Hello World 1 - The first steps

Last Updated Monday, 27 March 2006

IntroductionIf anyone has every picked up a book on programming, the first things they get you to do is complete a very simple exercise to write "Hello World" on the screen.Â This is usually a very exhilarating experience when you complete it because you know you are on the way to bigger and better things.

This tutorial aims to give you a grounding in the basic concepts for writing Joomla components.Â It will develop a very simple Hello World administrator component using patTemplate for the presentation layer.
RequirementsYou need for this tutorial:

- Joomla 1.0 or greater

Let's RollWe will be creating three files in this tutorial in a folder called /administrator/components/com_hello.Â Our component is called "hello" so the folder is given this name prefixed by "com_".Â Let's look at the files we need.

admin.hello.php
This file represents the main task handling file.Â The Joomla Administrator looks for this file, "admin.component_name.php", when it first loads the component.
admin.hello.html.phpThis file represents the preprocessor for final presentation.Â As for the previous file, this file needs to the named "admin.component_name.html.php".
tmpl/helloworld.htmlThis file represents that presentation layer, or the output, that will be displayed.Â There is no special naming convention for this file, although it is a good idea to have a name that closely relates to the task used to display it.The Event HandlerLet's look at our first file, the task handler, admin.hello.php.

```
<?php
/**
* @version 4.5.2
* @package HelloWorld
* @copyright (C) 2005 Andrew Eddie
* @license http://www.gnu.org/copyleft/gpl.html GNU/GPL
*/

/** 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( 'admin_html' ) );

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

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

Here is an outline of what this file is doing:

- The comment block at the top of the file defines meta data about the file, in particular the license and the copyright.Â This block has some special notations that are able to be parsed by an application called phpDocumentor.Â It is used to assemble API (Application Programmer Interface) documentation.Â The important thing here is to have a version and explicitly state the copyright and license terms for the file.
- Next we look for a constant called _VALID_MOS.Â Because this constant is defined elsewhere in Joomla, it ensures that only Joomla is trying to access this file.Â It prevents a user from opening the file explicitly in a browser.
- We then include the file that will support the presentation layer.Â $mainframe is a global variable in Joomla that has lots of useful methods attached to it.Â One of these methods is getPath.Â It helps you find common types of files.Â In this instance we want the admin.hello.html.php file to be include, so we pass 'admin_html' to the getPath method.
- Next we determine what task has been passed through the URL.Â To do this we use a utility function in Joomla called mosGetParam.Â It takes three arguments, a source array, a key in the array to search for and a default value to assign if that key is not found. Â In most instances we are not sure if the task is going to come through the URL or a form post, so we use $_REQUEST as the source array.
- We then pass the task to a switch control structure.Â We only have one thing to do in this example so it doesn't much matter if a task has been nominated or not.Â Whatever the case, we are going to call a static class method called helloScreens::helloWorld.

Preparing for OutputThe helloScreens class is defined in admin.hello.html.php so let's look at that file now:

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

```
* @version 4.5.2
* @package HelloWorld
* @copyright (C) 2005 Andrew Eddie
* @license http://www.gnu.org/copyleft/gpl.html GNU/GPL
*/

/** 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 helloScreens {
    /**
     * Static method to create the template object
     * @return patTemplate
     */
    function &createTemplate() {
        global $option, $mosConfig_absolute_path;
        require_once( $mosConfig_absolute_path
            . '/includes/patTemplate/patTemplate.php' );

        $tmpl =& patFactory::createTemplate( $option, true, false );
        $tmpl->setRoot( dirname( __FILE__ ) . '/tmpl' );

        return $tmpl;
    }
    /**
     * A simple message
     */
    function helloWorld() {
        // import the body of the page
        $tmpl =& helloScreens::createTemplate();
        $tmpl->setAttribute( 'body', 'src', 'helloworld.html' );

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

As you can see, the start of the file is the same as our previous file.Â It includes a single class called helloScreens.Â The first method of the class, createTemplate, is a standard format for creating the template object we will be using to generate output.Â This method includes the required patTemplate library file, creates a patTemplate object and then sets the root directory for template (html) files to the /tmpl directory in your component.

Now, there is a lot going on behind the scenes here.Â The template object has already loaded and parsed another template file called page.html which includes many standard wrappers and other things.Â One of the standard templates is called form and this one will help us with our display for our component.

The next method is helloWorld and you will recall invoking this method in the admin.hello.php file.Â After it creates the template object it sets an attribute in the preloaded template called form that I mentioned before.Â In the form template is a sub-template called body.Â This sub-template is where we graft in the output from our component.Â You don't have to understand how all this is happening yet.Â Just appreciate that the setAttribute method is telling the template object to set the source of the body template to your html file, helloworld.html.

The last thing to do is to display the "form" template.
The HTML Output FileSo now let's look at that html file:

```
 <mos:comment>
@version 4.5.2
@package HelloWorld
@copyright (C) 2005 Andrew Eddie
@license http://www.gnu.org/copyleft/gpl.html GNU/GPL
</mos:comment>

<h1>Hello World<h1>
```

There's not really much to this file at all.Â We have a comment block at the top similar to our php files above.Â In patTemplate, text enclosed in an xml comment tag is not displayed.Â The mos: is called the namespace and it distinguishes the xml tags for patTemplate from any other html or xml tags that may be in your html file.

After the comment block we can include any valid html we like.

That's all the preparation done.Â Save all these files, log into the Joomla AdminisratorÂ and change the last portion of the URL to:
index2.php?option=com_hello

You should see a message announcing your successful completion of this tutorial.

You can download the files for the tutorial here (helloworld_1.zip).