NOM,
Â Â Â Â "lang_prenom"Â Â Â Â Â Â Â Â Â Â Â Â =>Â _FO_VXP_PRENOM
Â Â Â Â));

3Â $handler_vxp_clientÂ =Â &Â xoops_gethandler('vxp_client');
4Â $clientsÂ =Â $handler_vxp_client->getObjects();

5Â $xoopsTpl->assign('clients',$clients);

include_onceÂ XOOPS_ROOT_PATH."/footer.php";

?> 1 : description du template utilisÃ©. 2 : Assignation des variables contenant les entÃªtes de colonnes 3 4 : recherches des clients Ã¡ l'aide du handler de vxp_client. 5 : Assignation des clients Le template doit Ãªtre mis dans le rÃ©pertoire /templates, il est le suivant : tableÂ class="outer"Â cellspacing="1"Â cellpadding="4">
Â Â tr>
Â Â Â Â thÂ align="center">$lang_nom}>th>thÂ align="center">$lang_prenom}>th>
Â Â tr>

Â Â sectionÂ name=iÂ loop=$clients}>
Â Â $smarty.section.i.indexÂ isÂ evenÂ }>assignÂ var="class"Â value="even"}>
Â Â assignÂ var="class"Â value="odd"}>
Â Â
Â Â trÂ >
Â Â Â Â tdÂ class="{$class}>">
Â Â Â Â Â Â aÂ href="index_vxp_client.php?id={$clients[i]->getVar('client_id')}>">
Â Â Â Â Â Â $clients[i]->getVar('nom')}>
Â Â Â Â a>td>
Â Â Â Â tdÂ class="{$class}>">
Â Â Â Â Â Â $clients[i]->getVar('prenom')}>
Â Â Â Â td>
Â Â tr>
Â Â section}>
table> On utilise la capacitÃ© de smarty de faire rÃ©fÃ©rence aux propriÃ©tÃ©s d'un objet pour accÃ©der aux informations getVar('client_id')}>. Voir le paragraphe relatif aux objets de la documentation de smarty.Ce module de dÃ©monstration est tÃ©lÃ©chargeable ici module vxp_client