

Hello World 2 - Getting personal

Last Updated Thursday, 08 December 2005

IntroductionIn our first tutorial we created a very simple component that just displayed information. In this tutorial we will show you how to add a toolbar, create a help file and also pass some dynamic data to your html file using patTemplate.

RequirementsYou need for this tutorial:

- Joomla 1.0 or greater
- The files from the Hello World 1 tutorial

Let's RollWe will be creating four new files in this tutorial:

toolbar.hello.php

This file represents the task

handler for the toolbar (like the task handler for the component). The Joomla Administrator looks for this file, "toolbar.component_name.php", when it first loads the component.

toolbar.hello.html.phpThis

file actually displays the toolbar for a given task. As for the

previous file, this file needs to the named

"toolbar.component_name.html.php".

tmpl/politehello.htmlThis is the html file for a new task that we will be adding.

help/helloworld.htmlThis is a plain html file with some sort of helpful message in it.

The Toolbar Event HandlerThe toolbar task handler, toolbar.hello.php, is very similar in structure to the event handler of the component:

```
<?php
/**
 * @version $Id: toolbar.languages.php,v 1.4 2005/01/06 01:13:18 eddieajau Exp $
 * @package Joomla
 * @subpackage Languages
 * @copyright Copyright (C) 2005 Open Source Matters. All rights reserved.
 * @license http://www.gnu.org/copyleft/gpl.html GNU/GPL, see LICENSE.php
 * Joomla! is free software and parts of it may contain or be derived from the
 * GNU General Public License or other free or open source software licenses.
 * See COPYRIGHT.php for copyright notices and details.
 */

/** ensure this file is being included by a parent file */
defined( '_VALID_MOS' ) or
    die( 'Direct Access to this location is not allowed.' );

// include support libraries
require_once( $mainframe->getPath( 'toolbar_html' ) );

// handle the task
$task = mosGetParam( $_REQUEST, 'task', '' );

switch ( $task ) {
    default:
        helloToolbar::helloWorld();
        break;
}
?>
```

From this you can deduce that "toolbar_html" includes the html support for the toolbar which is based on the helloToolbar class. At the moment we will have the same toolbar regardless of the task.

The Toolbar HTML HandlerThe file, toolbar.hello.html.php, actually does the work of displaying the toolbar:

```
<?php
/**
```

```

* @version 1.0
* @package HelloWorld
* @copyright Copyright (C) 2005 Open Source Matters. All rights reserved.
* @license http://www.gnu.org/copyleft/gpl.html GNU/GPL, see LICENSE.php
* Joomla! is free software and parts of it may contain or be derived from the
* GNU General Public License or other free or open source software licenses.
* See COPYRIGHT.php for copyright notices and details.
*/

/** ensure this file is being included by a parent file */
defined( '_VALID_MOS' ) or
    die( 'Direct Access to this location is not allowed.' );

/**
 * @package HelloWorld
 */
class helloToolbar {
    /**
     * Displays toolbar
     */
    function helloWorld(){
        mosMenuBar::startTable();
        mosMenuBar::apply( 'polite', 'Be Polite' );
        mosMenuBar::spacer();
        mosMenuBar::help( 'helloworld.html', true );
        mosMenuBar::endTable();
    }
}
?>

```

As for component task, we define a method in the toolbar class for each different toolbar layout we want to provide. So, in this example, we have our helloWorld method to display the toolbar. It is made up of a number of static methods in the mosMenuBar class (found in /administrator/includes/menubar.html.php).

You must start the toolbar by using the startTable method and finish it with the endTable method. What we have done is to use the standard Apply button but redefine the task it will call, "polite", and the text that displays beside it, "Be Polite". We then add a spacer and then a help button. When you click on the help button it will load the helloworld.html file; we will create this later. The second true argument indicates that the help file is found in the component's help directory. Adding the New EventUsing your code from Hello World 1, add the following case statement to the switch block in admin.hello.php:

```

case 'polite':
    politeHello();
    break;

```

Then add the following function to the end of the file:

```

/**
 * Polite hello event
 */
function politeHello() {
    global $my;

    helloScreens::politeHello( $my->username );
}

```

You can see that we must have set up a new screen method called politeHello. This method takes the name of the user to insert into the final display. We are making use of a global Joomla variable called \$my. This is an object that has information about you, the user logged in at the time. It has a property called username.

You may ask why we pass the name to the screen function? Why can't we just grab it there? Well, the reason is that this way, we are separating all of the data assembly operations from the presentation layer. The purpose of the helloScreens class is merely to insert the data into the html template. Similarly, the component, this politeHello function is not allowed to output any information or even add any html to the data, that's all to be done by the presentation layer. The principal

here is to keep your business logic out of the html, and your html out of the business logic. Preparing the TemplateUsing your code from Hello World 1, add this new method to the helloScreens class in admin.hello.html.php:

```
/**
 * A polite hello
 * @param string The name of a person
 */
function politeHello( $name ) {
    // import the body of the page
    $tmpl =& helloScreens::createTemplate();
    $tmpl->setAttribute( 'body', 'src', 'politehello.html' );

    $tmpl->addVar( 'body', 'name', $name );

    $tmpl->displayParsedTemplate( 'form' );
}
```

It's very similar to our previous example. The differences are that we are using the politehello.html file as the basis for output and we also add the users name as a variable to the html template. We do this by invoking the addVar method. It takes three arguments, the name of the template, the name of the variable in the template and then the value to insert in the variable. So, what we are saying is put the value of \$name into the name variable in the body template. Now, remember from the previous tutorial, that the body template has already been loaded and we are grafting in the contents of politehello.html.

The last thing to do is to display the "form" template.

A Well Mannered TemplateFirst, we have to fixed something up so that we can use the toolbar. Open helloworld.html from the previous tutorial and add the following lines to the end of the file:

```
<input type="hidden" name="option" value="{OPTION}" />
<input type="hidden" name="task" value="" />
```

The toolbar requires a hidden form element called task in order to operate. The other hidden element, option, tells Joomla to come back to this component when the form is submitted. Now, you ask, what is the {OPTION} thing. This is a template variable. In this case, the value for option has already been assigned in the template (in the same way we did for the name of the user above). All you have to do is place the name of the variable in uppercase and wrap it in curly braces.

Now we need to create our new html template file for our new screen. It's easiest to manage each screen in it's own html file (it gets too confusing otherwise). Create a file called politehello.html in your /tmpl directory with the following code::

```
<mos:comment>
@version 1.0
@package HelloWorld
@copyright (C) 2005 Andrew Eddie
@license http://www.gnu.org/copyleft/gpl.html GNU/GPL
</mos:comment>
```

```
<h1>Hello World</h1>
```

Welcome, {NAME}, to the Hello World tutorial.

```
<input type="hidden" name="option" value="{OPTION}" />
<input type="hidden" name="task" value="" />
```

You'll see that that we have a similar layout to what we had before except that we have place {NAME} where we want the name to be displayed. We finish off the file with the hidden form fields to ensure that the toolbar works properly.

A Bit More HelpThere's on last thing to do, and that's to create the screen help file. Create a file called helloworld.html and place it in a new directory called /help under your component directory (that is, com_hello). Insert the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang="en_US">
<head>
<title>Hello World</title>
<link href="../../../../help/css/help.css" rel="stylesheet" type="text/css" />
<meta name="copyright" content="(C) 2005 Andrew Eddie" />
<meta name="license" content="http://www.gnu.org/copyleft/gpl.html GNU/GPL" />
</head>
<body>
```

```
<h1>Hello World</h1>
```

Congratulations on completing the Hello World tutorials.

```
</body>
</html>
```

There's nothing special about this file, it's just regular html.

That's it. We're done. Save all these files, log into the Joomla Administrator and change the last portion of the URL to:

index2.php?option=com_hello

You will see the plain old hello message but also the new toolbar. Click on the Help button first. Your new help file should pop up and give you some meaningful message to inspire you. Next, click on the Be Polite button. Your screen should update with a message politely acknowledging you by name (well, user login name anyway).

You can download the files for the tutorial here ([helloworld_2.zip](#)).