B. Using Parameters

Last Updated Wednesday, 14 September 2005

This appendix describes how to use the parameter system in your Joomla Modules and Components. Parameters provided with components relate to them when they are added to menu items. The parameters will show when you edit the menu item.

XML Definition

The Modules and Components setup files define the parameters. Parameters are wrapped in a <params> collection tag and then individual parameters are defined in a <param> tag. A typical parameter block may look like:

Â <params name="" description="">

Â Â <param name="count" type="text" default="5" label="Number of items"

Â Â Â Â Â Â Â description="The number of items to show" />

Â </params>

The following attributes are allowed for the <params> tag:

Â Â Â *

Â Â Â Â Â name: The name of the group of parameters parameter

Â Â Â *

Â Â Â Â Â description: A description for the group of parameters

Both these attributes are optional.

The following attributes are allowed for the <param> tag:

Â Â Â *

Â Â Â Â Â name: the name of the parameter

Â Â Â *

Â Â Â Â Â type: the type of form element

Â Â Â *

Â Â Â Â Â default: the default value for the parameter

Â Â Â *

Â Â Â Â Â label: The text to place to the left of the edit control

Â Â Â *

Â Â Â Â Â description: The text to place to the right of the edit control.

The available types for parameters are as follows:

text

The "text" type provides a simple text box, for example:

<param name="count" type="text" default="5" label="Number of items"

Â Â Â Â Â Â Â description="The number of items to show" />

list

The "list" type provides for an HTML select list. You provide the necessary HTML options as child elements of this tag in the same way as you would for an HTML select list, for example:

<param name="hide_author" type="list" default="" label="Hide Author"

Â Â Â description="Show/Hide the item author - only affects this page">

Â <option value="">Use Global</option>

Â <option value="1">Hide</option>

Â <option value="0">Show</option>

</param>

radio

The "radio" type provides for an HTML radio group. You provide the necessary HTML options as child elements of this tag in the same way as you would for an HTML select list, for example:

```
<param name="show_leading" type="radio" default="1" label="Show Leading"

Â Â Â  description="Show leading items">

Â  <option value="0">No</option>

Â  <option value="1">Yes</option>

</param>
```

Radio list groups are formatted horizontally.

mos_section

The "mos_section" type provides a list of published sections, for example:

```
<param name="section_id" type="mos_section" default="0" label="Section"

Â  description="A content section" />
```

The section id (primary key) is returned.

mos_category

The "mos_category" type provides a list of published categories, for example:

```
<param name="catid" type="mos_category" default="0" label="Category"

Â  description="A content cateogry" />
```

The lists displays items in "Section-Category" format. The cateogry id (primary key) is returned.

mos_menu

The "mos_menu" type provides a list of the defined menu types used in the site, for example:

```
<param name="menutype" type="mos_menu" default="mainmenu" label="Menu Type"

Â  description="The name of the menu (default mainmenu)" />
```

mos_imagelist

The "mos_imagelist" type provides a list of images in the directory defined by the "directory" attribute, for example:

```
<param name="menu_image" type="imagelist" directory="/images/M_images" default=""
Â  label="Menu Image"
Â  description="A small image to be placed to the left or right of your menu item..." />
```

The directory is relative to the installation directory of Joomla.

textarea

The "textarea" type provides a simple textarea, for example:

```
<param name="description_text" type="textarea" default="" label="Description Text" rows="30" cols="5
Â  description="Description for page, if left blank it will load _WEBLINKS_DESC from your language file"/>
```

Note that rows and cols attributes are available for this parameter type.

The mosParameters Class

A helper class is available for working with parameters called mosParameters.

The mosParameters class constructor takes two arguments, the first the textual parameter values retrieved from the database, and the second in the xml setup file where the parameters are defined. See the following example:

```
global $mainframe;

$option = 'com_content';

// get params definitions
$params =& new mosParameters( $menu->params,
Â  $mainframe->getPath( 'com_xml', $option ), 'component' );
```

To display the parameters pass the params variable to your display function and simply echo the text returned by the "render" method, for example:

```
<?php

function displayFoo( &$params ) {

Â  echo $params->render();

}

?>
```

Always pass the params variable by reference as this conserves memory.

Extending Parameters

NOTE: Experimental

It is possible to add your own form elements by extending the parameters class. For example, to add a form element that lists users you might do something like the following:

```
class myParameters extends mosParameters {

Â  function _form_userlist() {

Â Â Â  global $database;

Â

Â Â Â  $database->setQuery( "SELECT a.id AS value, a.name AS text"

Â Â Â Â Â  . "nFROM #__users AS a"

Â Â Â Â Â  . "nWHERE a.blocked='0'"

Â Â Â Â Â  . "nORDER BY a.name"

Â Â Â  );

Â Â Â  $options = $database->loadObjectList();

Â Â Â  array_unshift( $options, mosHTML::makeOption( '0', '- Select User -' ) );

Â Â Â  return mosHTML::selectList( $options, "params[$name]", "class="inputbox"",

Â Â Â Â Â  'value', 'text', $value );

Â  }

}
```

```
// get params definitions

$params =& new myParameters( $params_text, $xml_file );


echo $params->render();
```

Using Parameters on the Site

Parameters replaces the bit-wise masking technique used in site components and modules.

The mosParameters class has three methods that you will use to access parameters:

get('name' [, 'default')

This methods will return the value of a parameter if it exists or is
set, otherwise it returns the 'default' value (or an empty string).

set('name', 'value')

This method method sets the value of a parameter. It returns the value set.

def('name', 'default')

This method combines both get and set. It will check to see if the
parameter of 'name' exists. If it does it returns it. If it doesn't it
sets it to 'default' and returns that value.

Here are some examples:

```
// Parameters

$menu =& new mosMenu( $database );

$menu->load( $Itemid );

$params =& new mosParameters( $menu->params );
```

```
$header = $params->get( 'header' );
```

```
$count = $params->def( 'count', 10 );
```

```
$params->set( 'readon', 1 );
```

A good example of how to use the def method is when you want to use a global state by default, for example, a parameter may be defined like:

```
<param name="hide_author" type="list" default="" label="Hide Author"

Â Â Â  description="Show/Hide the item author - only affects this page">

Â  <option value="">Use Global</option>

Â  <option value="1">Hide</option>

Â  <option value="0">Show</option>

</param>
```

Notice that the "Use Global" option is an empty string. You may then have the following code in your module or component:

```
$hide_author = $params->def( 'hide_author', $mosConfig_hideauthor );
```

If the "hide_author" parameter is not defined, that is, is an empty string or not present at all, then the parameter will be set to the second argument, in this case the global setting for hiding author names, and that value will be returned by the method. If the parameter is not empty, that is, either "0" or "1" then the parameter will not be changed and the actual setting is returned by the method.

Quick Fix for Old Modules

Modules written for version Mambo 4.5 that use parameters will not work correctly in Mambo 4.5.1 or Joomla 1.0 or later.

Developers are encouraged to upgrade the method of using parameters as it is very simple to do so. However, to get things running quickly you may insert the following code near the head of you module, before the parameters are used:

```
$params = mosParseParams( $module->params );
```