razorCMS
Theme Development Guide

Creating a theme
To create a new theme for razorCMS, there are a few things that need to be explained first, mainly the interface for introducing the theme into the system, as well as the handles and function calls that exist in the theme itself that integrate the system into the xhtml template or public facing front end as I like to call it.

Installing new themes
Themes are added to razorCMS in just the same way that any add-on is added to razorCMS. By using the add-on system called the blade pack interface, all themes can be uploaded into the blade_packs directory and activated using the blade manager in the simplest of ways. This makes adding new themes a breeze and gives you the ability to switch them off and on with a single click. Once activated, the themes can be allocated to one of 4 sockets in the settings manager. This gives the end user the ability to use a single theme allocated to the default socket, or choose from a further 3 themes when creating pages to offer a true multi themed approach.

The theme blade pack structure
It is very important that the structure for creating a new theme is correct, this ensures good compatibility with the system and more importantly with other add-ons.

A theme pack is split up into several parts, the blade pack connecting the theme to the system, a xml file containing all settings, an XHTML template file used for the html structure for the site, the CSS template containing all style information for the website front end, an editor css file to match the front end theme and any images or script files that are needed.

In addition to this, the blade_pack is distributed as a parcel file (zip), this ensure a single method for transportation, automatic installation and management.

The file structure of a theme blade pack is as follows, please ensure that the filenames all contain theme_ before the actual name of the theme, and that all filenames match, to keep things uniformed.

theme_themeName.xml
theme_themeName.php
theme_themeName
    |-------- themeName_xhtml.php
    |-------- themeName_css.css
    |-------- editor_css.css
    |-------- any other files

theme_themeName.xml
This is the settings file which holds all the info on the blade pack and is used when referencing the theme.

theme_themeName.php
This is the blade pack used to integrate the theme into the system.

theme_themeName
This is the folder where all theme blade pack files are placed.

theme_themeName / themeName_xhtml.php
This is the html/xhtml template file, this can be written in xhtml or html, in either strict, transitional or frameset, adjust the filename accordingly to suit and your doctype to match in your header section. Ensure any work is standards compliant.

theme_themeName / themeName_css.css
This is the style file, it contains all style information regarding the theme, this is stored as css in a filename ending in '.css'. Ensure any work carried out is standards compliant.

theme_themeName / editor_css.css
This is the style file for any WYSIWYG content editors, it contains a cut down version of the themename_css file, enabling the editor to show content in the editor, as though it was displaying just the content part of the main theme, this is stored as css in a filename ending in '.css'.

Any other files
This is any image files used in your theme, and javascript files that contain any special scripts.

Creating a new structure
The easiest way to get started is to download the sample theme blade pack archive, this is a template set on which all new blade packs can be built. The archive contains a sample xml file, blade pack connecter and a sample folder containing a sample XHTML and both CSS template files.

Once you have downloaded the sample archive and unpacked it, the first port of call should be to think of a name for your theme, try to make this original and unique, once you have this rename all of the files and folder names to match your new theme name by replacing themeName with your new name. Ensure all you files follow suit, try and keep them all called the same this will make it easier to manage the files in a system full of add on's.

Now we nearly have our structure in place we have one last task to complete, we must alter the xml file that you renamed just a minute ago, we need to change a few things in order to integrate the theme blade pack into the system and correctly set up the blade pack.

Open the xml file that you renamed and change the following things. First alter the details below to change the theme details. Name, version number, auther and a description about the theme. Ensure not to delete the xml tags in the <>.

```
    <details>
        <name>Public Theme - Sample</name>
        <version>v1.0</version>
        <class>theme</class>
        <author>smiffy6969</author>
        <description>Public facing theme giving your
razorCMS install a whole new look, this is a sample
        blade pack to help develop new
themes</description>
    </details>
```

Ensure you keep the layout the same to keep uniformity throughout blade packs.

Next alter the following to match your filenames, be sure to set the parcel filename too, and ensure when you create your parcel once your theme is complete, to give it the name you set here

```
    <archive>
        <archive_file>theme_sample.zip</archive_file>
        <xml_file>theme_sample.xml</xml_file>
        <bladepack_file>theme_sample.php</bladepack_file>
        <bladepack_dir>theme_sample</bladepack_dir>
    </archive>
```

Lastly, you can if you like change the default message displayed on successful installation, although this is set to a default message to explain how to use the theme, so you may want to just leave it as is.

```
    <note>
        <p>This theme blade pack will add the ability to
use a different theme on the
        front end of your website. To use this theme on
the front end of your website,
        complete the following steps.</p>
        <p>Activate this theme in the theme section of
the blade manager, once complete
        go to the settings manager and set which theme
slot you wish to allocate this theme
        to. You may allocate this theme to any slot.</p>
        <p>If set to default, this theme will be used on
your entire site, if set to one
        of the 3 additional slots, when creating or
editing a page, you may select for
        that page alone which slot you wish it to use for
it's theme.</p>
```

```
        <p>You may see what slot is allocated to each
page when managing content by looking
        for D (theme-default), 1 (theme-one), 2 (theme-
two), 3 (theme-three) in the column
        to the left of the page name.</p>
    </note>
```

Once complete, save the file and we will move on to the next file to change.

Next alter the php blade pack file you altered the name on earlier, funnily enough it's the one ending in .php, change the following to match your theme name, alter themeName to the name you gave your theme in the three places below.

```
//////////////////////
// Socket Allocation //
//////////////////////

// Add Blades to Sockets $bladeList['blade'] =
'socket'; //
$bladeList['theme_themeNameXHTML'] = 'public-change-
theme';
$bladeList['theme_themeNameCSS'] = 'public-css-address';
$bladeList['theme_themeNameEdCSS'] = 'editor-css-path';
```

Now alter the following to match the lines above, by changing themeName to the name of your theme. Once complete, change the file paths below to match the paths to your theme files.

```
// blade - load new xhtml template //
function theme_themeNameXHTML(&$xhtmlTemplate) {
    $xhtmlTemplate =
'blade_packs/theme_sample/sample_xhtml.php';
}
// end ////////////////////////////////

// blade - load new css template //
function theme_themeNameCSS(&$cssTemplate) {
    $cssTemplate =
'blade_packs/theme_sample/sample_css.css';
}
// end ////////////////////////////////

// blade - load editor css template //
function theme_themeNameEdCSS(&$cssfile) {
    $cssfile = 'blade_packs/theme_sample/editor_css.css';
}
// end ////////////////////////////////
```

So that's about all the changes you need to make to alter the sample theme to a custom theme that you want to create. In addition to this, if you would like to

add support for a script file, you can do so by adding an extra socket to your php file, and a script call in your xhtml file.

To do this simple add this extra socket to your php file under your other 3 sockets, remembering to change your theme name from themeName to your theme name.

```
$bladeList['theme_themeNameScript'] = 'public-script-
address';
```

once added you can add the blade itself, changing themeName for your theme name and the path to the location of your script.

```
// blade - load new script file //
function theme_themeNameScript(&$scriptPath) {
     $scriptPath =
'blade_packs/theme_sample/sample_script.js';
}
// end ////////////////////////////////
```

now this extra socket will add a path to your theme when it is called, but we need to add in the code to your head section of your xhtml file to do this. Open up your xhtml file, and add the following

```
<script type="text/javascript" src="<?php scriptPath(); ?
>"></script>
```

just before the </head> closing tag, this will now call the extra socket that we just added in. Once this is complete you will be able to use the script within your page.

Writing your theme
Next you need to write your theme, now unfortunately I can't help you with that, it is down to you and your artistic flair. But what can do is give you some advice on the system and how it integrates.

The three files you want to alter are the xhtml/html template file, the css template file and the editor css file, leave the editor until last, as you can do this by copying your main css file pasting it into your editor css file and simply changing it to match when using an editor. These are where your theme code should be placed. Any images or scripts used in creating the theme should also be placed in the same folder as the template files.

When creating your xhtml template, there are several function calls and sockets that need to be placed in the template, this will ensure that all information is displayed in the public front end, and that any add-ons are compatible with your theme.

Whilst it is ok to leave out any of the following function calls as these can omit things like top navigation, something that is not always desired, it is

recommended that all sockets be placed in your template. Failure to include a socket could result in an add-on not functioning with your template.

Please ensure that all code is standards compliant, themes written will be verified for compliant XHTML/HTML and CSS, only those passing will be included into the main repository of blade packs.

Function calls
The following function calls are used to display data within the template.

<?php loadSettings('sitename'); ?>

Loads the name of the website, do not alter this code when using it. Can be used in the title area of a template and anywhere you wish to display the website name.

<?php loadPageTitle(); ?>

If you have an optional page title stored for your page when creating content, this will display this title, if you have not got a page title and the menu title is used instead. Can be used in the title area of a template and anywhere you wish to display the loaded page title.

<?php cssLocation(); ?>

Is used for the reference to the css file, do not alter this code when using it. This corresponds to the css file location that was altered in the blade pack connector earlier. Use this as the css file reference in the link command, place it in the href part of the link declaration.

<?php scriptPath(); ?>

Is used for the reference to a script file, do not alter this code when using it. This corresponds to the script file location that was altered in the blade pack connector earlier. Use this as the script file reference in the link command, place it in the src part of the script declaration.

<?php loadSettings('siteslogan'); ?>

Used to display the site slogan, do not alter this code when using it. This will display the website slogan where ever it is placed in the template.

<?php loadLinks('categoryName'); ?>

Use this to display category links from any given category, change the text ‘catagoryName’ to the name of the category
Your pages are stored under. Default category names that ship with the core download include ‘top-navigation’ for navigation across the top of the page, ‘sidebar’ for displaying a side navigation and ‘footer’ for displaying footer navigation.

<?php loadSlabContents(); ?>

Use this to display the page contents, this should only be used once on a page, do not alter the code when using it. Place this where you wish to have page content displayed on your site.

<?php loadInfoContents(); ?>

Use this to display the info column on your site, this is the place that all page content saved in the infobar category are placed when allocated to a page within the content manager. Do not alter this code when using it.

<?php loadSettings('copyright'); ?>

Use this to display the copyright data that is set in the settings manager. Do not alter this code when using it.

Sockets
The following sockets are important, they must be placed into the xhtml template in the correct place to ensure all add-ons are compatible with your template.

<?php BsocketB('public-xhtml-head1'); ?>

Place this socket on the line before the css link declaration in the head of your html/xhtml document.

<?php BsocketB('public-xhtml-head2'); ?>

Place this socket on the line after the css link declaration in the head of your html/xhtml document.

<?php BsocketB('public-xhtml-header'); ?>

Place this socket in the header part of your html/xhtml document, ensure it comes after your website name or logo.

<?php BsocketB('public-xhtml-topnav'); ?>
If you have a top navigation, place this on the line after the function call to load the navigation links, if you do not have a top navigation in your template, place this as the last line in your header section.

<?php BsocketB('public-xhtml-content'); ?>

Place this socket on the line directly after the function call to load the page content.

<?php BsocketB('public-xhtml-leftnav'); ?>

Place this socket on the line directly after the function call to load any sidebar navigation. If your template has no sidebar navigation, place this socket before the above content socket.

<?php BsocketB('public-xhtml-leftbar'); ?>

Place this socket on the line directly after the function call to load info contents. This is normally placed in the side bar, but can be placed anywhere the info contents are loaded.

<?php BsocketB('public-xhtml-footer'); ?>

Place this socket as the last line in the footer section of your template.

<?php BsocketB('public-xhtml-endofdoc'); ?>

Place this socket on the line above your closing body tag </body>.

If you are at all unsure of the placement of any sockets when writing your theme blade pack, please visit the support forum for help and advice. All sockets will be checked before adding a theme blade pack into the main repository.

Optional calls

Display site name as a link to the home page

To display the main site name at the top of your page, with a link back to the home directory location, use the following call.

```
<h1><a href="<?php echo $_SESSION['PHP_SELF']; ?>"><?php
loadSettings('sitename'); ?></a></h1>
```

Drop down menus

Everyone loves a good drop down menu don't they, razorCMS works with drop down menus, so here's a way to get them working.

First of all you will want to use this with the top navigation category, as drop downs don't really work on side menus, although they can be adapted to work with side menus.

Anyhow, first you need to add some pages to your top navigation category in your back end, then you need to create sub categories and assign them to each of the pages in the top navigation. Once complete, you can add pages to the sub categories (these will be the page links that actually drop down).

Once you have completed this, you can add the following code to give a pure css drop down menu.

Add this to your css file, this will add in the basic drop down menu structure that is already present in the default razor theme

```
/*
##################################################
######## */
/*
##################################################
######## */
/* Topnav css */
/*
##################################################
######## */
/*
##################################################
######## */

#topnav {
   margin: 0px 0px 0px 0px;
   padding: 0px;
   width: 944px;
   /*background-color:  #000000;*/
   background-image: url(images/navbar.jpg);
   background-position: center;
   background-repeat: repeat-x;
   border-left: 2px solid #000000;
   border-right: 2px solid #000000;
   text-align: left;
   float: left;
}


#topnav ul, #topnav ul ul {
   padding: 0;
   margin: 0;
   list-style: none;
   width: auto;
}


#topnav ul a {
   padding-top: 2px;
   padding-bottom: 4px;
   padding-left: 10px;
   padding-right: 10px;
   display: block;
   width: auto;
   height: auto;
   color: #60bfff;
   text-decoration:none;
   font-size: 85%;
}

#topnav ul li a:hover {
   color: #94d4ff;
   background-color:  #333333;
   background-image: none;
```

```
}

#topnav a.active {
   color: #ff0000;
}

#topnav a.active:hover {
   color: #ff0000;
   background-color:  #333333;
   background-image: none;
}

#topnav ul li ul li a{
   color: #1791e4;
}

#topnav ul li ul li a:hover {
   color: #60bfff;
}


/* top level non hover property */
#topnav ul li {
   float: left;
   width: auto;
   border-right: 1px solid #000000;
   height: auto;
}


#topnav ul li ul{
   position: absolute;
   left: -999em;
}


/* top level hover property */
#topnav ul li:hover, #topnav ul li.hoverfix {
   float: left;
   width: auto;
   background-color:  #333333;
   background-image: none;
   height: auto;
}


/* Drag in 2nd level menu */
#topnav ul li:hover ul, #topnav ul li.hoverfix ul {
   left: auto;
   z-index: 2;
   border-left: 1px solid #000000;
   border-right: 1px solid #000000;
   border-bottom: 1px solid #000000;
   background-color:  #333333;
}
```

```
/* 2nd level non hover property */
#topnav ul li:hover ul li, #topnav ul li.hoverfix ul li {
   clear: left;
   background-image: none;
   height: auto;
   border: 0px;

}


/* 2nd level hover property */
#topnav ul li:hover ul li:hover, #topnav ul li.hoverfix ul li.hoverfix {
   clear: left;
   background-image: none;
   height: auto;
   border: 0px;
}
```

Be sure to alter this code, to load in your images, and to match your theme look, as this is the default code from the default css file in razor, but it is a good start at creating a drop down menu.

Unfortunately due to IE6 not working with hover properties on li elements, this will not work in IE6 or earlier, but is compatable in IE7, 8, firefox, opera, and other major browsers.

If you wish to get this working with IE6, you will need to apply a javascript hack, to do this, simply add the navbar.js file from default theme in razor to your html file.

You can accomplish this by following the instructions earlier in this document on adding a script file to your theme.