

Output Overrides

Overview

There are many competing requirements for web designers ranging from accessibility to legislative to personal preferences. Rather than trying to over-parameterise views, or trying to aim for some sort of line of best fit, or worse, sticking its head in the sand, "Joomla!" has added the potential for the designer to take over control of virtually all of the output that is generated.

Except for files that are provided in the "Joomla!" distribution itself, these methods for customization eliminate the need for designers and developers to "hack" core files that could change when the site is updated to a new version. Because they are contained within the template, they can be deployed to the Web site without having to worry about changes being accidentally overwritten when your System Administrator upgrades the site.

While Joomla! only allows for overriding views and some HTML, HUBzero has extended this functionality to allow for overriding CSS as well. This allows for even more individualistic styling of components and modules on HUBs.

Component Overrides

Note: Not all HUBzero modules will have layouts or CSS that can be overridden.

Layouts

Layout overrides only work within the active template and are located under the `/html/` directory in the template. For example, the overrides for "corenil" are located under `/templates/corenil/html/`.

It is important to understand that if you create overrides in one template, they will not be available in other templates. For example, "rhuk_milkyway" has no component layout overrides at all. When you use this template you are seeing the raw output from all components. When you use the "Beez" template, almost every piece of component output is being controlled by the layout overrides in the template. "corenil" is in between having overrides for some components and only some views of those components.

The layout overrides must be placed in particular way. Using "Beez" as an example you will see the following structure:

```
/templates
  /beez
    /html
      /com_content (this directory matches the component directory name)
```

OUTPUT OVERRIDES

```
    /articles          (this directory matches the view directory name)
    default.php (this file matches the layout file name)
    form.php
```

The structure for component overrides is quite simple:
`/html/com_{ComponentName}/{ViewName}/{LayoutName}.php`.

Sub-Layouts

In some views you will see that some of the layouts have a group of files that start with the same name. The category view has an example of this. The blog layout actually has three parts: the main layout file `blog.php` and two sub-layout files, `blog_item.php` and `blog_links.php`. You can see where these sub-layouts are loaded in the `blog.php` file using the `loadTemplate` method, for example:

```
echo $this->loadTemplate('item');
// or
echo $this->loadTemplate('links');
```

When loading sub-layouts, the view already knows what layout you are in, so you don't have to provide the prefix (that is, you load just `'item'`, not `'blog_item'`).

What is important to note here is that it is possible to override just a sub-layout without copying the whole set of files. For example, if you were happy with the Joomla! default output for the blog layout, but just wanted to customize the item sub-layout, you could just copy:

```
/components/com_content/views/category/tmpl/blog_item.php
```

to:

```
/templates/rhuk_milkyway/html/com_content/category/blog_item.php
```

When Joomla! is parsing the view, it will automatically know to load `blog.php` from `com_content`

OUTPUT OVERRIDES

natively and `blog_item.php` from your template overrides.

Cascading Style Sheets

Over-riding CSS is a little more straight-forward over-riding layouts. Take the `com_groups` component for example:

```
/components
  /com_groups
    ...
    com_groups.css    (the component CSS file)
```

To override the CSS, we simply copy or create a new CSS file named the same and place it in the template's overrides:

```
/templates
  /corenil
    /html
      /com_groups    (this directory matches the component directory name)
        com_groups.css    (this file matches the CSS file name)
```

To push CSS from a component to the template, add the following somewhere in the component:

```
ximport('Hubzero_Document');
Hubzero_Document::addComponentStylesheet('com_example');
```

Module Overrides

Note: Not all HUBzero modules will have layouts or CSS that can be overridden.

Layouts

Modules, like components, are set up in a particular directory structure.

```
/modules
  /mod_latest_news
```

OUTPUT OVERRIDES

```
/tmpl
  default.php    (the layout)
  helper.php    (a helper file containing data logic)
  mod_latest_news.php  (the main module file)
  mod_latest_news.xml  (the installation XML file)
```

Similar to components, under the main module directory (in the example, `mod_latest_news`) there is a `/tmpl/` directory. There is usually only one layout file but depending on who wrote the module, and how it is written, there could be more.

As for components, the layout override for a module must be placed in particular way. Using "corenil" as an example again, you will see the following structure:

```
/templates
  /corenil
    /html
      /mod_latest_news  (this directory matches the module directory
name)
        default.php    (this file matches the layout file name)
```

Take care with overriding module layout because there are a number of different ways that modules can or have been designed so you need to treat each one individually.

Cascading Style Sheets

Over-riding CSS files works in precisely the same way as over-riding layouts. Take the `mod_reportproblems` module for example:

```
/modules
  /mod_reportproblems
    ...
    mod_reportproblems.css  (the module CSS file)
```

To override the CSS, we simply copy or create a new CSS file named the same and place it in the template's overrides:

```
/templates
```

OUTPUT OVERRIDES

```
/corenil
  /html
    /mod_reportproblems    (this directory matches the module directory name)
      mod_reportproblems.css (this file matches the CSS file name)
```

To push CSS from a module to the template, add the following somewhere in the module:

```
ximport('Hubzero_Document');
Hubzero_Document::addModuleStylesheet('mod_example');
```

Plugin Overrides

Note: Not all HUBzero plugins will have layouts or CSS that can be overridden.

Layouts

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.

As with components and modules, the layout override for a plugin must be placed in a particular way. Using "corenil" as an example again, you will see the following structure:

OUTPUT OVERRIDES

```
/templates
  /corenil
    /html
      /plg_groups_forum (this directory follows the naming pattern of
f plg_{group}_{plugin})
        /browse (this file matches the layout directory name)
          default.php (this file matches the layout file name)
```

Take care with overriding plugin layout because there are a number of different ways that plugins can or have been designed so you need to treat each one individually.

Cascading Style Sheets

Over-riding CSS files works in precisely the same way as over-riding layouts. Take the forum plugin for groups for example:

```
/plugins
  /groups
    /forum
      forum.css (the plugin CSS file)
```

To override the CSS, we simply copy or create a new CSS file named the same and place it in the template's overrides:

```
/templates
  /corenil
    /html
      /plg_groups_forum (this directory follows the naming pattern of
f plg_{group}_{plugin})
        forum.css (this file matches the CSS file name)
```

To push CSS from a module to the template, add the following somewhere in the module:

```
ximport('Hubzero_Document');
Hubzero_Document::addPluginStylesheet('groups', 'forum');
```

Pagination Links Overrides

This override can control the display of items-per-page and the pagination links that are used with lists of information. Most HUBzero templates will come with a pagination override that outputs what we feel is a good standard for displaying pagination links and controls. However, feel free to alter this as you see fit. The override can be found here:

```
/templates/{TemplateName}/html/pagination.php
```

When the pagination list is required, Joomla! will look for this file in the default templates. If it is found it will be loaded and the display functions it contains will be used. There are four functions that can be used:

pagination_list_footer

This function is responsible for showing the select list for the number of items to display per page.

pagination_list_render

This function is responsible for showing the list of page number links as well as the Start, End, Previous and Next links.

pagination_item_active

This function displays the links to other page numbers other than the "current" page.

pagination_item_inactive

This function displays the current page number, usually not hyperlinked.

Quick Reference

Using the corenil template as an example, here is a brief summary of the principles that have been discussed.

Note: Not all HUBzero components, plugins, and modules will have layouts that can be overridden.

Component Output

To override a component layout (for example the default layout in the article view), copy:

```
/components/com_content/views/article/tmpl/default.php
```

OUTPUT OVERRIDES

to:

```
/templates/corenil/html/com_content/article/default.php
```

To override a component CSS (for example the stylesheet in the com_groups), copy:

```
/components/com_groups/com_groups.css
```

to:

```
/templates/corenil/html/com_groups/com_groups.css
```

To push CSS from a component to the template, add the following somewhere in the component:

```
ximport('Hubzero_Document');  
Hubzero_Document::addComponentStylesheet('com_example');
```

Module Output

To override a module layout (for example the Latest News module), copy:

```
/modules/mod_latest_news/tmpl/default.php
```

to:

```
/templates/corenil/html/mod_latest_news/default.php
```


OUTPUT OVERRIDES

To override a module CSS (for example the stylesheet in the mod_reportproblems), copy:

```
/modules/mod_reportproblems/mod_reportproblems.css
```

to:

```
/templates/corenil/html/mod_reportproblems/mod_reportproblems.css
```

To push CSS from a module to the template, add the following somewhere in the module:

```
ximport('Hubzero_Document');  
Hubzero_Document::addModuleStylesheet('mod_example');
```

Plugin Output

To override a plugin layout (for example the Forum plugin for groups), copy:

```
/plugins/groups/forum/views/browse/tmpl/default.php
```

to:

```
/templates/corenil/html/plg_groups_forum/browse/default.php
```

To override a plugin CSS (for example the stylesheet for the forum plugin for groups), copy:

```
/plugins/groups/forum/forum.css
```

to:

OUTPUT OVERRIDES

/templates/corenil/html/plg_groups_forum/forum.css

To push CSS from a plugin to the template, add the following somewhere in the plugin:

```
ximport('Hubzero_Document');  
Hubzero_Document::addPluginStylesheet('groups', 'forum');
```

Customise the Pagination Links

To customize the way the items-per-page selector and pagination links display, edit the following file:

/templates/corenil/html/pagination.php