Client-Side Development Made Easy

Listen-up ASP.NET developers! If you're not writing *some* JavaScript in your interactive web applications, you're probably not delivering the high performing, dynamic applications your customers expect. As a class, ASP.NET developers have been particularly shielded from JavaScript because ASP.NET has long written JavaScript automatically to enable ASP.NET client-side features. That hasn't changed, but as the web continues to mature, users are expecting increasingly dynamic and interactive applications- interactivity that can only be delivered with some JavaScript coding.

The problems with JavaScript code, and the reasons many of us have avoided it in the past, can generally be summarized as follows:

1. It's difficult to test (especially cross browser)
2. It's difficult to debug
3. It requires duplication of effort already done server-side
4. It requires learning a new programming model and lots of manual work

Fortunately, tools like FireBug in Firefox, IE Developer Toolbar and Visual Studio 2008/2010 have started to minimize the impact of the first two points. It's now just about as easy to test, debug, and even step JavaScript code as it is any other application code. ASP.NET AJAX and the Microsoft AJAX Library have really helped reduce effort duplication, too. You can now easily write code on the server and quickly expose functionality to JavaScript via auto-generated proxies.

That leaves "manual work" as the chief "JavaScript complaint." Or in other words, the idea that to do things client-side requires you to learn a whole new way of doing things, and that way usually requires you to write lots of code to get anything done. So, if you accept that you must start writing JavaScript to create rich, high performance applications, your goal should be finding ways to reduce this manual work and finding ways to reuse your server-side knowledge on the client.

Enter the RadControls for ASP.NET AJAX. Built entirely on top of Microsoft ASP.NET AJAX, and with nearly identical server-side and client-side APIs, they are tools that make the transition to JavaScript programming extremely easy. The RadControls have always enabled you deliver rich client-side experiences without writing *any* code - and they still do - but "real world" applications often require some customization. With RadControl client-side APIs, this process is not only easy, it's kinda fun.

Reuse Your Server-Side Knowledge

First and foremost, when it comes time for you to start enhancing your applications with JavaScript, you must face the task of learning how to do on the client what you've always done on the server. With the RadControls for ASP.NET AJAX, there is little to learn. The RadControl APIs are nearly identical server- and client-side, which means if you can set a property on the server, you can also set it on the client. If you can handle an event on the server, you can also handle it on the client.

Let's take a look at the RadGrid for ASP.NET AJAX. The RadGrid client-side API makes it very easy to bind your grid to data from a web service using JavaScript. In fact, you can even bind a RadGrid to web server data client-side without writing *any* JavaScript, like this:

ASPX

```
<telerik:RadGrid runat="server">

    ...

        <ClientSettings>

            <DataBinding Location="~/YourWeb-
        Service.asmx"

                    SelectMethod="GetData"
        SelectCountMethod="GetCount" />

        </ClientSettings>

</telerik:RadGrid>
```

By simply setting three properties- Location, SelectMethod, and SelectCountMethod- your RadGrid is instantly ready to bind directly to a web service and load data on the client. No "manual" JavaScript required. This approach can significantly improve your application's performance because you are radically reducing the amount of data that you must send between the browser and server for

Grid operations like paging, sorting, and filtering.

Maybe you have a custom scenario, though, and you *want* to write some JavaScript. Using the RadGrid client-side API, you can quickly and easily bind the grid to results returned by a web service and take complete control of your client-side binding:

JAVASCRIPT

```
<script type="text/javascript">
   function getSomeData()
   {
        YourNamespace.YourWebService.
GetData(updateGrid);
   }

   function updateGrid(result)
   {
        var tableView = $find("<%= RadGrid1.
ClientID %>").get_masterTableView();
        tableView.set_dataSource(result);
        tableView.dataBind();
   }
</script>
```

Look familiar? It should. Here's how we might bind a RadGrid server-side:

C#

```
var data = MyBusinessLayer.GetData();
radGrid1.DataSource = data;
radGrid1.DataBind();
```

Notice that the property and method names are very similar. The only difference on the client is that the Telerik client-side API follows the Microsoft AJAX Library naming conventions, so all properties are prefixed with "set_" or "get_" and all methods begin with lowercase names (such as "dataBind()" versus "DataBind()"). Other than that, the APIs are identical, and it's this crossover that makes it very easy to move from server-side development to client-side coding.

Real applications require more than "read only" data interactions, though. What if you want to delete an item from your RadGrid client-side? Again, the RadGrid provides a code-less method:

ASPX

```
<telerik:RadGrid ID="RadGrid1" DataSourceID=
"ObjectDataSource1" runat="server">
   <MasterTableView
AllowAutomaticDeletes="True"
DataKeyNames="YourKeyId">
      <Columns>
         <telerik:GridClientDeleteColumn
ConfirmText="Are you sure you want to delete
the selected row?" HeaderStyle-Width="35px"
ButtonType="ImageButton" />
     </Columns>
   </MasterTableView>
</telerik:RadGrid>
```

That's it! When you add a GridClientDeleteColumn, rows can be instantly deleted on the client without writing any code. On the next post to the server the changes will be persisted to your data source. But for more custom scenarios, you can also easily handle one of the RadGrid's client-side events, in this case OnRowDeleting:

JAVASCRIPT

```
<script type="text/javascript">
    function RowDeleting(sender, eventArgs)
    {
       var rowId = eventArgs.
getDataKeyValue("OrderID");

       //Call service to delete item, then
rebind the grid (again client-side)
       YourNameSpace.YourWebService.
DeleteByKey(rowId, updateGrid);
    }
</script>
```

This pattern is repeated across the RadControls, where properties are provided that enable rich client-side functionality without requiring any manual code, but a powerful API is available for more complex scenarios. In all cases, the RadControl client-side APIs mimic their server-side counterparts, making it very easy to take code you would have written server-side and move it to the client.

Embracing Client-side Shortcuts

If you find yourself writing some "serious" client-side code, you'll likely want to take advantage of ASP.NET's support for jQuery. jQuery is an open source JavaScript library that makes JavaScript programming nothing short of fun (I know- "JavaScript" and "fun" in the same sentence is jarring, but don't doubt until you've given jQuery a try). With jQuery, you can easily create dynamic HTML animations with very little code. If we want to make an "error message" DIV appear with an attention grabbing animation, for instance, we could simply write code like this:

JAVASCRIPT

```
function updateGrid(results)
{
     //If results are empty, display an er-
ror using jQuery
     if(results == null)
        $("#msgDiv").addClass("error").
show("slow");
     else
        //Bind the grid client-side
}
```

The RadControls for ASP.NET AJAX leverage jQuery internally to deliver great client-side performance and interactivity with low "page weight" (or less code that has to be downloaded to the client). You can even use the version of jQuery embedded in the RadControls for your own development like this:

ASPX

```
<asp:ScriptManager ID="ScriptManager"
runat="server">
   <Scripts>
      <asp:ScriptReference
Assembly="Telerik.Web.UI" Name="Telerik.Web.
UI.Common.Core.js" />
      <asp:ScriptReference
Assembly="Telerik.Web.UI" Name="Telerik.Web.
UI.Common.jQuery.js" />
      <asp:ScriptReference Path="~/jquery-
telerik.js" />
   </Scripts>
</asp:ScriptManager>
```

JAVASCRIPT (jquery-telerik.js)

```
//Make jQuery from Telerik assembly avail-
able via the

//default $ shortcut
var jQuery = window.jQuery = window.$ =
$telerik.$;
```

With that code in your application and the RadControls, you'll be ready to deliver incredible client-side interactivity to your customers with no need to separately maintain the jQuery JavaScript file. For more great examples of how you can use jQuery to enhance the RadControls, don't miss these posts on the Telerik Blogs:

- RadComboBox jQuery Blog Post
- Animate RadGrid Headers using jQuery
- RadGrid Client Side Data Binding using jQuery

Wrapping-up

If you think JavaScript is a dying language, you'd be mistaken. Increasingly, modern web applications are utilizing JavaScript to deliver dynamic, responsive experiences. You can deliver these same experiences in your own applications today, and you can do it without learning a whole new programming model or banging your head against the wall. With ASP.NET AJAX and the Telerik RadControls for ASP.NET AJAX, you can quickly build web applications that leverage the power of the client without hurting your productivity, enabling you to easily deliver "top performance" web applications

It's so easy, in fact, it almost feels like cheating (we promise we won't tell). JavaScript development has never been so productive or felt so "natural." Find out for yourself by downloading a free trial of the RadControls for ASP.NET AJAX today. I think you'll discover Telerik really has finally made client-side development easy!

Download your free trial of
Telerik RadControls for ASP.NET AJAX