Chapter No. 6
"Joomla! 1.5 Template Reference"
In this package, you will find:
A Biography of the author of the book
A preview chapter from the book, Chapter NO.6 "Joomla! 1.5 Template Reference"
A synopsis of the book’s content
Information on where to buy this book

About the Author

Tessa Blakeley Silver's background is in print design and traditional illustration. She evolved over the years into web and multi-media development, where she focuses on usability and interface design.

Prior to starting her consulting and development company hyper3media (pronounced hyper-cube media: http://hyper3media.com), Tessa was the VP of Interactive Technologies at eHigherEducation, an online learning and technology company developing compelling multimedia simulations, interactions, and games, which met online educational requirements such as 508, AICC, and SCORM. She has also worked as a consultant and freelancer for J. Walter Thompson and The Diamond Trading Company (formerly known as DeBeers), and was a Design Specialist and Senior Associate for PricewaterhouseCoopers' East Region Marketing department.

Tessa has authored a few books fro Packt Publishing, including WordPress Theme Design.

I send a huge "thank you" to the Packt team who has made this title possible, and whose help in getting it out into the world has been invaluable. Special thanks to Mithun as well as Niko, Harry, and Siddharth, for their editing work. I'd also like to thank the large and wonderful Joomla! community and all who participate and power the open source world and strive to improve the accessibility of the Web for all. Additional thanks go out to my very patient family who spent quite a few evenings without me while I worked on this title.
Joomla! 1.5 Template Design

The goal of this title is to explain the basic steps of creating a Joomla! 1.5 template, then walk you through some more advanced techniques that can be used to enhance your template. This book's approach is different than my previous Joomla! 1.0 template book's approach ("Joomla! Template Design: Create your own professional-quality templates with this fast, friendly guide", Tessa Blakeley Silver, Packt Publishing). I've reviewed a lot of feedback on the previous 1.0 book, via email exchange, Packt's site, and book reviews. It's clear that I and my publisher were a bit off on our target readers' needs. I had attempted to write a book on creating Joomla! templates for someone who was not as familiar with standard web development techniques. This approach made for a book that was a little awkward to write, as it took me away from my own standard design and development process in order to focus on "Photoshop slice n' dice"/ WYSIWYG techniques that, of course, required me to spend more time explaining why those antiquated development methods don't work for dynamic Joomla! templates and the mundane basics of XHTML/CSS development.

I'd like to thank all of you in the Joomla! community who took the time to read the first book and email me with comments and post book reviews. This is your book. While it still discusses some helpful design approaches, tips and tricks, in Chapter 2 and Chapter 9, this book focuses squarely on the development, creation, and enhancement of Joomla! 1.5 templates and, therefore, does not cover general "how to" information about Joomla's basic features and capabilities and assumes you have some level of understanding about the basics of web development.

Joomla! 1.5 has excellent online documentation, which can be found at http://docs.joomla.org. There is also a large community of supporters who host all sorts of helpful articles and forums about Joomla. I do not try to replace or duplicate those sources or Joomla's documentation, but intend this book to be a companion to them.

My motive is to save you some time searching the Web, Joomla.org's extensive documentation, and various forum boards, trying to find relevant information on how to create and modify templates. This book should help you understand how Joomla! 1.5 templates work, and show you how to design and build a rich, in-depth web site interface on your own. Throughout the book, wherever applicable, I'll point you to the relevant Joomla! 1.5 documentation, along with many other useful on-line articles and sites.

I've attempted to create a realistic Joomla! template example that anyone can take the concepts from and apply to their own standard web site, while at the same time, show how flexible Joomla! 1.5 and its template capabilities are. I hope this book's case study example shows that Joomla! can be used to create truly unique and beautiful web sites.
What This Book Covers

Chapter 1 introduces you to the Joomla! CMS by making sure you know what you'll need to be aware of about the Joomla! 1.5 template project that you're ready to embark on. The chapter also covers the development tools that are recommended and the web skills you'll need to begin developing a Joomla! 1.5 template.

Chapter 2 takes a look at the essential elements you need to consider when planning your Joomla! template design. It discusses the best tools and processes for making your template design a reality. I explain my own rapid design comping technique and give you some tips and tricks for developing color schemes and graphic styles for your Joomla! template. By the end of the chapter, you'll have a working XHTML and CSS-based "comp" (or mockup) of your template design, ready to be coded up and assembled into a fully-functional Joomla! template.

You say you're not that creative? The upside of this process means that you don't have to design your own XHTML/CSS mockup to move on to Chapter 3. You can use any basic, static XHTML/CSS design (though you'll need to understand the XHTML and CSS used in it) and convert it into a Joomla! template using the methods explained in Chapter 3.

Chapter 3 uses our final XHTML and CSS mockup from Chapter 2 and shows you how to add Joomla's 1.5 jdoc tags to it. Along the way, this chapter covers the essentials of what makes a Joomla! template work. At the end of the chapter, you'll have a working Joomla! template.

Chapter 4 discusses the basic techniques of debugging and validation that you should be employing throughout your template's development. It covers the W3C's XHTML and CSS validation services and how to use the Firefox browser and some of its extensions as a development tool, not just another browser. This chapter also covers troubleshooting some of the most common reasons "good code goes bad", especially in IE 6 and IE7, and best practices for fixing those problems, giving you a great-looking template across all browsers and platforms. We'll also take a look at some SEO enhancements you can make to your site.

Chapter 5 discusses how to properly set up your Joomla! template's templateDetails.xml file and package up your files using the ZIP file format, so that they install into Joomla! correctly. It also discusses running some test installations of your template's package in the Joomla's Administration panel so you can share your Joomla! template with the world.

For More Information:
Chapter 6 covers key information about easy-to-look-up headers that will help you with your Joomla! template development—from the many CSS id and class styles that Joomla! itself outputs, to Joomla's jdoc tags and their controlling attributes. It also covers key 1.0 to 1.5 update information for those of you trying to update a Joomla! 1.0 template to a native 1.5 template. The information in this chapter is listed along with key links to bookmark, to make your template development as easy as possible.

Chapter 7 dives into taking your working, debugged, validated, and properly packaged Joomla! template from Chapters 3, 4, and 5, and starting to enhance it with a dynamic layout that has collapsible columns, using Joomla's PHP code. We'll then spruce up our main menu using the Suckerfish CSS-based method and Adobe Flash media.

Chapter 8 continues showing you how to enhance your Joomla! template, by taking a look at the most popular methods for leveraging AJAX techniques in Joomla! Using extensions. I'll also give you a complete background on AJAX and when it's best to use those techniques or skip them. The chapter also reviews some cool JavaScript toolkits, libraries, and scripts that you can use to simply make your Joomla! Template appear "Ajaxy".

Chapter 9 covers how to enhance your template by adding a params.ini file, so that your template's users have more dynamic control over your template's configuration right from the Administration panel. For you PHP gurus out there, we'll also cover the basics you need to know to get started with creating your own "module chrome" and "template override" files. We'll also review the main tips from the previous chapters and cover some key tips for easily implementing today's coolest CSS2 and CSS3 tricks into your template. Finally, we'll go over a few final PHP tips to help you enhance your template's usability, so that it can handle right-to-left languages.
Now that you've had some hands-on experience with making templates, you've probably noticed that Joomla! outputs quite a bit of CSS class and id rules. Even if you intend to control much of that output with module chrome and template overrides, it's helpful to know where to look for and how to set up those overrides.

You can always use your DOM Source Inspector to see what's wrapped around markup that you're currently working with, but clearly, it will be helpful to have a heads-up on what to look for within the DOM as well.

We'll use this chapter to go over jdoc tags for templates, the standard CSS class and id rules that Joomla! outputs, how module and template override files are organized, as well as useful Joomla! PHP code you can use in your template to aid in making it more user-friendly and dynamic. Of course, wherever possible, I'll let you know the relevant Joomla! documentation links to bookmark, to give you in-depth detail and save you a little time searching through the Joomla! documentation site and the Web.

Last, take note that I'll mention how these Joomla! 1.5 items differ in use from a Joomla! 1.0 template, so that those of you looking to update a Joomla! 1.0 template to 1.5 can quickly get a handle on what to update in your templates and what new features to add.

Consider this chapter your "cheat sheet".

**Jdoc include tags**

I've mentioned a few times that the jdoc include tags are new to Joomla! 1.5 templates. Previously in Joomla! 1.0, more complicated, abstract PHP code, originally created for Mambo, was used. The jdoc tags are much cleaner, visually make sense (no more guessing what attribute values like "-3" mean), and, thus, are much easier to remember.
Site header information tag
This is pretty simple: the tag outputs all the appropriate meta tags and header
information that corresponds to your site and each individual page:

```xml
<jdoc:include type="head" />
```

Joomla! 1.0 to 1.5 conversion
If you're converting a 1.0 template to 1.5, you'll replace this PHP function in your
1.0 template's header with the above `jdoc` tag:

```php
<?php mosShowHead(); ?>
```

The component include tag
Wherever you place this include, all component content will appear (from articles to
poll results to contact forms, and so on):

```xml
<jdoc:include type="component" />
```

Joomla! 1.0 to 1.5 conversion
The 1.0 equivalent of this tag is the `mosMainBody` function. You'll replace this PHP
function with the above `jdoc include`:

```php
<?php mosMainBody(); ?>
```

Module position tags
With module tags, we have a few options to work with. So, we can control what
modules load into the area, thus assigning their positions as well as what style to
output the module content with:

```xml
<jdoc:include type="modules" name="position" style="styleName" />
```
Module position styles

In the jdoc include example above, within the style attribute, you can place one of the following six style names to various effect:

<table>
<thead>
<tr>
<th>Style name</th>
<th>Effect</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>none or raw</td>
<td>Modules are displayed in plain text, without titles.</td>
<td>Content</td>
</tr>
<tr>
<td>xhtml</td>
<td>Modules are displayed wrapped in a single &lt;div&gt; tag, with titles in &lt;h3&gt; header tags. (This is preferred for most applications of Joomla.)</td>
<td>&lt;div class=&quot;moduletable&quot;&gt; &lt;h3&gt;Title&lt;/h3&gt; Content &lt;/div&gt;</td>
</tr>
<tr>
<td>rounded</td>
<td>Modules are displayed wrapped in several &lt;div&gt; tags with titles in &lt;h3&gt; header tags, allowing for more complex CSS styling, such as the container techniques discussed in detail in Chapter 4 or applying stretchable, rounded corners.</td>
<td>&lt;div class=&quot;module&quot;&gt; &lt;div&gt; &lt;div&gt; &lt;div&gt; &lt;h3&gt;Title&lt;/h3&gt; Content &lt;/div&gt; &lt;/div&gt; &lt;/div&gt;</td>
</tr>
<tr>
<td>table</td>
<td>Modules are displayed in a table with a single row column. This is also the default setting. You'll never really need to use it.</td>
<td>&lt;table class=&quot;moduletable&quot; cellpadding=&quot;0&quot; cellspacing=&quot;0&quot;&gt; &lt;tbody&gt; &lt;tr&gt; &lt;th&gt;Title&lt;/th&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;Content&lt;/td&gt; &lt;/tr&gt; &lt;/tbody&gt; &lt;/table&gt;</td>
</tr>
<tr>
<td>Style name</td>
<td>Effect</td>
<td>Sample</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| horiz      | Modules are again displayed in a table with multiple column rows, giving it the effect of being displayed horizontally, rather than vertically like the default. | `<table>  
|            |        | `<tbody>`  
|            |        | `<tr>`  
|            |        | `<td align="top">`  
|            |        | `<table cellpadding="0" cellspacing="0" class="moduletable">`  
|            |        | `<tbody>`  
|            |        | `<th valign="top">Title</th>`  
|            |        | `</tr>`  
|            |        | `<tr>`  
|            |        | `<td>Content</td>`  
|            |        | `</tr>`  
|            |        | `</tbody>`  
|            |        | `</table>`  
|            |        | `<td>  
|            |        | `<!--next table cell starts-->`  
|            |        | `<td align="top">`  
|            |        | `<table cellpadding="0" cellspacing="0" class="moduletable">`  
|            |        | `<tbody>`  
|            |        | `<th valign="top">Title</th>`  
|            |        | `</tr>`  
|            |        | `<tr>`  
|            |        | `<td>Content</td>`  
|            |        | `</tr>`  
|            |        | `</tbody>`  
|            |        | `</table>`  
|            |        | `</td>`  
|            |        | `</td>`  

| outline    | This is used to preview modules and help aid in module position names. | In your Administration panel, go to Extensions | Template Manager | yourTemplate | Preview for a sample of this output. |
Joomla! 1.0 to 1.5

For those of you trying to update a 1.0 template, replace your old `mosLoadModules` tag:

```php
<?php mosLoadModules ( 'modName', styleNumber); ?>
```

With the new `jdoc include`:

```xml
<jdoc:include type="modules" name="positionName" style="styleName" />
```

Where `modName` is located in your `mosLoadModule` tag. Be sure to replace it with the module `positionName` in your `jdoc include` tag, and where your `styleNumber` was in your `mosLoadModule` tag, replace it with the corresponding `styleName` in your `jdoc include` tag. The following will help you match up and select the appropriate style name:

- `table = 0` (this still is the default)
- `horiz = 1`
- `none (or raw) = -1`
- `xhtml = -2`
- `rounded = -3`

**Menu output options**

Very similar to 1.0, Joomla! 1.5 does need a little special attention to menu output. Yes, menus are modules, and yes, even though you set your module position to output `rounded` or `xhtml`, you might want to log in to your admin panel and make sure that your menus are outputting the XHTML markup you desire.

The default for Joomla! 1.5 is now **List**, which will also display nested lists for multi-level menu items (the **Extend** extension is no longer needed). However, if you're upgrading a 1.0 template and need the menu to output in a horizontal or vertical table, you'll need to assign the correct XHTML output in the menu's Module Manager.

Go to **Extensions | Module Manager** and select the relevant menu item.
On the far right, select the appropriate **Menu Style** from the drop-down list. Your choices are **List, Legacy - Vertical, Legacy - Horizontal, and Legacy - Flat List** (note: this won't include nested lists; not recommended). The following screenshot illustrates selecting the menu style.

- **List**: This will output your menus in clean WC3-compliant lists and nest any submenu items a parent or top-level or root menu item may have. This is highly recommended, as it allows your menus to be easily (and beautifully) styled with CSS, degrade gracefully in older browsers and yet still appear functional in older, text-based browsers. (Many mobile-and disability-focused browsers are text-based and strip out CSS styles.)

- **Legacy - Vertical**: This option will display the top-level menu items only in a table format that stacks vertically. You'll probably select this only if you're using an older 1.0 that you've upgraded to 1.5 or installed in Joomla! 1.5's "legacy mode".

For More Information:  
• **Legacy - Horizontal**: This option will display the top-level menu items only in a table that extends out horizontally. You'll probably select this only if you're using an older 1.0 that you've upgraded to 1.5 or installed in Joomla! 1.5's legacy mode.

• **Legacy - Flat List**: This option does display in a W3C-compliant list mode, yet only displays the top-level menu items. Again, you'll probably select this only if you're using an older 1.0 that you've upgraded to 1.5 or installed in Joomla! 1.5's legacy mode.

### Using overrides

As mentioned in *Chapter 3*, template overrides are specialized files the Joomla! system checks for in your template's `html` folder. Template overrides include module chrome and **component overrides**. If Joomla! discovers that a particular file exists, for a specific module or component, in your template's `html` folder, Joomla! will reference that file's output, rather than its core output. The most common approach is to use the Beez template overrides, if you'd like to have truly accessible, table-less XTHML output from Joomla!. You can also create your own files.

Of course, the Joomla! system is not "psychic"! You can't just create an override and place it anywhere in your template directory, named anything you want, and hope Joomla! figures out your intentions. There is a specific folder and order placement you have to follow in order for Joomla! to understand that your file is intended to override specific core output. We'll take a closer look at this next.

### Module overrides and chrome

In the earlier *jdoc* section, when we looked at module tags and layout control, we noted that you can specify `rounded`, `xhtml`, `horiz`, and `vert` styles. You can also easily set up additional chrome by editing or creating a file within the `html` directory, inside the appropriate `mod_moduleName` folder.
Inside the `html` directory, the following files are available to set up your own module chrome (see the next screenshot): `modules.php`, `mod_search/default.php`, `mod_poll/default.php`, `mod_login/default.php`, `mod_latestnews/default.php`, `mod_newsflash/_item.php`, `default.php`, `horiz.php`, and `vert.php`.

Keep in mind, while the stated folder structure is essential to follow, and each folder should have a `default.php` file in it, you can also create additional helper or view files if needed to support that layout. Notice that the Beez template `mod_newsflash` override takes advantage of this. Again, the purpose of overrides is to separate out the Joomla! CMS core from your site's presentation. This means you can change how your content output from the components and modules are displayed, but any updates to Joomla's core system will not require you to change your files (or it shouldn't; I can't guarantee the Joomla! development team will never produce a core system change that stops working with the API hooks used in the template overrides or module chrome, but that is the goal). This aspect gets a bit beyond the scope of this title, but if you'd like to know more, please reference the Joomla! documentation site: [http://docs.joomla.org/Understanding_Output Overrides](http://docs.joomla.org/Understanding_Output Overrides).
How module chrome works

If you open up the Beez module.php file, you'll notice (under quite a bit of useful comments) some PHP code that looks like this (starting around line 24):

```php
function modChrome_beezDivision($module, &routeParams, &attribs)
{
    $headerLevel = isset($attribs['headerLevel']) ? (int)
    $attribs['headerLevel'] : 3;
    if (!empty ($module->content)) : ?
        <div class="moduletable<?php echo $params->get('moduleclass_sfx'); ?>">
            <?php if ($module->showtitle) : ?
                <h<?php echo $headerLevel; ?>><?php echo $module->title; ?></h<?php echo $headerLevel; ?>
            <?php endif; ?>
            <?php echo $module->content; ?>
        </div>
    <?php endif; }

You would reference this chrome override by calling the second half of the function name, after modChrome: modChrome_beezDivision. You would then reference beezDivision in your jdoc module position style tag in the following fashion:

```
<jdoc:include type="modules" name="user1" style="beezDivision" />
```

Your template will now call in this particular chrome override, as opposed to the standard xhtml style previously called in. Unfortunately, for display purposes, this chrome override is not really that much different from the xhtml chrome style.

That's OK, if we change the last value of the $headerLevel variable from 3 to 2 (about line 26 in the module.php file) like so:

```php
$headerLevel = isset($attribs['headerLevel']) ? (int)
    $attribs['headerLevel'] : 2;
```
We can quickly see the end result in our template’s user1 module position, as the header is now an h2 instead of an h3 tag (see the following screenshot):

From this example, you can see how to expand and start creating your own module chrome. You can, basically, copy the above beezDivision function, paste it underneath the existing function, and tweak the $headerLevel variable as well as any XHTML markup you see surrounding the Joomla! PHP code.

To really construct module chrome from scratch, I would recommend you be comfortable with PHP and understand the various variable and pJoomla! parameter you can use. You can find out more about these by referencing this page in the Joomla! documentation: http://docs.joomla.org/Applying_custom_module_chrome.

Don't confuse module chrome with module overrides! You'll note in the previous screenshot, showing the html directory with its module override directories, that in addition to the modules.php page, there are module overrides for specific module types. You can tweak, adjust, or write up these views from scratch, in addition to the chrome. The chrome that we just tweaked above is for general output, and most module views that you tweak will also end up being pulled, so to speak, through the chrome. For example, you may tweak the poll's module view (remember, the poll also has a component view) to be in an ordered list instead of an unordered list. That view, with the ordered list, will then also get pulled through the special chrome you set up, wrapping it in the header and div tags you specified.

Component overrides
Similar to module chrome and overrides, component overrides require that you copy in (or create from scratch) files inside specially named directories inside your html directory. The following screenshot displays these component override directories and files:
Also similar to module view overrides, component overrides just simply need to be there. If they're not there, Joomla! automatically reverts to its core output. In Chapter 9, we'll take a look at tweaking a component override for the article pages output. And once again, similar to module view overrides and module chrome, really whipping up component overrides from scratch requires a deeper understanding of Joomla! 1.5 and PHP than a book for creating templates aimed primarily at web designers can really get into.

After taking a look at our modest but very useful tweak in Chapter 9, if you feel your eye for PHP syntax is pretty good and you'd like to try your hand at more component overrides, be sure to check out that Understanding Output Overrides in the Joomla! documentation: http://docs.joomla.org/Understanding_Output_Oversrides.
Pagination

You can also override and tweak your pagination layout with a file called pagination.php inside the html folder. Be aware, this file is entirely PHP. You can look through it and easily see the XHTML markup to tweak, but be careful! The XHTML markup is being built up and outputted inside the $html variable in PHP. If you accidentally delete or overwrite any of the PHP syntax surrounding the XHTML, you'll break this file. Having a good eye for PHP syntax is a must for tweaking this file.

Additional template information

OK, we've already been chatting about how useful it is to, at the very least, have that eye for PHP syntax. It's, of course, even more useful to have a little PHP under your belt. I'm going to quickly stray even further into PHP development, and then come right back to templating and Joomla! basics. (Promise!) Even if you have no interest whatsoever in PHP development, bear with me, this little bit of background information on Joomla! 1.5 may help you better understand controlling your template.

Joomla! 1.5 was rebuilt using object-oriented programming, also affectionately called as OOP. One advantage of OOP is that you can use design patterns to aid in development. Joomla! 1.5 heavily relies on a design pattern called the Model View Controller or MVC pattern. The MVC design pattern ensures that separate files containing specific PHP code are used to tell the content management system (the CMS) three main things. Mainly: what its purpose and core function is (the Model), how to control visual display (aka the View), and how to do specific things such as update, delete, and edit content, and set CMS administration preferences, and so on (the Control).

Now, what the heck does this mean to you as a template designer? To start, as we saw above, we're no longer constrained to that core Joomla! table output. The "View" of MVC is indeed separate from the rest of the system, and thus, I don't have to go through all sorts of advanced discussions about how to hack tables out of the Joomla! system in a way that makes your CMS vulnerable to being incompatible with updates and has nothing to do with your template.

The next feature is that within this OOP, MVC framework, your whole template can be referenced as an 'object', and that simply means it can be referenced in your index.php file with PHP code using the $this->propertyName variable. We've already taken advantage of it several times in our template, mostly to help target our template directory using the baseurl property name. That is:

```php
<link rel="stylesheet" href="<?php echo $this->baseurl; ?>/templates/go_green/css/template.css" type="text/css" />
```
The following are additional template properties you might find useful in dynamically enhancing your template:

```php
<?php echo $this->base; ?> // outputs the full http:// path not just the url name
<?php echo $this->_file; ?> // outputs the server path to the file called in (not the same as the url path!) i.e.: /~user/httpdocs/1.5dev/templates/go_green/index.php
<?php echo $this->title; ?> // outputs the Article Title
<?php echo $this->description; ?> // outputs the Article Description
<?php echo $this->template; ?> // outputs the template's Name
```

For a more complete listing of what's available in your template's object array, check out the Joomla! documentation: [http://docs.joomla.org/Objects%2C_methods_and_properties_available_from_your_template](http://docs.joomla.org/Objects%2C_methods_and_properties_available_from_your_template).

You'll also find the countModules method useful for helping you set up dynamic layouts for collapsible columns. We'll cover how to use this method in detail in Chapter 7.

```php
$this->countModules('positionName');
```

You'd replace positionName with the name of the module position you want to count the modules in; that is, right, user1, left, footer, and so on.

For more information on the countModules method, check out these links in the Joomla! documentation:

- [http://docs.joomla.org/JDocumentHTML/countModules](http://docs.joomla.org/JDocumentHTML/countModules)
- [http://docs.joomla.org/Operators_for_use_with_the_countModules_function](http://docs.joomla.org/Operators_for_use_with_the_countModules_function)

**Common Joomla! CSS**

As you can see, via template overrides, you can pretty much define any CSS ids or classes you want. For those of you who are into creating and tweaking template overrides, unless you're going to create a highly custom, private, not-for-the-public template, my recommendation is you continue to use Joomla's general CSS ids and classes for component and module output as much as possible.
This is a good way to ensure your template is familiar to other Joomla! administrators, especially if you want to offer your template to the public or for commercial sale. It's easy for them to look up and customize CSS rules rather than forcing them to discover all the new and interestingly-named CSS ids and classes you created. For those of us working with Joomla's core output or the Beez template overrides (which attempts to use Joomla's standard CSS), here is a list of some of the most common CSS ids and classes. Those of you familiar with Joomla! 1.0 template design will be pleased to find these haven't really changed.

This list has been put together after a bit of research and a lot of experimentation with the Web Developer Toolbar CSS tools. It is probably not complete, but if you account for these items in your CSS rules, you'll be pretty well covered for most Joomla! projects, and it will be easy to spot any ids or classes not covered here and add them to your CSS sheet.

The Joomla.org forum has a post with a fairly comprehensive list, most of which you'll recognize here (although it does have some items on it that I don't seem to pick up in my template with the Beez overrides), so it's definitely worth checking out: http://forum.joomla.org/viewtopic.php?t=125508.

**Joomla! 1.5 CSS ids**

<table>
<thead>
<tr>
<th>CSS ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#active_menu</td>
<td>This is generated by the type=&quot;modules&quot; include. Use it to style and control the currently selected main menu item.</td>
</tr>
<tr>
<td>#blockrandom</td>
<td>This is generated by the type=&quot;component&quot; include when you're using the wrapper. This is the iFrame's id.</td>
</tr>
<tr>
<td>#contact_email_copy</td>
<td>This is generated by the type=&quot;component&quot; include when you're in the contact form page view. This is a field name id.</td>
</tr>
<tr>
<td>#contact_text</td>
<td>This is generated by the type=&quot;component&quot; include when you're in the contact form page view. This is a field name id.</td>
</tr>
<tr>
<td>#emailForm</td>
<td>This is generated by the type=&quot;component&quot; include when you're in the contact form page view. This is a field name id.</td>
</tr>
<tr>
<td>#mainlevel</td>
<td>This is generated by the type=&quot;modules&quot; include. Use it to style and control the main menu div holding each main menu item.</td>
</tr>
<tr>
<td>#mod_login_password</td>
<td>This is generated by the type=&quot;modules&quot; include. This is a field name id.</td>
</tr>
<tr>
<td>#mod_login_remember</td>
<td>This is generated by the type=&quot;modules&quot; include. This is a field name id.</td>
</tr>
</tbody>
</table>
#mod_login_username  This is generated by the type="modules" include. This is a field name id.
#poll  This is generated by the type="modules" include by the poll module. You can control the placement of the entire id with this.
#search_ordering  This is generated by the type="component" include when you're in the search form page view. This is a field name id.
#search_searchword  This is generated by the type="component" include when you're in the search form page view. This is a field name id.
#searchphraseall  This is generated by the type="component" include when you're in the search form page view. This is a field name id.
#searchphraseany  This is generated by the type="component" include when you're in the search form page view. This is a field name id.
#searchphraseexact  This is generated by the type="component" include when you're in the search form page view. This is a field name id.
#voteid1,#voteid2,#voteid3, and so on  This is generated by the type="modules" include. This is generated by the poll module and are field name ids for the radio buttons.

---

**Joomla! 1.5 CSS classes**

.article_separator  This is generated by the type="component" include. You can style the space/separations between articles in the blog or news flash views.
.back_button  This is generated by the type="component" include code. It's used to style the main back button, which is similar to hitting the back button in your browser.
.blog  This is generated by the type="component" include if you're in blog view.
.blog_more  This is generated by the type="component" include if you're in blog view. It indicates there are more blog stories in the links below.
.blogsection  This is generated by the type="component" include if you're in blog view. It formats additional blog links.
.button  This is generated by the type="modules" include. Use it to consistently style and control buttons generated by any of the modules.
### Joomla! 1.5 Template Reference

<table>
<thead>
<tr>
<th>CSS Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.buttonheading</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> if you're in blog view. Use this to control the layout and style of the PDF, email, and print controls.</td>
</tr>
<tr>
<td>.category</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> code if you're in blog view. Use it to control the layout and style of links to categories such as &quot;Latest News&quot; or &quot;Popular&quot; or &quot;Most Read&quot;.</td>
</tr>
<tr>
<td>.componentheading</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> if you're in latest news or blog view.</td>
</tr>
<tr>
<td>.contact_email</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> code when you're in the contact form page view. Use it to control the overall placement and style of all the contact form elements.</td>
</tr>
<tr>
<td>.content_rating</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. Style the ratings output of the content that has been voted on.</td>
</tr>
<tr>
<td>.content_vote</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. Style the link or button that allows the user to vote on the content.</td>
</tr>
<tr>
<td>.contentdescription</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. Style the descriptions of the content that can be voted on.</td>
</tr>
<tr>
<td>.contentheading</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code>. Use it to style the titles of articles and headings.</td>
</tr>
<tr>
<td>.contentpaneopen</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. It indicates the start of the content.</td>
</tr>
<tr>
<td>.contenttoc</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> code. Use it to style the TOC listings some content may generate.</td>
</tr>
<tr>
<td>.createdate</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. It controls the style of the displayed creation date of an article or a blog entry.</td>
</tr>
<tr>
<td>.fase4rdf</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code>. It's part of a great type of dynamic formatting class offered and lets you style the news RSS feeds you can set up through Joomla.</td>
</tr>
<tr>
<td>.frontpageheader</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code>. If you're using the home page module, style the front page headers with this class.</td>
</tr>
<tr>
<td>.inputbox</td>
<td>This is generated by the <code>type=&quot;component&quot; include</code> as well as the <code>type=&quot;modules&quot; include</code>. Use this to consistently style and control all form fields generated by the <code>mosMainBody</code> or a module.</td>
</tr>
</tbody>
</table>
.latestnews This is generated by the type="modules" code. The class is wrapped around a list of latest news links, which you can control with additional rule calls; that is, .latestnews td or .latestnews li, depending on the output options you've chosen.

.mainlevel This is generated by the type="modules" include. It lets you style and control main menu items displayed in the #mainlevel id.

.modifydate This is generated by the type="component" include. It accompanies date information if an article has been modified.

.module This class is generated by the type="modules" include when using the "rounded" style option.

.moduletable This class is generated by the type="modules" include when using the table, horiz, none, or xhtml style options.

.mosimage This is generated by the type="component" include. Use it to control and style images placed with articles.

.mosimage_caption This is generated by the type="component" include. Use it to control and style image captions placed with articles.

.mostread This is generated by the type="modules" code. It is similar to .latestnews. The class is wrapped around a list of latest news links, which you can control with additional rule calls; that is, .latestnews td or .latestnews li, depending on the output options you've chosen.

.newsfeed This is generated by the type="component" include in the News Feeds view. Use it to control and style the overall news feed display.

.newsfeeddate This is generated by the type="component" include in the News Feeds view. Use it to control and style the news feed displayed dates.

.newsfeedheading This is generated by the type="component" include in the News Feeds view. Use it to control and style the news feed headers.

.pagenav This is generated by the type="component" include. Use it to control and style the overall placement of next and previous page navigation.

.pagenav_next This is generated by the type="component" include. Use it to control and style the next page button.

.pagenav_prev This is generated by the type="component" include. Use it to control and style the previous page button.

.pagenavbar This is generated by the type="component" include. Use it to control and style the overall placement of next and previous page navigation.

.pagenavcounter This is generated by the type="component" include. Use it to control and style the overall placement of the page counter under the navigation.
### Joomla! 1.5 Template Reference

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.pathway</td>
<td>This class is generated by the <code>mospathway();</code> include.</td>
</tr>
<tr>
<td>.polls</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the poll module, and you can use it to set alternating backgrounds for your poll-select items.</td>
</tr>
<tr>
<td>.pollsborder</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the poll module, and you can use it to style the outside border of the module. Not to be confused with the <code>.pollstableborder</code> class!</td>
</tr>
<tr>
<td>.pollstableborder</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the poll module, and you can use it to style the border of the table generated by the module.</td>
</tr>
<tr>
<td>.readon</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include as well as the <code>type=&quot;modules&quot;</code> include. Use this to consistently style and control all the &quot;Read More&quot; links for truncated news, news flashes, and blog items.</td>
</tr>
<tr>
<td>.search</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the search module, and you can use it to control and style the main search field.</td>
</tr>
<tr>
<td>.sectionentry1</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the poll module, and you can use it to set alternating backgrounds for your poll-select items.</td>
</tr>
<tr>
<td>.sectionentry2</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. This is generated by the poll module, and you can use it to set alternating backgrounds for your poll-select items.</td>
</tr>
<tr>
<td>.sectionheader</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include as well as the <code>type=&quot;modules&quot;</code> include. You can use it to control section header titles displayed by modules and content.</td>
</tr>
<tr>
<td>.small</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include as well as the <code>type=&quot;modules&quot;</code> include. It's used to denote author names and other data related to an article or blog post.</td>
</tr>
<tr>
<td>.smalldark</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include as well as the <code>type=&quot;modules&quot;</code> include.</td>
</tr>
<tr>
<td>.sublevel</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include as well as the <code>type=&quot;modules&quot;</code> include. It's also used to denote sub items of navigation.</td>
</tr>
<tr>
<td>.syndicate</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. Use this to style the syndicate button layout or borders of your syndicate module.</td>
</tr>
<tr>
<td>.syndicate_text</td>
<td>This is generated by the <code>type=&quot;modules&quot;</code> include. Use this to style the syndicate layout if you're using text instead of buttons.</td>
</tr>
<tr>
<td>.text_area</td>
<td>This is generated by the <code>type=&quot;component&quot;</code> include. Use it to control and style the text areas of forms much like the <code>.inputbox</code> class.</td>
</tr>
</tbody>
</table>
Joomla! 1.0 to 1.5 conversion

The fact that Joomla! has left its core CSS ids and classes alone is a great aid in helping you update 1.0 templates to 1.5. Once you’ve changed your legacy mos... tags to the new, appropriate jdoc include tags, you should find that your CSS is able to style just fine. Of course, if you went from a table-based layout to a table-less layout with overrides, you’ll need to tweak your CSS to affect divs with those same CSS rules instead of tables, which can be a little different, but the general bulk of the work shouldn’t be too bad.

Template parameters

Joomla! 1.5 allows you to pass parameters from the Administration panel to your template, giving your Joomla! administrator additional control over the template in some very useful ways. We’ll cover how to set up a simple parameter function in detail in Chapter 9. For your reference, the essentials you need to know are covered in the next few sections.

Define a parameter in the templateDetails.xml file

You’ll want to place your parameter definitions within <param> tags inside the <params>... </params> tags at the bottom of the templateDetails.xml file before the closing </install> tag. For example:

```xml
<install>
...
<params>
  <param name="logoType" type="list" default="image" label="Logo type" description="Type of Logo">
    <option value="graphicHead">Graphic</option>
    <option value="textHead">Text</option>
  </param>
  <param name="logoText" type="text" default="" size="50" label="Logo Text" description="Your Logo Text" />
  <param name="sloganText" type="text" default="" size="50" label="Slogan" description="Your Slogan Text" />
</params>
...
</install>
```
Retrieve a parameter in the template file

Most importantly, you'll need to create a `params.ini` file and make sure it is writeable (Joomla! will install your template, and if this file is included in your `templateDetails.xml` file, install it to be writable, if the server allows.) The `params.ini` file is key, as that's where the results will be stored for later retrieval by the template.

To retrieve a parameter within your `index.php` page, place the following code were you'd like the parameter to be displayed or referenced by PHP code.

```php
<?php $myParam = $this->params->get( 'parameterName' ); ?>
```

The `parameterName` will be whatever you specify it to be in your `templateDetails.xml` file. So, based on my sample code above, the parameter name can be `logoType`, `logoText`, or `sloganText`.

Useful standard parameter types

Here are the most useful types I've used in a template:

- **text:**
  ```html
  <param name="parameterName" type="text" default="Some text"
  label="Enter some text" description="" size="10" />
  ```

- **textarea:**
  ```html
  <param name="parameterName" type="textarea" default="default"
  label="Enter some text" description="" rows="10" cols="5" />
  ```

- **list:**
  ```html
  <param name="parameterName" type="list" default="" label="Select an item" description=""

  <option value="item1">Item 1</option>
  <option value="item2">Item 2</option>

  </param>
  ```

- **radio buttons:**
  ```html
  <param name="parameterName" type="radio" default="0" label="Select an option" description=""

  <option value="0">1</option>
  <option value="1">2</option>

  </param>
  ```
• hidden variable:
  <param name="parameterName" type="hidden" default=""/>

• calendar display:
  <param name="parameterName" type="calendar" default="5-30-2009"
label="Select your birth date!" description="" format="%d-%m-%Y"
/>

Joomla's documentation site has a great tutorial on template parameters, which
includes a full list of standard parameter types and how to implement them:
http://docs.joomla.org/Tutorial:Template_parameters.

Again, be sure to check out Chapter 9, where we'll implement a parameter function
into our case study template.

Summary
We've taken a look at the essentials you'll need most to constantly look up, from
jdoc include tags and standard CSS output, to overrides, chrome, and template
parameters. I've also included key points to be noted for all you template developers
upgrading 1.0 templates to 1.5. Dog-ear this chapter for handy reference, and let's get
ready to start cooking. First up: dynamic layouts, drop-down menus, and Flash.
Where to buy this book

You can buy Joomla! 1.5 Template Design from the Packt Publishing website:

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.