

Layouts

Overview

Note: Plugin views are an additional feature brought through HUBzero libraries. A standard, non-HUBzero Joomla! install will not have this capability.

The majority of plugins will not have view files. Occasionally, however, a plugin will return HTML and it is considered best practices to have a more MVC structure to your plugin and put all HTML and display code into view files. This allows for separation of the logic from presentation. There is a second advantage to this, however, which is that it will allow the presentation to be overridden easily by any Joomla! 1.5 template for optimal integration into any site.

Overriding plugin, module, and component presentation in templates is further explained in the [Templates: Overrides](#) section.

Directory Structure & Files

Plugins, like components and modules, are set up in a particular directory structure.

```
/plugins
  /groups
    forum.php    (the main plugin file)
    forum.xml    (the installation XML file)
  /forum
    /views
      /browse
        /tmpl
          default.php    (the layout)
          default.xml    (the layout installation XML file)
```

Similar to components, under the views directory of the plugin's self-titled directory (in the example, forum) there are directories for each view name. Within each view directory is a /tmpl/ directory. There is usually only one layout file but depending on who wrote the plugin, and how it is written, there could be more.

Implementation

Loading a plugin view

LAYOUTS

```
class plgExamplesTest extends JPlugin
{
    ...

    public function onReturnHtml()
    {
        // Include the HUBzero library that allows plugin views to work
        ximport('Hubzero_Plugin_View');

        // Instantiate a new view
        $view = new Hubzero_Plugin_View(
            array(
                'folder'=>'examples',
                'element'=>'test',
                'name'=>'display'
            )
        );

        // Set any data the view may need
        $view->hello = 'Hello, World';

        // Set any errors
        if ($this->getError()) {
            $view->setError( $this->getError() );
        }

        // Return the view
        return $view->loadTemplate();
    }
}
```

In the example, we're instantiating a new plugin view and passing it an array of variables that tell the object where to load the view HTML from. `folder` is the plugin group, `element` is the plugin, and `name` is the name of the view that is to be loaded. So, in this case, it would correspond to a view found here:

```
/plugins
  /examples
    /test
      /views
        /display
          /tmpl
```

LAYOUTS

```
default.php    (the layout)
default.xml    (the layout installation XML file)
```

Also note that we're returning `$view->loadTemplate()` rather than calling `$view->display()`. The `loadTemplate()` method captures the HTML output of the view rather than printing it out to the screen. This allows us to store the output in a variable and pass it around for later display.

The plugin view file

Our view (`default.php`) is constructed the same as any module or component view file:

```
<?php defined('_JEXEC') or die('Restricted access'); // no direct access ?>
<p>
  <?php echo $this->hello; ?>
</p>
```