BASICS OF TELERIK R.A.D.CONTROLS SKINNING

PREFACE

With the advent of the new trends and opportunities of Web 2.0, more and more developers and designers have realized that a successful web application should smoothly balance between server and client technology. That is why modern websites make heavy use of CSS and JavaScript, which were highly underestimated in the previous years. The new tendencies, as well the requirements of the W3C necessitated a clear division between functionality and presentation, which was the pathway to XHTML. The inevitable consequence was that many companies began to hire professionals with clearly divided spheres of expertise – pure server-side developers and pure client-side ones.

This tutorial is intended for intermediate / advanced developers from both groups, as we understand, that many of our customers are server- and client-side developers, and designers at the same time. We will try to explain the basics of creating successful skins for our r.a.d.controls components with a minimum of effort and expense of time.

CREATING CUSTOM SKINS FOR R.A.D.CONTROLS

Each distribution of r.a.d.**controls** includes the Quick Start Framework (QSF), which is also available online on our website. The QSF is a special tool, presenting a given component in a real-life environment – application scenarios, functions, skins preview, client and server API, etc.

1. UNDERSTANDING THE BROWSER OUTPUT CODE

As r.a.d.controls are server controls, before starting the actual work on creating a custom skin, first of all the developer should be well-acquainted with the generated HTML output of the control, so make sure that you find the relevant article discussing this issue in the help file of the component you are about to skin.

Obviously, it is impossible to include everything in such static file, and you will notice, that parts of the code, especially the one, changed dynamically with JavaScript are not available there. Fortunately, there are a couple of really useful tools facilitating the task of viewing the browser-generated source code and making the life of the developer easier. You can use free of charge these two:

Microsoft Internet Explorer Developer Toolbar (current version Beta 2)

Mozilla Web Developer Extension (current version 1.0.2)

These small apps provide a variety of useful tools, that are of extreme importance for creating custom skins for r.a.d.controls – options for viewing the generated page source on the client, DOM-tree viewer, disable / enable JavaScript, CSS, images, forms, etc.

Another cool way of viewing the current page source state is with JavaScript – you can create a small script of the type:

Listing 1

```
	// view generated page source
function viewGeneratedPageSource()
{
	f = window.open('', 'ViewSource', 'width=800,
height=600');
	f.document.write('<xmp>' +
document.documentElement.innerHTML + '</xmp>');
}
```

That will show every dynamic change of the page each time you fire the function. You can easily extend it by adding formatting to the parsed code with regular expressions, replace(), split(), etc.

2. Six easy steps to design a skin, based on an existing one

In order to make things easier, we will assume that we are creating a custom skin not from a scratch, but one based on an existing skin. Let's take r.a.d.menu's ***Inox*** skin. The first logical step is to find the Appearance / Skinning section of the control in the QSF and to select a skin, which resembles most the design of the skin we want to create. The main points of comparison should be:

Listing 2

1. Color
2. Fonts
3. Design parameters – rounded corners, border-width, etc.
4. Any other point of resemblance you could find useful – this will save a lot of effort and time

1. Make sure you remember the name of the chosen skin. All r.a.d.controls skins are CSS based, and include .css file, images, arrows, icons, buttons etc. These are stored in **~/RadControls/ControlName/Skins/SkinName**, for example:

   The **Inox** skin of r.a.d.menu is stored in the **~/RadControls/Menu/Skins/Inox** folder. If you navigate there, you will find the **Style.css** file, containing the CSS definitions and the **img/** folder with all necessary images for the skin. It is important to remember, that the folder, containing the skin has the same name of the skin itself.

2. After you have found the skin you will modify (for example the **Inox** skin), copy it's folder in the RadControls/**Menu**/**Skins** directory, and rename it. Make sure you give it a logical name, otherwise you risk finding it difficult to get back to it if needed. Let's name it **MyFirstSkin**.

3. One of the common things between the latest versions of r.a.d.controls is that we have tried to unify the skinning mechanism for most of our components and thus to make it easier to modify and understand. In order to go behind the logic of the r.a.d.controls CSS classes, please open **Styles.css** in the **MyFirstSkin/** folder. You will notice that the name of the classes is formed by: **theNameOfTheControl** + **underscore** + **theNameOfTheSkin**, i.e.:

Listing 3

```
.RadMenu_Inox
```
(if you had copied the Inox skin of r.a.d.menu)

Obviously, the next step we have to take is to give the correct name of the classes in our custom skin. The easiest way is with *Find and Replace*. Replace all instances of **_Inox** with **_MyFirstSkin**. Now, in the .aspx change the skin property to **Skin="MyFirstSkin"** and hit F5. If the above steps were followed correctly, you will have an exact copy of the **Inox** skin but with a different name - **MyFirstSkin**. Now we are ready to modify the skin according to our new design.

> **IMPORTANT:** In order to avoid multiple class accumulation, r.a.d.controls skins make heavy use of CSS class inheritance, i.e. every class inherits the rules of the parent container. For example:

Listing 4

```
.RadMenu_Inox .link
```

> This means that any `.link`, that is not within the `.RadMenu_Inox` will **not** apply in r.a.d.menu.

4. Make sure you understand the output code, the responding CSS classes and the images contained inside the **img/** folder. If you face difficulties, please, refer to chapter **1. Understanding Browser's Output Code**. A good way of *debugging* CSS is to set `border: solid 1px red !important;` to the classes you are not sure about, as this will override any other rules and will highlight them.

5. Open your design file with Adobe© PhotoShop or with your favorite image software (You can also take a look at our skins library at http://www.telerik.com/skins where we have provided almost all r.a.d.controls skins along with the respective .psd files). Use the horizontal and vertical rulers to outline the different elements of your new skin. Make sure you preserve the original names of the image files (simply overwrite the current ones) – this will help you avoid confusion with existing images. After reloading the page, you will see your new images on the place where they are supposed to be. Do **not** get frustrated if they look a little (or much) displaced. That is because you still haven't set their proper dimensions in the Styles.css file.

6. Find the image filenames in Styles.css and edit the correspondent css classes. Set new dimensions, borders, colors, backgrounds, background-repeats, fonts and any other property required by design. It is important to have in mind several points:

Listing 5

The direction of the background-repeat, or is there a real need of using background images instead of solid colors;
Paddings and margins – if not set properly, these behave slightly different in different browsers and you might get unexpected and unpleasant results;
Widths, heights, line-heights and displays;
Be aware of the browser-specific CSS properties we have used for some of the classes (`-moz-propertyName:` - Mozilla extensions, `_propertyName:` – IE hacks, `filter:`, `important!`).

Conclusion

The correct understanding of the skinning logic, combined with the good notion of the HTML architecture of a particular component takes a great part in the process of designing custom skins for r.a.d.controls.

In order to create a successful, useful and browser-compatible skin, please, feel free to examine the code in the existing skins, or make use of the tools we have described in the first chapter.

Good luck!