Le modÃªle objet de Xoops

CatÃ©gorie : Fiches techniques

PubliÃ© par Christian le 17/04/2005

Le modÃªle objet de XOOPSpar OryxvetLe modÃªle objet de xoops est basÃ© sur 2 classes XoopsObject et XoopsObjectHandler toutes les deux codÃ©es dans le fichier /kernel/object.php. Ces 2 classes forment la couche d'accÃ¨s aux donnÃ©es persistantes telle qu'elle est dÃ©crite dans le DAO pattern. C'est en les spÃ©cialisant que l'on implÃ©mente le modÃªle pour une entitÃ© particuliÃªre (souvent associÃ© Ã¡ une seule table de BD) comme le font toutes les classes du rÃ©pertoire kernel qui constitue le noyau de xoops. J'encourage fortement les programmeurs de nouveaux modules xoops s'appuyant sur des tables de BD Ã¡ utiliser ce modÃªle. Les raisons d'utiliser ces 2 classes sont multiples : -> uniformiser la maniÃªre de programmer et Ãªtre prÃªs des standards de codage de xoops ->augmenter la lisibilitÃ© du code et ainsi faciliter la maintenance ->fiabiliser votre code en s'appuyant sur du code xoops maintes fois validÃ© ->mais aussi bÃ©nÃ©ficier d'un certain nombre de mÃ©canismes gÃ©nÃ©riques integrÃ©s dans Xoops XoopsObject possÃªde tous les services relatifs Ã¡ la gestion d'un objet (une instance de la classe) et de ses attributs (getter et setter) alors que XoopsObjectHandler sert de manipulateur ou de contrÃ´leur des instances (insertion, modification ou sÃ©lection d'objets.
Dans une premiÃªre partie, je dÃ©cris en dÃ©tail ces 2 classes en prenant exemple sur une classe du noyau xoops XoopsPrivmessage qui s'appuie sur la table priv_msgs puis dans une seconde partie je prÃ©sente un petit gÃ©nÃ©rateur de code de ce modÃªle.XoopsObjectHandler et XoopsObjectNous allons utiliser comme exemple la table priv_msgs qui permet d'envoyer des message privÃ©s entre membres du portail.
  Figure 1 : Decription de la table priv_msgs**XoopsObject** est la classe mÃªre permettant de manipuler les attributs d'un objet donnÃ©es. Il s'appuie sur le tableau $vars pour manipuler les attributs de l'objet. $vars contient les attributs de l'objet sous la forme clÃ©, valeur. La clÃ© dÃ©signe le nom de l'attribut c'est Ã¡ dire souvent le nom d'une colonne de la table de BD. XoopsObject offre des mÃ©canismes gÃ©nÃ©riques d'accÃ¨s aux donnÃ©es ; ses principales mÃ©thodes sont :->initVar ($key, $data_type) permettant d'initialiser la dÃ©finition d'un attribut ->setVar($key, $value) qui met Ã¡ jour l'attribut ->getVar($kkey) qui restitue la valeur de l'attribut ->cleanVars () nettoie les attributs des caractÃªres spÃ©ciaux pour les stocker dans la BD La classe associÃ©e au Privmessage ne doit ainsi dÃ©finir que les attributs qu'elle veut gÃ©rer classÂ XoopsPrivmessageÂ extendsÂ XoopsObject
{

/**
Â *Â constructor
Â **/
Â Â Â Â functionÂ XoopsPrivmessage()
Â Â Â Â {
Â Â Â Â Â Â Â Â $this->XoopsObject();
Â Â Â Â Â Â Â Â $this->initVar('msg_id',Â XOBJ_DTYPE_INT,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar('msg_image',Â XOBJ_DTYPE_OTHER,Â 'icon1.gif',Â false,Â 100);
Â Â Â Â Â Â Â Â $this->initVar('subject',Â XOBJ_DTYPE_TXTBOX,Â null,Â true,Â 255);
Â Â Â Â Â Â Â Â $this->initVar('from_userid',Â XOBJ_DTYPE_INT,Â null,Â true);

Â Â Â Â Â Â Â Â $this->initVar('to_userid',Â XOBJ_DTYPE_INT,Â null,Â true);
Â Â Â Â Â Â Â Â $this->initVar('msg_time',Â XOBJ_DTYPE_OTHER,Â null,Â false);
Â Â Â Â Â Â Â Â $this->initVar('msg_text',Â XOBJ_DTYPE_TXTAREA,Â null,Â true);
Â Â Â Â Â Â Â Â $this->initVar('read_msg',Â XOBJ_DTYPE_INT,Â 0,Â false);
Â Â Â Â }
} Il existe plusieurs intÃ©rÃªts d'utiliser un tableau plutÃ´t qu'autant d'attribut de classe que de variables (couramment fait en java). Il est plus efficace en terme de temps d'exÃ©cution de manipuler de tels tableaux car ce modÃ¨le est plus proche de la structure de la requÃªte http postÃ©e. En travaillant sur un tableau on Ã©conomise le dÃ©chargement du tableau en variable. La requÃªte (issue par exemple d'un formulaire de mise Ã¡ jour d'un objet) s'utilise directement pour charger un objet. Voici un exemple ou par convention les champs HTML vont Ãªtre nommÃ©s comme le nom des attributs. $pmÂ =&Â $pm_handler->get($idPm);Â
//Â RÃ©cupÃ©rationÂ deÂ l'objetÂ
$pm->setVars(($_post)Â ;Â //Â MiseÂ Ã¡Â jourÂ deÂ l'objetÂ Ã¡Â partirÂ deÂ laÂ requÃªteÂ
$pm_handlerÂ ->insert($pm)Â ;Â //Â miseÂ Ã¡Â jourÂ physiqueÂ dansÂ laÂ BD C'est un peu lourd puisque l'on passe toute la requÃªte mais le setVars fera le " mÃ©nage " en vÃ©rifiant si la clÃ© est un attribut gÃ©rÃ© par la class. Une autre solution est de parcourir les vars de la class pour l'aller chercher que ceux dont on a besoin : foreachÂ ($pm->getVars()Â asÂ $keyÂ =>Â $value)Â {
Â Â ifÂ (isset($_POST[$key]))Â {Â
Â Â Â Â $pm->setVar($key,$_POST[$key]);
Â Â }
} Quelque soit l'option choisie, en passant l'objet au handler, la mise Ã¡ jour est stockÃ©e dans la BD : $pm_handlerÂ ->insert($pm)Â ; **XoopsObjectHandler** est une classe abstraite qui une fois implantÃ©e permet de manipuler les objets d'une class particuliÃ¨re (maclasse) Ã©tendant la class xoopsObject. Ce Handler (ou manipulateur) peut se rÃ©cupÃ©rer Ã¡ l'aide de la mÃ©thode xoops_gethandler du fichier /include/function.php en passant le nom de la classe maclasse que l'on veut manipuler. Attention toutefois car cette mÃ©thode se base sur une convention de nom, il faut avoir nommÃ© le handler xoopsmaclasseHandler. On peut aussi simplement instancier la class.
$pm_handlerÂ =&Â xoops_gethandler('privmessage');
//Â OuÂ bienÂ
$pm_handlerÂ =Â newÂ XoopsPrivmessageHandler($xoopsDB) Voici les principales mÃ©thodes proposÃ©es par ce handler qu'il s'agit d'implÃ©menter :->create : crÃ©ation d'un nouvel objet Ã¡ l'aide de la mÃ©thode (rÃ©fÃ©rence au pattern fabrique) ->insert : modification d'un objet ->delete suppression d'un objet ->get($id) : accÃ¨s direct Ã¡ un objet Ã¡ l'aide de son identifiant
Les diffÃ©rentes implÃ©mentations de XoopsObjectHandler des classes du rÃ©pertoire /kernel introduisent 2 autres mÃ©thodes bien utile non prÃ©sents dans la class mÃ¨re : ->getObjects renvoie sous forme de tableau de xoopsobjet une sÃ©lection d'objet Ã¡ partir d'un ensemble de critÃ¨res (class criteria du rÃ©pertoire /class). Ceci est bien utile pour construire un formulaire de recherche ->getCount renvoie le nombre d'occurrences d'une sÃ©lection d'objets Ã¡ partir de critÃ¨res
Typiquement voici un exemple de sÃ©lection de l'ensemble de privateMsg d' un utilisateur puis l'affichage du sujet et de la date du message : $criteriaÂ =Â newÂ Criteria('to_userid',Â $xoopsUser ->getVar('uid'));
$pmsÂ =&Â $pm_handler->getObjects($criteria);
foreachÂ ($pmsÂ asÂ $pmÂ)Â {
echoÂ $pm->getVar('subject').'-'.$pm->getVar('msg_time')Â ;
} L'utilisation avec un template smarty sera aussi simple en affectant l'ensemble de l'objet :
$pmsÂ =&Â $pm_handler->getObjects(newÂ Criteria('to_userid',Â $xoopsUser->getVar('uid')));
$xoopsTpl->assign('pms',$pm); AssociÃ© au template smarty : sectionÂ name=iÂ loop=$pms}>

$pms[i]->vars.Â subject.value}>
$pms[i]->vars.Â msg_time.value}>
section}> Class_generator : un gÃ©nÃ©rateur trÃªs simple.Pour Ã©viter le fastidieux codage de l'implÃ©mentation de ces 2 classes, nous avons dÃ©veloppÃ© un petit gÃ©nÃ©rateur qui nous a fait gagnÃ© du temps et a fiabilisÃ© et homogÃ©nÃ©isÃ© le code de type DAO. Il s'appuie sur un seul template smarty dÃ©crivant le fichier des 2 classes implÃ©mentent respectivement XoopsObject et XoopsObjectHandler. Le template peux bien entendu Ãªtre modifiÃ© pour prendre en compte les besoins spÃ©cifiques. Ce gÃ©nÃ©rateur utilise uniquement les informations issues de la base de donnÃ©es ce qui impose d'avoir dÃ©jÃ¡ une table de base de donnÃ©e sur laquelle s'appuyer. Class_generator gÃ©nÃªre un fichier par table de Base de donnÃ©es. Il se base aujourd'hui sur la convention que la table source ne doit possÃ©der qu'une seule clÃ© primaire. Ceci pour gÃ©nÃ©rer la mÃ©thode du handler getId($maClePrimaire). Une fois le module installÃ©, l'accÃ¨s au gÃ©nÃ©rateur s'effectue dans la partie administration. Dans le formulaire " GÃ©nÃ©rer une class " aprÃªs avoir sÃ©lectionnÃ© le module sur lequel on travaille et la table de BD existante, le clic sur le bouton " gÃ©nÃ©rer " gÃ©nÃªre un fichier nommÃ© du nom de la table de BD sÃ©lectionnÃ©e dans le rÃ©pertoire /class du module.Le module est tÃ©lÃ©chargable sur le site dev.oryxvet.com OryxVet