



Libraries

- Import of libraries through `jimport()` function
- `jimport('joomla.application.component.controller');`
- `jimport('joomla.application.helper');`
- `jimport('geshi.geshi');`
- Available libraries
 - `domit` – Lightweight DOM parser for PHP
 - `geshi` – Generic Syntax Highlighter
 - `pear` – PEAR packages
 - `phpgacl` – Generic Access Control Lists
 - `phputf8` – Handling UTF8 characters in a non-UTF8 environment
 - `tcpdf` – PHP class for generating PDF documents
 - `joomla` – The Joomla! Framework



Joomla! MVC rules

- Database interactions only through JModel, JTable or helpers
- Data are never modified by JView
- Keep PHP in layout-files at a minimum
- Use one JController-method per task (edit, save, add, publish)



Benefits of MVC

Solid structure for components

- Internal logic separated from design
- Easier to maintain than random code-structures
- Open source standard for all components

Extra Joomla! functionality

- Multiple menu-items per layout
- Database abstraction through JTable
- Template overrides for every layout-file



MVC workflow

- Joomla! bootstrap procedure calls “example.php”
- “example.php” initializes ExampleController from “controller.php”
- JController
 - “example.php” calls JController::execute()
 - JController::execute() calls MyController::display()
 - MyController::display() calls JController::display()
- JView
 - JController::display() initializes MyView from “views/item/view.html.php”
 - JController::display() calls MyView::display()
 - MyView::display() fetches data from JModel
 - MyView::display() initializes variables for layout-files
 - MyView::display() calls JView::display()
- JView::display() calls layout file and buffers it to JDocument



Using tasks (2)

example.php

```
$controller->execute( JRequest::getCmd('task'));
```

controller.php

```
function __construct() {  
    parent::__construct();  
    $this->registerTask( 'add', 'display' );  
}
```

```
function display() {  
    if( $this->getTask() == 'add' ) { ... }  
}
```

```
// Added automatically to task map  
function edit() { ... }
```



ExampleViewItem extends JView

```
defined( 'JEXEC' ) or die();  
  
jimport( 'joomla.application.component.view' );  
  
class ExampleViewItem extends JView {  
    function display( $tpl = null ) {  
        ...  
        ...  
    }  
}
```



Searching in different paths for MVC classes

- JController:
 - `$this->addViewPath(JPATH_COMPONENT_ADMINISTRATOR.DS.'views');`
 - `$this->addModelPath(JPATH_COMPONENT_ADMINISTRATOR.DS.'models');`
 - `JModel::addIncludePath(JPATH_COMPONENT_ADMINISTRATOR.DS.'models');`
 - `include_once JPATH_COMPONENT.DS.'helpers'.DS.'helper.php' ;`
- JView:
 - `JModel::addIncludePath(JPATH_COMPONENT_ADMINISTRATOR.DS.'models');`
 - `$this->addHelperPath(JPATH_COMPONENT.DS.'helpers');`
 - `$this->addTemplatePath(JPATH_COMPONENT.DS.'tmp');`
- JModel:
 - `JTable::addIncludePath(JPATH_COMPONENT_ADMINISTRATOR.DS.'tables');`
 - `$this->addTablePath(JPATH_COMPONENT_ADMINISTRATOR.DS.'tables');`
 - `include_once JPATH_COMPONENT.DS.'helpers'.DS.'helper.php' ;`



Common database fields

Field	Type	Description
id	int(11)	MySQL identifier
title	text	Title of an item
alias	varchar(255)	Alias of the title
published	tinyint(1)	Boolean
checked_out	int(11)	Reference to user ID
checked_out_time	datetime	Time of checkout
hits	int(11)	Hits counter
ordering	int(11)	Ordering of item
access	tinyint(3)	Reference to ACL ID
params	text	Serialized array of parameters



XML installation file (8): Modules & plugins

```
<modules>
```

```
  <module module="mod_X" title="My module" position="left" order="1"
  client="site">
```

```
    <files folder="mod_X">...</files>
```

```
  </module>
```

```
</modules>
```

```
<plugins>
```

```
  <plugin plugin="Y" order="100" group="content">
```

```
    <files folder="plugin_X">...</files>
```

```
  </plugin>
```

```
</plugins>
```



Parameters (4a): Build your own parameter

File “administrator/components/com_example/elements/increment.php”:

```
class JElementIncrement extends JElement
{
    function fetchElement($name, $default_value, &$amp; $node, $control_name)
    {
        $options = array() ;
        for( $i = 0; $i <= 30; $i++ ) {
            $o = new JObject() ;
            $o->id = $i ;
            $o->name = $i ;
            $options[] = $o ;
        }
        return JHTML::_('select.genericlist', $options, $control_name.'['.$name.']", ",
            'id', 'name', $default_value);
    }
}
```



JFilterInput

```
jimport('joomla.filter.filterinput');
```

```
JFilterInput::clean( $mixed, $type )
```

\$mixed = source string or array containing strings

\$type = int, float, boolean, word, alnum, cmd, base64, string, array, path, none

```
JFilterInput::checkAttribute( $array )
```

\$array = array of two values (\$name and \$value)

\$value is checked to make sure it doesn't contain the following strings:

expression, style, javascript:, behaviour:, vbscript:, mocha:, livescript:



Sliding panes

view.html.php

```
jimport('joomla.html.pane');  
$pane = & JPane::getInstance('sliders');  
$this->assignRef('pane', $pane);
```

layout file:

```
echo $this->pane->startPane("mypane");  
echo $this->pane->startPanel( $title, "details" );  
echo $this->pane->endPanel();  
echo $this->pane->endPane();
```



JDate

- `jimport('joomla.utilities.date');`
- Examples:
 - `$jdate = JDate() ; // current timestamp`
 - `$jdate = JDate($unix_timestamp);`
 - `$jdate = JDate('2008-12-31 00:00:00');`
 - `$mysqldate = $jdate->toMySQL();`
 - `$timestamp = $jdate->toUnix();`
 - `$custom = JFactory::getDate()->toFormat('%a %d %b %Y - %H:%M');`
 - `$custom = $jdate->toFormat(JText::_ ('DATE_FORMAT_LC'));`
 - `echo JHTML__('date', $date, JText::_ ('DATE_FORMAT_LC'));`
- Available formats
 - `DATE_FORMAT_LC` = Maandag, 31 december 2009
 - `DATE_FORMAT_LC2` = Maandag, 31 december 2009 18:15
 - `DATE_FORMAT_LC3` = 31 december 2009
 - `DATE_FORMAT_LC4` = 31.12.09
- Tip: Always save dates within MySQL as “datetime”



Using user parameters

Fetch a parameter

```
$user =& JFactory::getUser() ;  
$user->getParam( 'timezone', $registry->getValue( 'config.offset' ) ) ;
```

Set a parameter

```
$user =& JFactory::getUser() ;  
$params =& $user->getParameters();  
$params->setParam( 'foo', 'bar' );  
$user->set( 'params', $params->toString());  
$user->save();
```



Joomla! routing


- `JRoute::_($url, $xhtml = true, $ssl = 0)`
 - `JRoute::_('index.php?option=mycomponent&view=item&id=1');`
- `router.php`
 - `MyComponentBuildRoute(&$query = array()) {`
 - ...
 `return $segments ;`
 - `MyComponentParseRoute(&$segments = array()) {`
 - ...
 `return $vars ;`
- The Slug
 - `index.php?option=com_content&view=article&id=9:my-article&Itemid=5`
 - Built using the “alias” of a title



Using xdebug

Debugger and profiler tool for PHP (xdebug.org)

- Eclipse xdebug plugin
- PHP integration
 - Pre-compiled Zend extension
 - PECL extension (“pecl install xdebug”)
 - Manual compilation
- PHP code
 - xdebug_call_file() / xdebug_call_line() / xdebug_call_function()
 - xdebug_memory_usage()
 - xdebug_peak_memory_usage()
 - xdebug_time_index()
 - var_dump() # modified

 Fatal error: Maximum execution time of 1 second exceeded in /home/httpd/html/test/xdebug/docs/stack.php on line 34				
Call Stack				
#	Time	Memory	Function	Location
1	0.0001	58564	{main}()	../stack.php:0
2	0.0004	62764	foo()	../stack.php:47



Using AJAX in your component

Use MooTools to listen to an event

```
$('#a.ajax').addEvent('click', function() {  
    var myUrl = "index.php?option=com_x&format=raw&task=y" ;  
    var myAjax = new Ajax( myUrl, { onSuccess: function(r) {  
        $('#div.ajax').innerHTML(r);  
    }});  
    myAjax.request();  
});
```

The file “view.raw.php” outputs HTML/XML/JSON data

- Do not include your own PHP-script directly, but use Joomla! MVC
- Use MooTools to simplify your JavaScript
- If you're up to it, create XML through JSimpleXMLElement



Reserved variables in JModel

Variable	Description	Defined in
\$_name	The model name	JModel
\$_db	JDatabase instance	JModel
\$_state	State of the model	JModel
\$_errors	Errors	JObject