

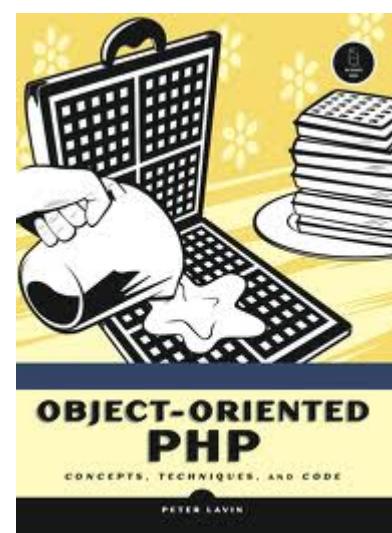
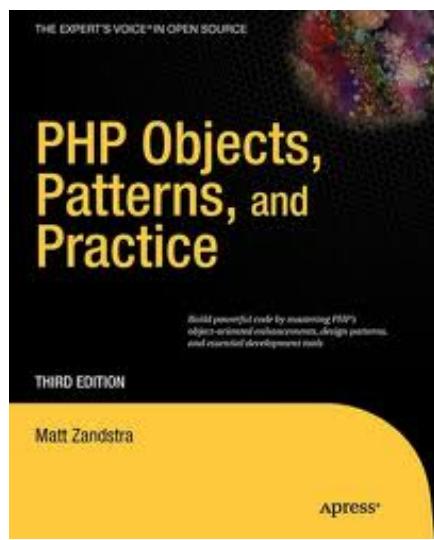
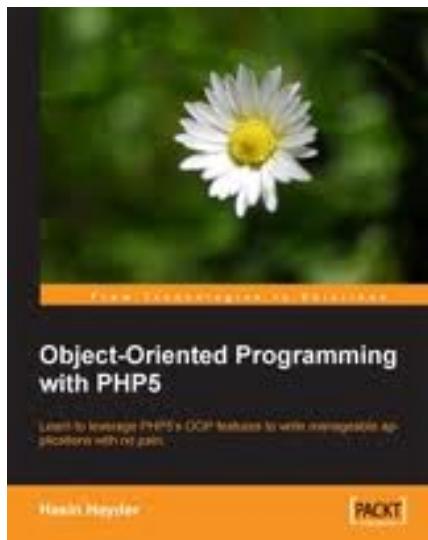


**XOOPS**  
powered by you

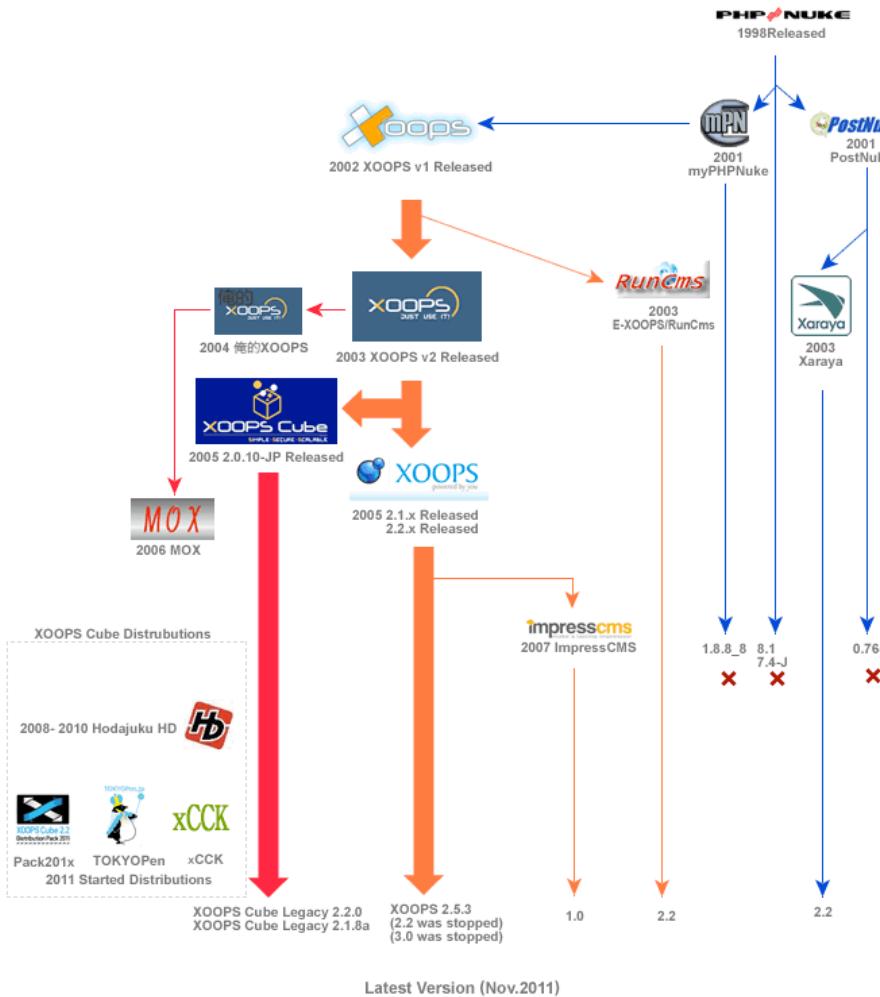
# OOP Adventures with XOOPS

Michael Beck  
[www.xoops.org](http://www.xoops.org)

# PHP and OOP



# History of XOOPS

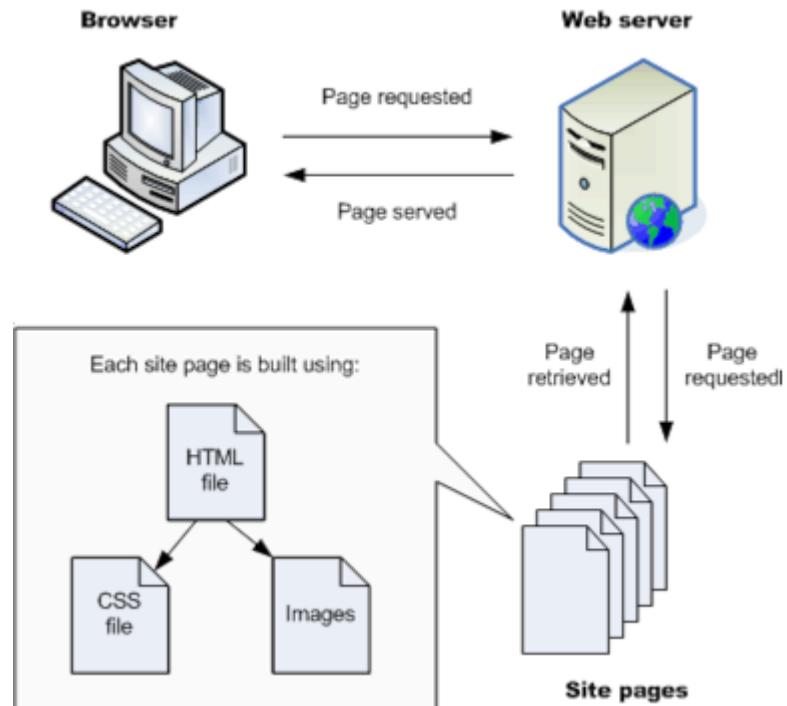


eXtensible  
Object  
Oriented  
Portal  
System

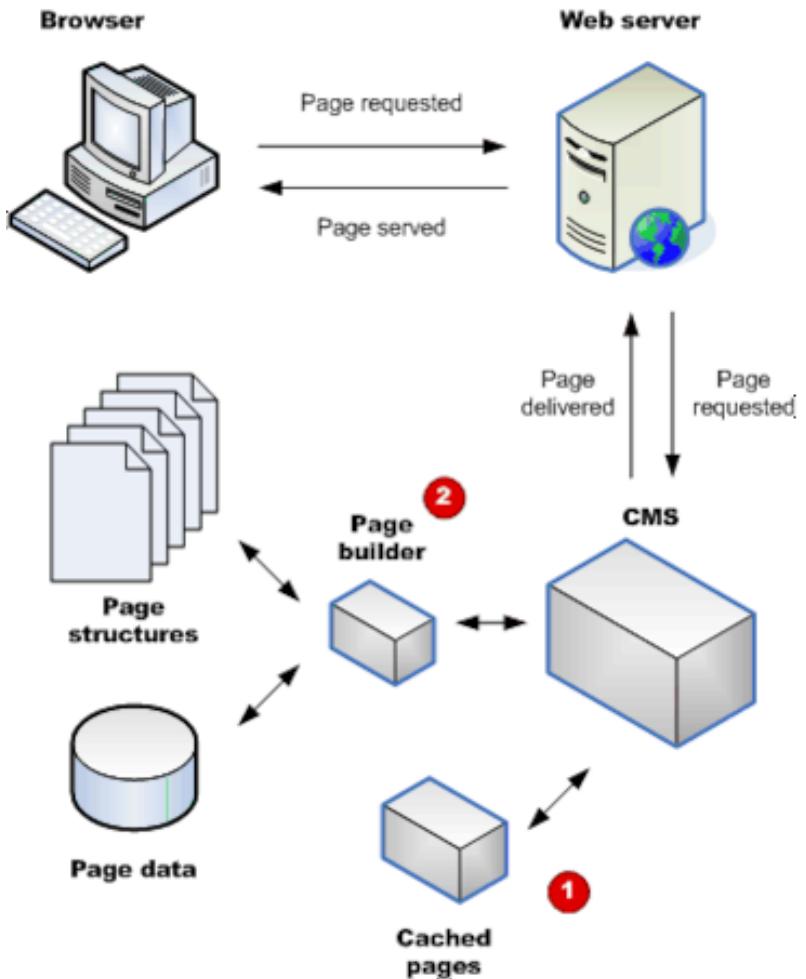
Pronounced « zups »

# What is CMS?

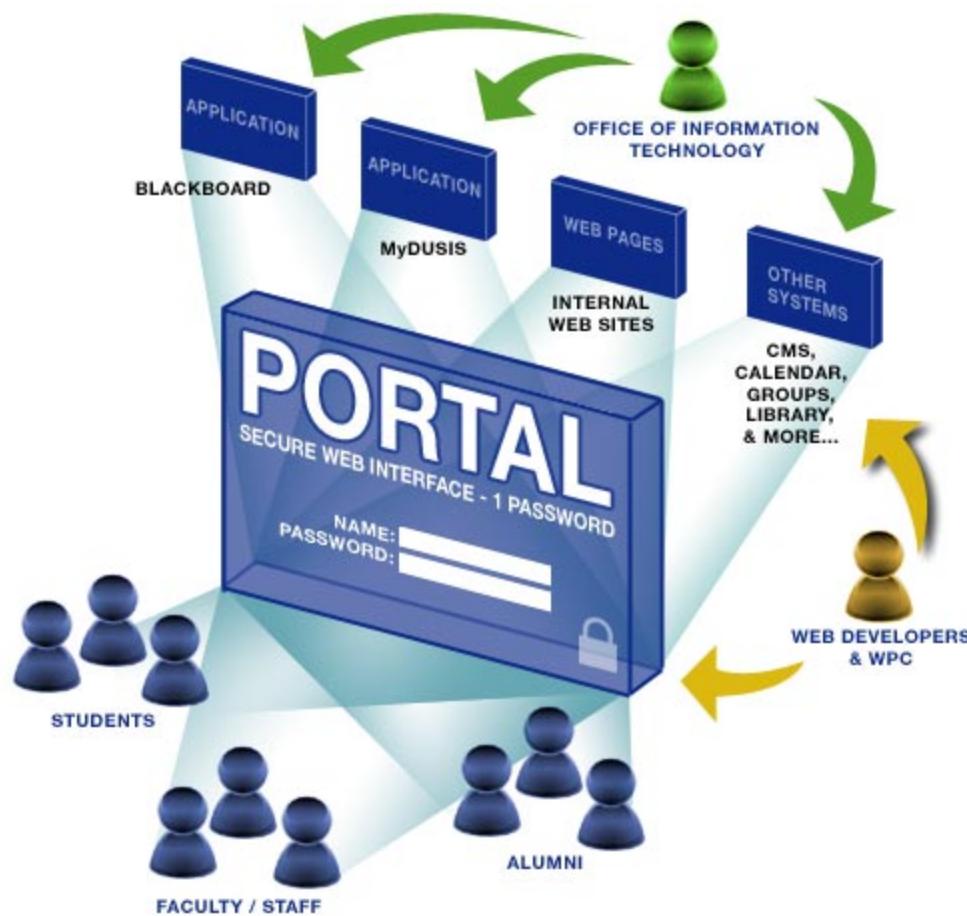
## Static Page Management



## Dynamic Page Management



# What is CMS Portal?



# Some of XOOPS Users

Some of the most prolific users include:

- Centos ([www.centos.org](http://www.centos.org))
- Libyan Ministry of Defense (<http://www.defense.gov.ly/>)
- Libyan National Election Commission (<http://www.hnec.ly/>)
- Brazilian equivalent of our CIA (<http://www.abin.gov.br/>)
- all government Website of State of Parana in Brazil are built with XOOPS:

<http://www.parana.pr.gov.br/>

<http://www.turismo.pr.gov.br>

- Ohio/Indian/Northern Kentucky PHP User Group ([www.oink-pug.org](http://www.oink-pug.org))
- Fukui Prefecture (Japan)
- Computerworld.dk (Denmark)
- Goldseiten.de (Germany)
- Koreus.com (France)



# Highlights of XOOPS

## **Database-driven**

XOOPS uses a relational database (currently MySQL) to store data required for running a web-based content management system.

## **Fully Modularized**

Modules can be installed/uninstalled/activated/deactivated with a click using the XOOPS module administration system. Over 500 modules are [available for download](#)

## **Personalization**

Registered users can edit their profiles, select site themes, upload custom avatars, and much more!

## **User Management**

You can search for users by various criteria, send email and private messages to users through a template-based messaging system.

## **Supported World-wide**

XOOPS was created and is maintained by a team of several hard-working volunteers working from all over the world. The XOOPS community has more than dozen official support sites around the world for support of non-English speaking users. Since it is Open Source (under GPL), it can be freely modified, and the modifications contributed back to the project

## **Multi-byte Language Support**

Fully supports multi-byte languages, including Japanese, Simplified and Traditional Chinese, Korean, etc., with L-to-R support.

## **Versatile Group Permissions System**

Powerful and user-friendly permissions system which enables administrators to set permissions by group.

## **Theme-based skinnable interface**

XOOPS is driven by a powerful theme system. Both admins and users can change the look of the entire web site with just a click of a mouse. There are also over 1000 themes [available for download!](#)

## **Caching**

Excellent caching capability, reducing the load on the server. It's more highly configurable than many other CMS solutions.

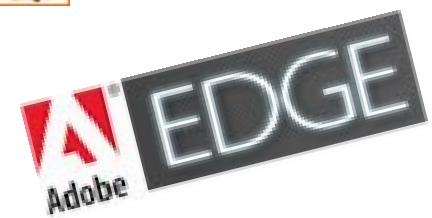
# XOOPS Awards



Best Technology Award



Pack Publishing:  
Top-5 finalist in 2010 Best CMS Award category



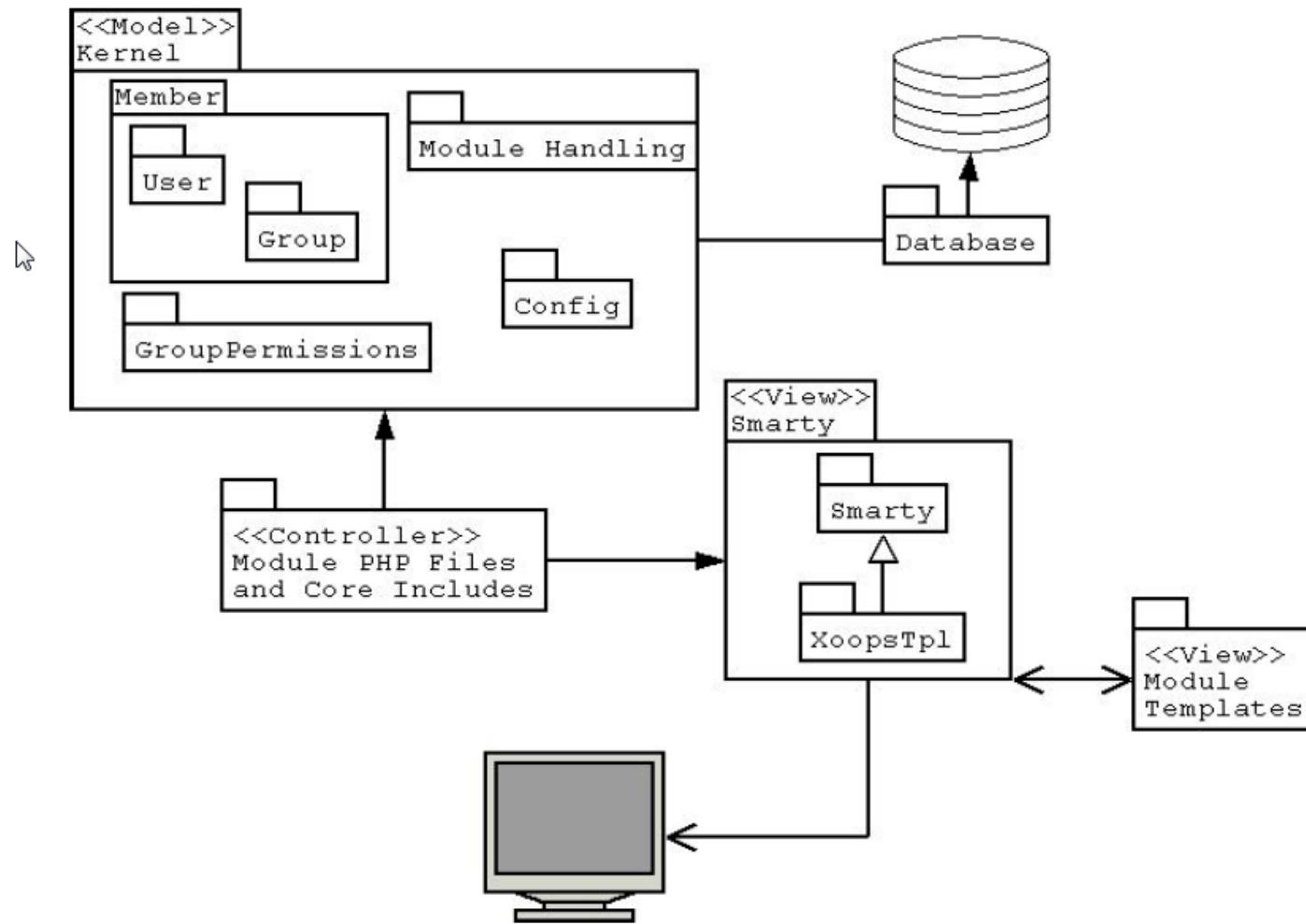
First Runner Up  
development category



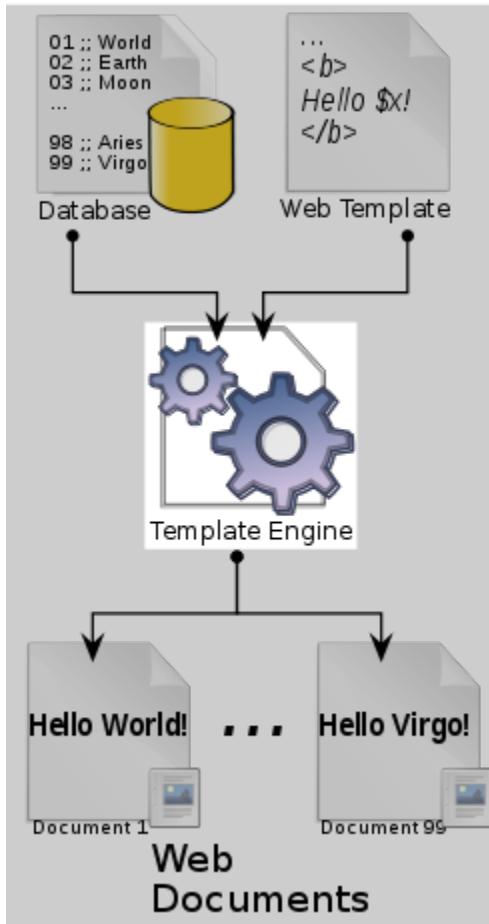
Pack Publishing:  
- Top-5 finalist in 2009 Best Overall CMS Award category



# XOOPS Architecture is MVC-based



# XOOPS and Smarty (View)



A ([web](#)) **template engine** is software that is designed to process [web templates](#) and *content information* to produce output [web documents](#). It runs in the context of a [template system](#).

Smarty is a template engine for PHP, facilitating the separation of presentation (HTML/CSS) from application logic. This implies that PHP code is application logic, and is separated from the presentation.

Benefits of using template engines include:

- Encouraging organization of source code into operationally-distinct layers (see e.g., [MVC](#))
- Enhancing productivity by reducing unnecessary reproduction of effort
- Enhancing teamwork by allowing separation of work based on skill-set (e.g., artistic vs. technical)

Class **xoopsTpl** (file class/template.php) is a subclass of the class Smarty. All Smarty class methods can be used.

# XoopsObject & XoopsPersistableObjectHandler

Xoops has two classes to aid in the class development **XoopsObject** and **XoopsPersistableObjectHandler**

The idea behind them is that a class can extend XoopsObject to describe an object, whereas extending XoopsPersistableObjectHandler will give more like an interface for handling the objects, i.e. get, insert, delete and create objects. E.g. for a *ThisObject* class, you create a *ThisObjectHandler* to get, insert, delete and create ThisObject objects.

The advantages of extending these two classes are: for XoopsObject:

*Automatic access (inheritance) to methods, easing the assignment/retrieval of variables  
Automatic access to methods for cleaning/sanitizing variables*

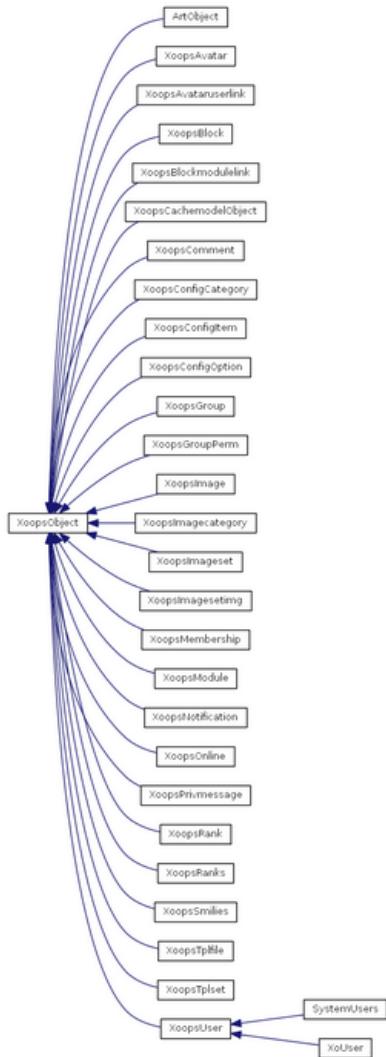
and for XoopsPersistableObjectHandler:

*A place to put all those functions working with more than one object (e.g. a "getAllObjects()" function).*

These functions will become easier to track down in the file system (since they are connected to a class, it is just a matter of finding the class and not going through the function files in the module/core/PHP native in search for it).

An additional idea is that the XoopsPersistableObjectHandler-extending class should be a Data Access Object, i.e. the class, which handles database calls - and leaving the XoopsObject-extending class to have object-describing methods, such as methods which handle and manipulate variables, calling methods on the handler for retrieving, updating and inserting data in the database.

# XoopsObject



XoopsObject should be handled as an abstract class and has the following useful functions:

- \* initVar(\$key,\$data\_type,\$value,\$required,\$maxlength,\$options) - initialize a variable. Use this, when extending XoopsObject instead of declaring variables the normal way. for \$data\_type, see below. \$options is string for select options
- \* getVar(\$key,\$format) - retrieves an object variable (\$format is 's' for display on page, 'f' for form info, or 'n' for no cleaning).
- \* getVars() - retrieves array of key=>value pairs
- \* cleanVars() - "cleans" the object variables, which have changed, according to their type set in initVar()
- \* clone() - create a clone(copy) of the current object
- \* setVar(\$key,\$value,\$not\_gpc) - assign a value \$value to a variable \$key; sets object dirty
- \* setVars(\$var\_array) - assign values to variables from a key=>value pair array; sets object dirty
- \* assignVar(\$key,\$value) - assign a value to a variable
- \* assignVars(\$var\_array) - assign values to variables from a key => value pair array

# Some XoopsObjects in XOOPS

**\$xoopsConfig** : general configuration for the site

- \$xoopsConfig['adminmail']
- \$xoopsConfig['slogan']

**\$xoopsUser** : object representing the user currently logged in

- \$xoopsUser->uid()
- \$xoopsUser->uname()

**\$memberHandler** : object handling users and groups

- \$memberHandler->getGroups()
- \$memberHandler->addUserToGroup()

# Some XoopsObjects in XOOPS

**\$xoopsModule** : object representing the actual module

**\$xoopsNotificationHandler** : object handling notifications

- \$xoopsNotificationHandler->subscribe()
- \$xoopsNotificationHandler->triggerEvent()

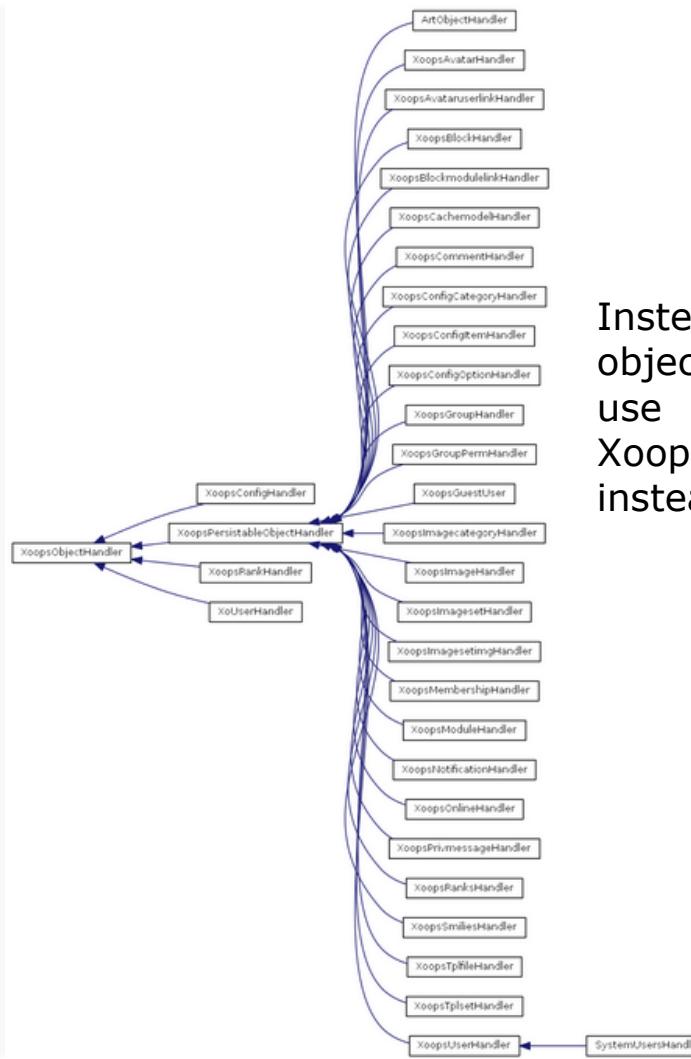
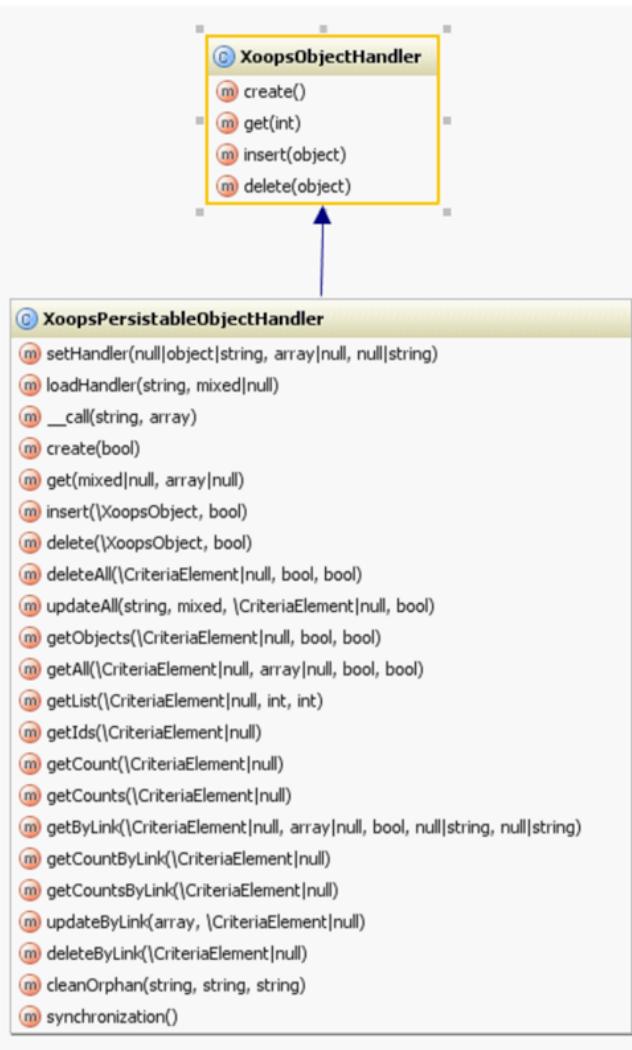
**\$xoopsTpl** : representing a Smarty Object

- \$xoopsTpl->display('my\_template.html')

**\$xoopsForm** : object representing a web form and allowing to handle its controls

**\$xoopsMailer** : object handling sending emails

# XoopsPersistableObjectHandler



Instead of instantiating the objects directly, the idea is to use XoopsPersistableObjectHandler instead.

# XoopsPersistableObjectHandler

An object can be retrieved from the database with this code:

```
$class_handler =&xoops_gethandler('classname');//NOT classnameHandler  
$thisobject =&$class_handler->get($objectid);
```

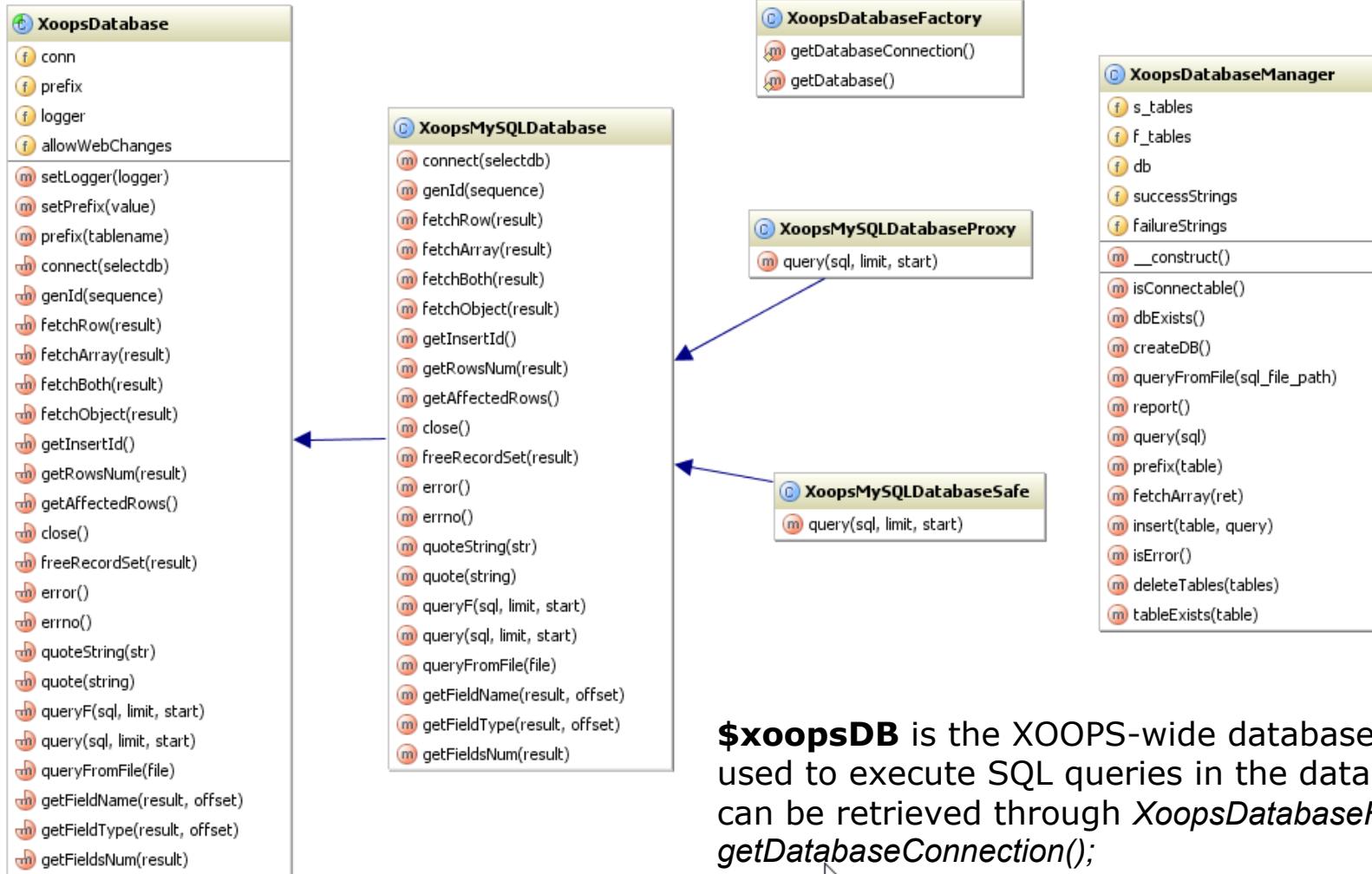
This is for an existing object in the database. To create a new object, use this:

```
$class_handler =&xoops_gethandler('classname');//NOT classnameHandler  
$thisobject =&$class_handler->create();
```

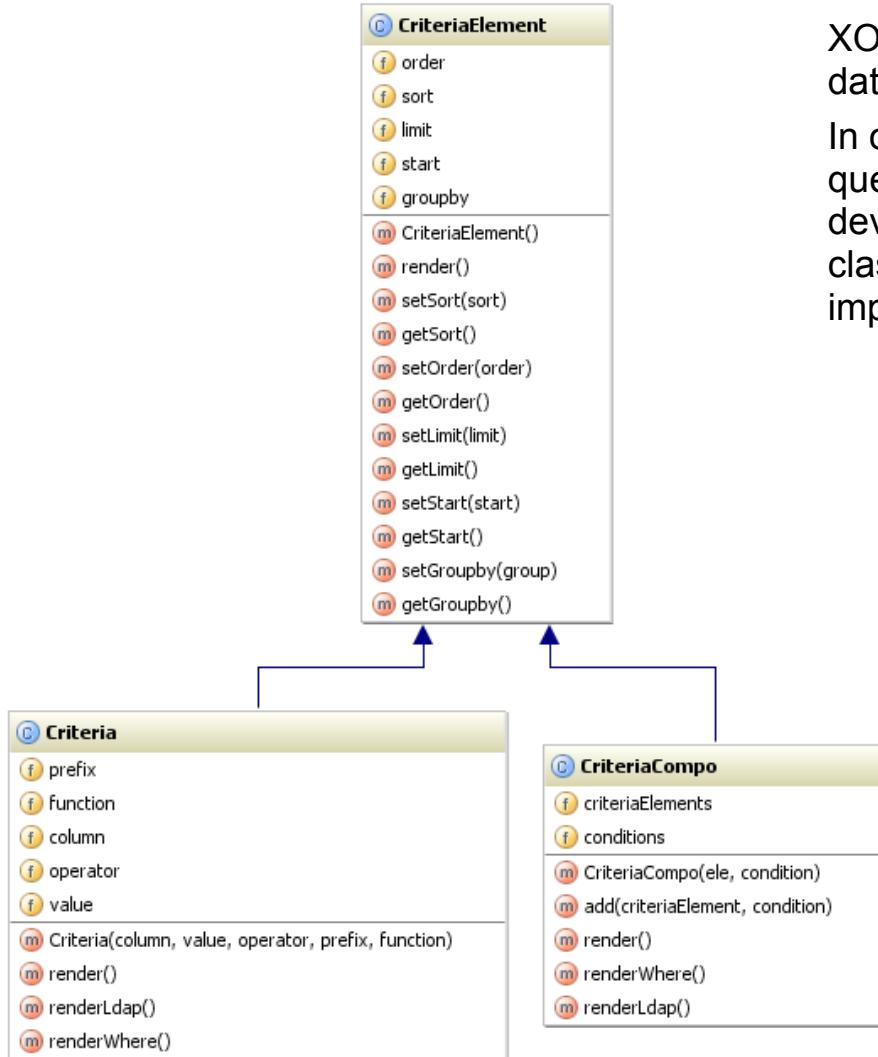
This is for core classes. Module class handlers should be fetched with

```
$class_handler =&xoops_getmodulehandler('classname', 'dirname');//NOT classnameHandler
```

# Objects and Databases



# Objects and Databases: Criteria



XOOPS provides the `Criteria` class to create a database query syntax.

In order to ensure efficiency and security of the data query library, it is recommended that module developers use `XoopsPersistableObjectHandler` class combined with the `Criteria` class that implements the operation of the database.

# Objects & Databases: Criteria

Criteria is used to create a single query statement

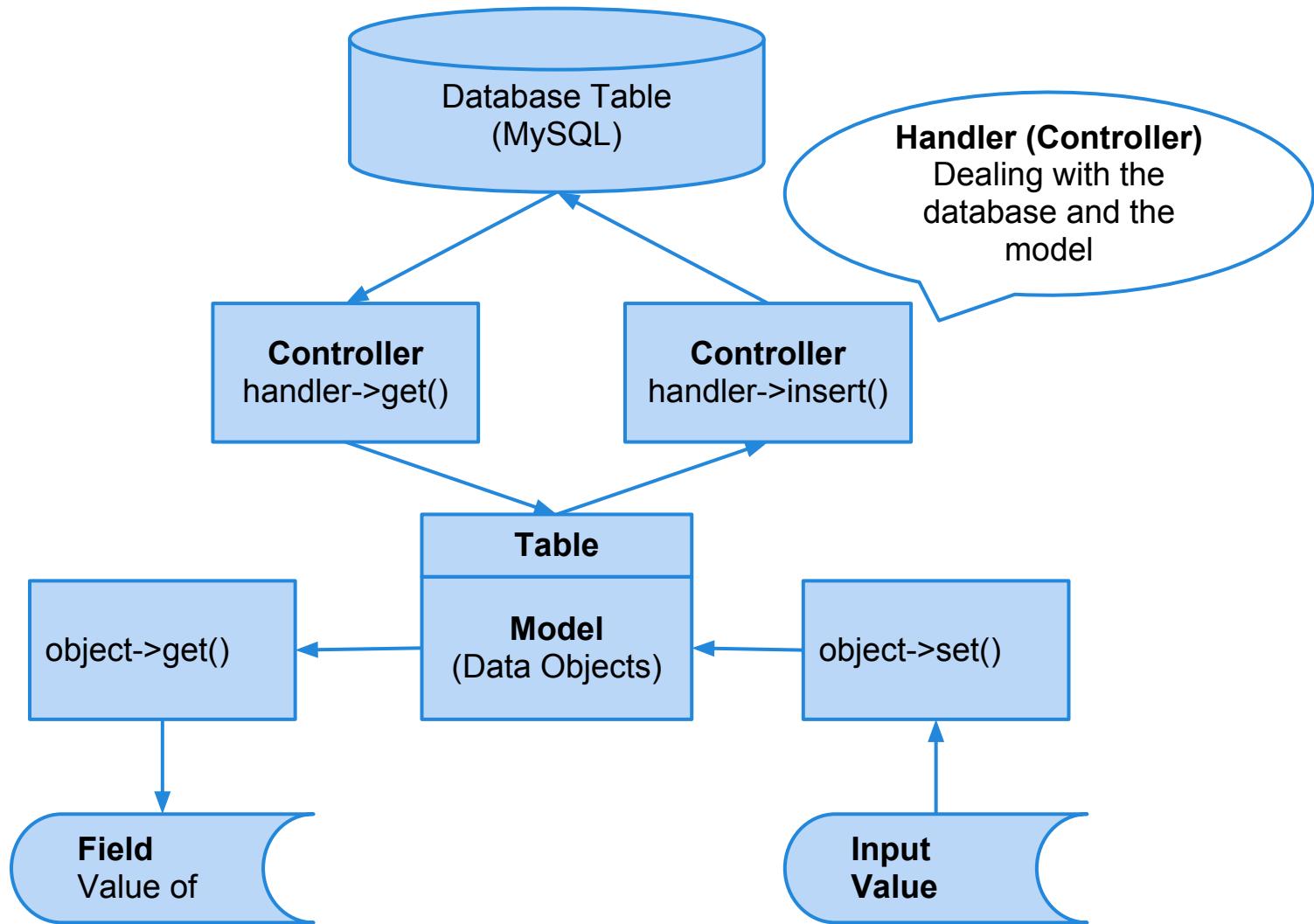
1. `$criteria = new Criteria("foo_id", 100);`
2. `// Query resolution, generally called by XoopsPersistableObjectHandler where`
3. `$criteria->render();`
4. *//The instance will generate the following conditional statement:*
5. *//"[... where] foo\_id=100";*

# Objects & Databases: CriteriaCompo

**CriteriaCompo** is used to create multi-condition combination query

```
1. <?php
2. $criteria      = new CriteriaCompo();
3. $criteria_foo_id = new Criteria("foo_id", 100);
4. $criteria->add($criteria_foo_id);
5. // Add a conditional AND operation
6. $criteria->add(new Criteria("foo_status", 1, "="), "AND");
7. //Or
8. $criteria->add(new Criteria("foo_status", 1));
9. // Add a condition of the OR operation
10. $criteria->add(new Criteria("foo_level", 100, ">"), "OR");
11. // Add another combined query
12. $criteria_another = new CriteriaCompo(new Criteria("foo_category", "test"));
13. $criteria_another->add(new Criteria("foo_owner", "administrator"));
14. $criteria->add($criteria_another, "OR");
15. // Query resolution, generally called XoopsObjectHandler
16. $criteria->render();
17. // the instance will generate the following conditional statement
18. "[... where] (foo_id = 100 AND foo_status = 1 OR foo_level > 100) OR (foo_category = 'test' AND
   foo_owner = 'administrator')";
19. ?>
```

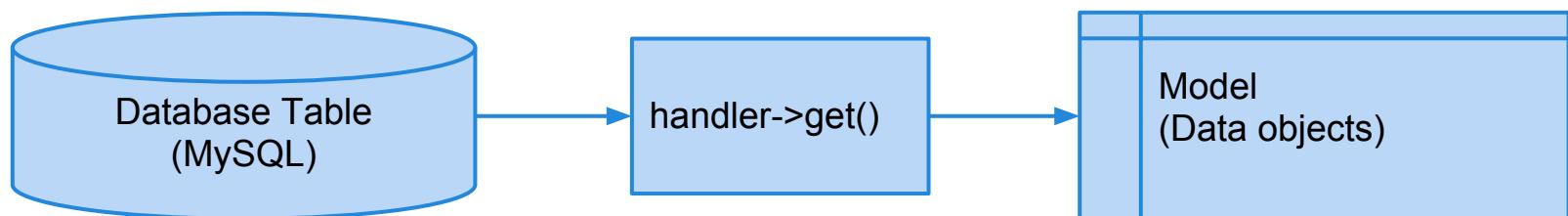
# Objects and Database Interactions



# Getting Object through Handler

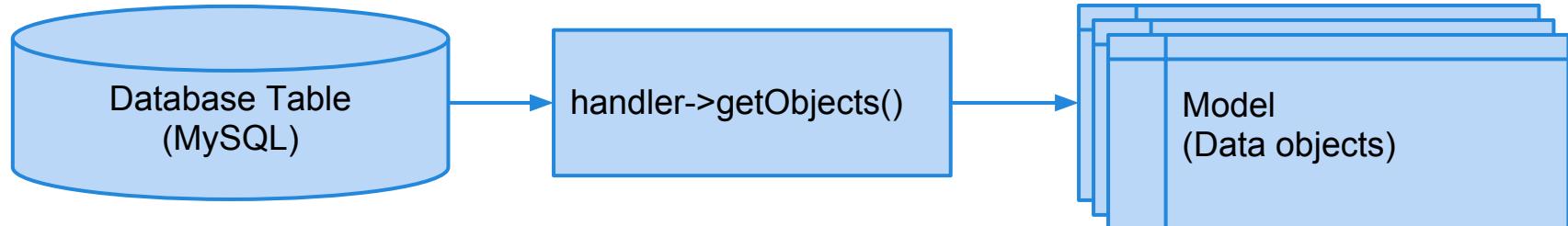
Get data from a single object instance using Handler, and specifying primary key:

```
1. // modules/newbb/viewforum.php
2. // Get the forum ID
3. $forum_id = intval($_GET['forum']);
4. // Create NewbbForumObjectHandler
5. $forum_handler =&xoops_getmodulehandler('forum', 'newbb');
6. // Instantiate NewbbForumObject
7. $forum_obj =& $forum_handler->get($forum_id);
```



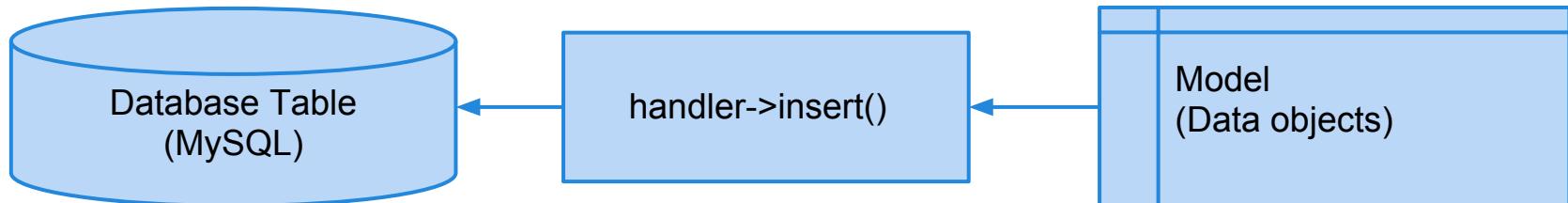
# Get all the Objects in the table

```
1. // modules/newbb/moderate.php
2. // Determine the the forum_id need to meet the conditions
3. $criteria = new Criteria("forum_id", "(0, {$forum_id})", "IN");
4. // Determine the maximum amount of data to be read
5. $criteria->setLimit($xoopsModuleConfig['topics_per_page']);
6. // Read the initial position
7. $criteria->setStart($start);
8. // Sort based on the rule that the field names
9. $criteria->setSort($sort)
10. // Forward or reverse
11. $criteria->setOrder($order);
12. // Read data to meet the conditions and instantiated the $moderate_objs
13. $moderate_objs = $moderate_handler->getObjects($criteria);
```



# Add/Update Object in the database

```
1. // modules/newbb/admin/admin_cat_manager.php
2. // Create a new category object
3. $category_obj =& $category_handler->create();
4. ...
5. // Assignment from the POST data submitted
6. $category_obj->setVar('cat_title', @$_POST['title']);
7. $category_obj->setVar('cat_image', @$_POST['cat_image']);
8. $category_obj->setVar('cat_order', $_POST['cat_order']);
9. $category_obj->setVar('cat_description', @$_POST['cat_description']);
10. $category_obj->setVar('cat_url', @$_POST['cat_url']);
11. ...
12. // Object data is inserted into the database
13. if (!$category_handler->insert($category_obj)) {
14. ...
15. }
```

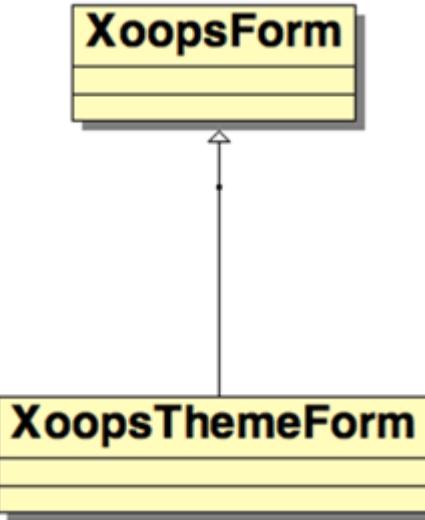
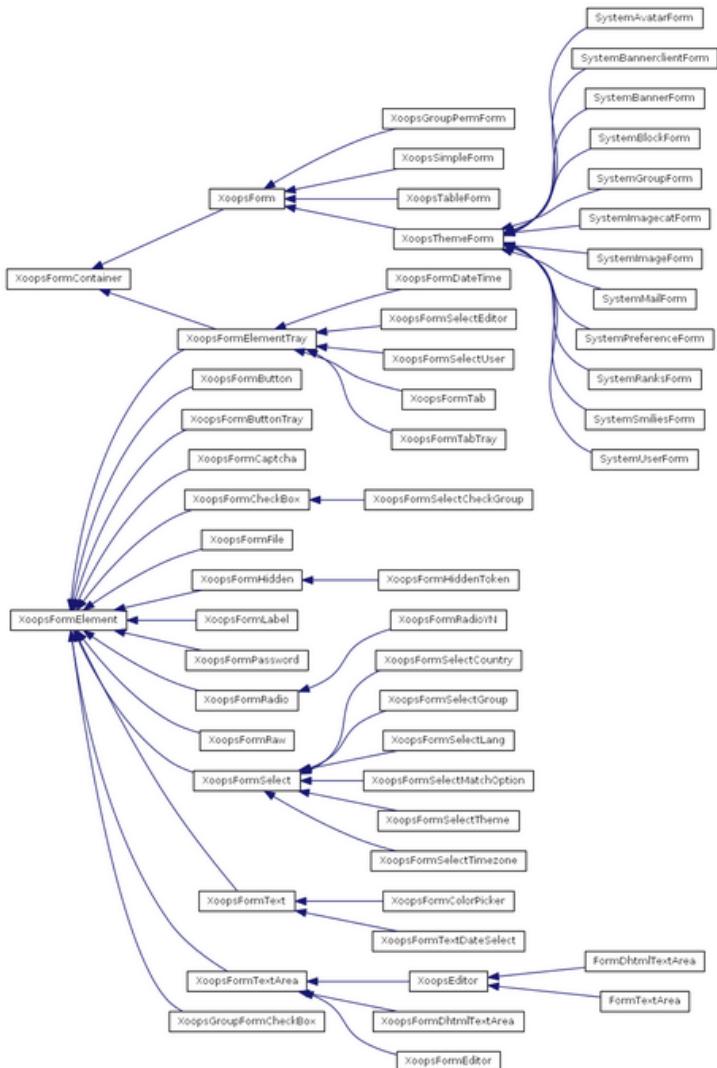


# Initialize the Object manually

Objects can be retrieved from the database, but they also can be initialized manually with predefined values:

```
1. class article extends XoopsObject
2. {
3.     //Constructor
4.     function __construct()
5.     {
6.         global $xoopsModuleConfig;
7.         $this->XoopsObject();
8.         $this->initVar("modules_id", XOBJ_DTYPE_INT, 0, false, 5);
9.         $this->initVar("modules_name", XOBJ_DTYPE_TXTBOX, $xoopsModuleConfig["module_name"], false);
10.        $this->initVar("modules_version", XOBJ_DTYPE_TXTBOX, $xoopsModuleConfig["module_version"], false);
11.    }
```

# Inheritance with Form Objects



In our first example we want our form to create a different view, depending on a User action.

Currently XoopsThemeForm class does not change form's action parameter after declaring the constructor - this parameter is passed to the constructor only once, but we want to be able to change it depending on actions of the users and how they use the form.

# Inheritance with Form Objects



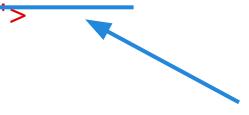
# Inheritance with Form Objects

```
function render()
{
    $ele_name = $this->getName();
    $ret = '<form name="" . $ele_name . " id="" . $ele_name . "' action="" . $this->getAction() . '" method="" . $this-
>getMethod() . " onsubmit="returnxoopsFormValidate_'. $ele_name . '();"' . $this->getExtra() . '>
        <table width="100%" class="outer" cellspacing="1">
...
}
```



We need to override the function render(), we replace this->getAction by our new property \$newAction, which gives:

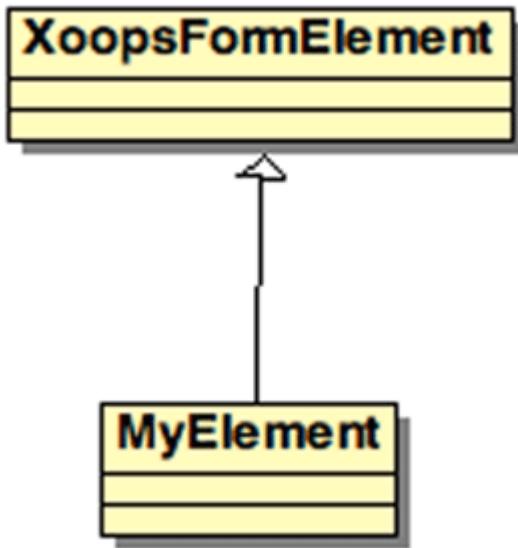
```
function render()
{
    $ele_name = $this->getName();
    $ret = '<form name="" . $ele_name . " id="" . $ele_name . "' action="" . $this->newAction. '" method="" . $this-
>getMethod() . " onsubmit="returnxoopsFormValidate_'. $ele_name . '();"' . $this->getExtra() . '>
        <table width="100%" class="outer" cellspacing="1">
...
}
```



```
$my_form=new MyForm(parameter);

If(condition){
// Now we change the action URL of the form
$MyForm->setNewAction('test.php')
}
```

# Inheritance with Form Objects 2



Here, we want our form to display images previously uploaded to the server.

There is no element in the forms of XOOPS to display an image. So we will create it by extending the class XoopsFormElement, which is the class responsible for creating a form elements, such as FormColorPicker etc.

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

class MyElement extends XoopsFormElement
{
    var $_content;

    function __construct($caption = "", $value = "", $name = "")
    {
        $this->setCaption($caption);
        $this->setName($name);
        $this->_value = $value;
    }

    function setContent($content)
    {
        $this->_content=$content;
    }

    function getContent($encode=false){
        return $encode ? htmlspecialchars($this->_content, ENT_QUOTES) : $this->_content;
    }
    function render()
    {
        return $this->getContent();
    }
}
```

---

```
$myPhoto=new myElement();
$myPhoto->setContent('<div class="photos">photos</div>');
$my_form->addElement($myPhoto);
```

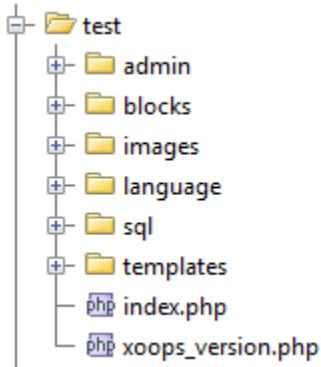
# XOOPS Modules

A module for XOOPS is similar to a program on your desktop PC. It is basically a Web-based application.

Generally, a module provides a number of pages, adding functionality to the site such as announcements/news as in the News module or a bulletin board as in the NewBB module.

Most modules also provide blocks for showing the latest added or most popular content that can be placed on the pages of any module, providing fast access to other areas of the site from within a module's pages.

# XOOPS Module File Structure



Module PHP files are accessed by the user through the browser's address field. These files must include a XOOPS core file, which will

1. Include the XOOPS API functions and classes
2. Instantiate the Database object, setting up the connection to the database
3. Instantiate the user object and retrieve user data if the user is logged in
4. Instantiate the current module's object, containing information about the current module
5. Retrieve the current module's configuration options from the database and save it in a globally available array variable
6. Include relevant language files for localized language.

The next step is to include the root file header.php which will create an instance of XoopsTpl, a XOOPS-specific child class to the Smarty class supplied by a third party

# TDMCreate: Module Generator

**TDMCreate**

Preferences | Update | Blocks | Templates | Comments | Uninstall | Go to module      TDMCreate : Index

Index Modules Tables Build Module About Help

**Index**

Modules Tables Build Module About Help

**Statistics**

There are 1 modules stored in the Database  
There are 0 tables stored in the Database

**Configuration Check**

✓ Minimum PHP required: 5.3 (your version is 5.3.15)  
✓ MYSQL minimum version required: 5.0.7 (your version is 5.5.20-log)  
✓ Minimum XOOPS required: 2.5.5 (your version is 2.5.6 Alpha)  
✓ Minimum ModuleAdmin required: 1.1 (your version is 1.1)

XOOPS  
TDMCreate is maintained by the XOOPS Community

# VIDEO

<http://www.youtube.com/watch?v=dg7zGFCopxY>

# Generated XOOPS Module Objects

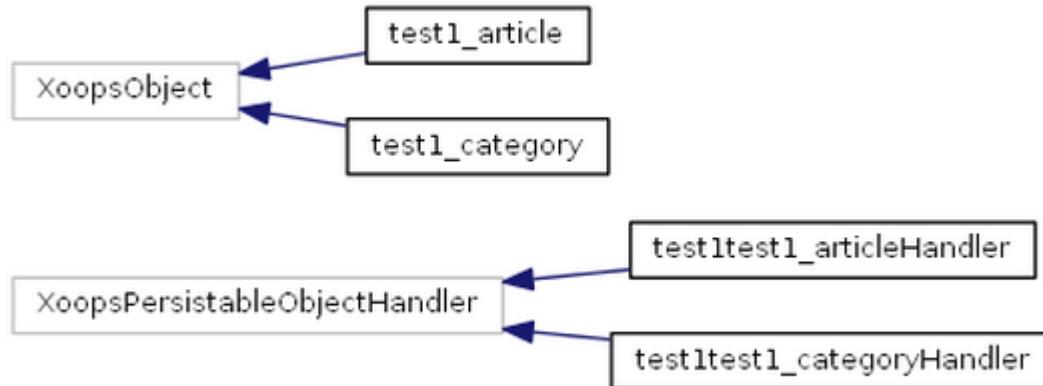


Table: Category

Screenshot of the MySQL database structure for the "category" table:

localhost » xtest256a3 » x9c5_mod_test1_category					
Browse		Structure		SQL	
#	Name	Type	Collation	Attributes	Null
1	<a href="#">category_id</a>	int(11)		UNSIGNED	No
2	<a href="#">category_pid</a>	int(5)		UNSIGNED	No
3	<a href="#">category_title</a>	varchar(255)	utf8_general_ci		No
4	<a href="#">category_desc</a>	text	utf8_general_ci		No
5	<a href="#">category_img</a>	varchar(255)	utf8_general_ci		No
6	<a href="#">category_weight</a>	int(5)			No
7	<a href="#">category_color</a>	varchar(10)	utf8_general_ci		Yes

Table: Article

Screenshot of the MySQL database structure for the "article" table:

localhost » xtest256a3 » x9c5_mod_test1_article					
Browse		Structure		SQL	
#	Name	Type	Collation	Attributes	Null
1	<a href="#">article_id</a>	int(8)			No
2	<a href="#">article_title</a>	varchar(255)	utf8_general_ci		No
3	<a href="#">article_content</a>	text	utf8_general_ci		No
4	<a href="#">article_status</a>	int(10)			No
5	<a href="#">article_waiting</a>	int(10)			No
6	<a href="#">article_online</a>	tinyint(1)			No

# **THANK YOU!**

Please send feedback to:  
**mambax7@gmail.com**



**[www.xoops.org](http://www.xoops.org)**