TOOLBOX

Rich AJAX Data Controls, Analyzing HTTP Traffic, And More

Scott Mitchell

All prices confirmed at press time and are subject to change. The opinions expressed in this column are solely those of the author and do not necessarily reflect the opinions at Microsoft.

Contents

Rich Data Web Controls for Client-Centric Ajax Development

The ASP.NET AJAX framework offers two models for building Ajax-enabled Web applications: server-centric and client-centric. With the server-centric model, page developers continue to use the standard ASP.NET Web controls, but place them within an UpdatePanel control. When a control in an UpdatePanel causes a postback, the UpdatePanel replaces the typical postback with a JavaScript-induced partial page postback and seamlessly updates the controls within it with the server's response. With the client-centric model, page developers are tasked with writing the JavaScript to initiate a partial-page postback and to update the page on response. Programming using the client-centric model requires that the page developer use the client-side ASP.NET AJAX libraries and write JavaScript and HTML themselves to initiate partial page postbacks and to update the page on response.

ASP.NET developers new to Ajax development typically prefer the server-centric model, as it is easier to implement and uses the existing Web controls ASP.NET developers are already familiar with. However, the client-centric model offers more control over the page's behavior and the information shuttled between the browser and the Web server, thereby enabling a more unique and responsive user experience. These two models force developers to make a tough decision: use ASP.NET's existing Web controls to ease implementation, or forgo the familiar controls and write a lot of JavaScript and HTML in order to maximize performance.

The good news is that there is a third option, thanks to the Ajax Data Controls project version 1.0, which is a collection of ASP.NET controls for displaying data in an Ajax-enabled Web application. The Ajax Data Controls marry the ease of server-centric development with the performance enhancements enjoyed by client-centric development.

Adding an Ajax Data Control to a Web page and configuring its appearance and behavior is done much in the same way as any other Web control—from Visual Studio, drag the control from the Toolbox onto the page and configure its settings via the Properties window. But unlike ASP.NET's built-in data controls, the Ajax Data Controls retrieve their data from the server using client-side script, providing a more responsive user experience and consuming less bandwidth than the standard data controls when used within an UpdatePanel.

For example, to display categories from the Northwind database using the Ajax Data Control's GridView, start by adding the GridView to an ASP.NET page. Next, create a script service that queries the database and returns the data to display. Finally, write a bit of JavaScript in the page to bind the GridView to the data. **Figure 1** shows the page's declarative markup while **Figure 2** shows the resulting page when viewed through a browser. When the page is visited by a browser, the client-side pageLoad function executes and retrieves the data from the script service. Once the server has returned the data, it is bound to the GridView using JavaScript code that's similar to the server-side C# code used to bind data to ASP.NET's GridView control.

Figure 1 Adding the GridView to an ASP.NET Page

Copy Code

```
<AjaxData:GridView ID="gvCategories" runat="server" CellPadding="4"
CellSpacing="0">
    <HeaderStyle CssClass="HeaderStyle" />
```

```
    <AlternatingRowStyle CssClass="AlternatingRowStyle" />
</AjaxData:GridView>

<script type="text/javascript">
    function pageLoad(sender, e) {
        MyScriptService.GetAllCategory(onLoadSuccess);
    }

    function onLoadSuccess(result) {
        var myGrid = $find('gvCategories');

        myGrid.set_dataSource(result);
        myGrid.dataBind();
    }
</script>
```

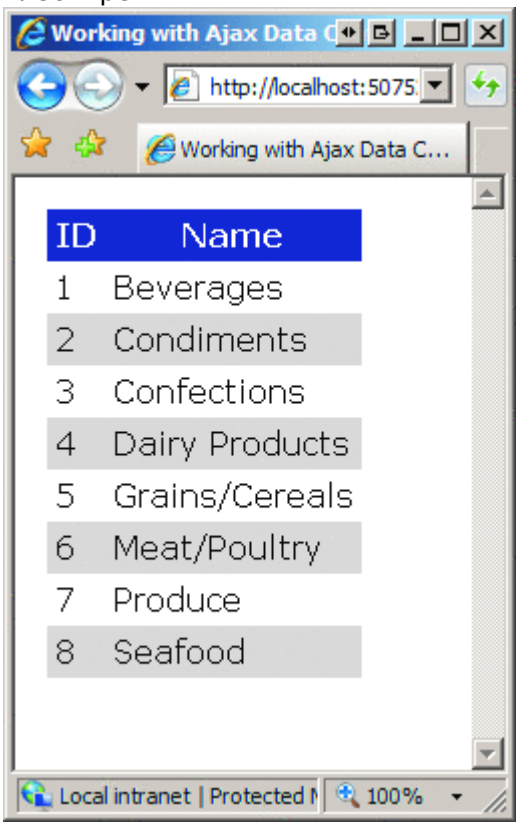


Figure 2 **The Data Displayed**

The Ajax Data Controls include GridView, DataList, Repeater, and Pager. With just a sprinkle of JavaScript, and without having to write any HTML, you can display, sort, page through, edit, and delete data using familiar concepts while enjoying the benefits of client-centric development. A number of common data display scenarios are also easy to implement, such as conditional formatting, using different column types in the GridView (images, checkboxes, hyperlinks, and so on), nested data controls, and integration with the Ajax Control Toolkit. There are also rich features not offered by the standard ASP.NET data controls, including drag and drop and column reordering.

Price: Free, open-source.

codeplex.com/AjaxDataControls

Blogs of Note

Good bloggers share anecdotes with their peers to let them know what technologies they are using, to pass on what works, what doesn't, and what roadblocks to expect. Rick Strahl's Web Log is an excellent example of great blogging.

Rick is the founder and lead developer of West Wind Technologies, a company that sells a number of Web-based applications and utilities, so he spends his days living in the trenches—writing software, bumping into problems, finding solutions. His blog serves as a virtual water cooler, a place where Rick shares clever tips and tricks along with the new technologies he's using, problems he's encountered, and workarounds he's devised.

Because West Wind Technologies primarily builds products for the Web, the majority of Rick's posts center on such technologies. There is a rich collection of posts on jQuery and JavaScript, plus insights on ASP.NET and AJAX, Silverlight, Web Services, Visual Studio, and IIS. And in addition to the regular blog posts Rick adds every two to three days, he also presents a number of in-depth whitepapers on a range of topics from compilation and deployment options in ASP.NET to load balancing and stress testing Web applications to setting up and running Subversion (a free source code control system).

Rick's blog offers a wealth of information and know-how that has been learned through real-world experience. This extensive experience serves as an invaluable resource for Web developers of all skill levels.

west-wind.com/weblog

One of my other favorite blogs is **My Secret Life as a Spaghetti Coder**, in which Sam Larbi shares his thoughts and insights with the world. Sam works as a developer creating both Web and desktop applications using languages and technologies like C#, C++, Ruby, ASP.NET, and ColdFusion. His blog covers a similarly diverse spectrum of topics.

There are posts focusing on software development methodologies, musings on Web development, and Sam's experiences with test-driven development. You'll find plenty of fun entries on topics such as game programming, gift ideas for programmers, and how to get kids interested in programming.

Sam also has a lot to say about all of the non-technology-related aspects of being a software developer: working with others, meetings, maintaining good customer relations, personal development, and so forth. There's a great post on how to respond to a major blunder with a boss or customer. Sam's advice is to embrace failure—take responsibility, explain how you'll avoid the mistake going forward and how you will rectify the current situation. There are also great tips on how to deal with troublesome customers and what knowledge and people skills you need to become an invaluable employee in the eyes of your employer.

codeodor.com

Inspect and Analyze HTTP Traffic

When you visit a Web site, your browser sends an HTTP request for each resource and receives an HTTP response from the server with the requested content. Being able to inspect each HTTP request and response is useful in a number of Web site development scenarios. For example, when faced with a slow loading Web site, a good first step is to inspect what, exactly, is being transmitted from the server to the client when a request for an underperforming page is visited. Perhaps there is a very large CSS or JavaScript file being shuttled back and forth or maybe the page's background image is exceptionally hefty. Inspecting the HTTP traffic is also a helpful step when debugging Ajax applications, as it allows you to see the precise content being shuttled back and forth during a partial postback.

One of my favorite tools for inspecting HTTP traffic is Fiddler version 2.2, created by Eric Lawrence, a program manager for the Internet Explorer team at Microsoft (see **Figure 3**). Fiddler serves as a local HTTP proxy; it sits between your browser and the outside Internet. When Fiddler is enabled, every HTTP request made from your browser is first sent to Fiddler, which logs the request before sending it along to its intended destination. When the HTTP response returns, it first arrives at Fiddler, which logs it and then returns it to the browser.

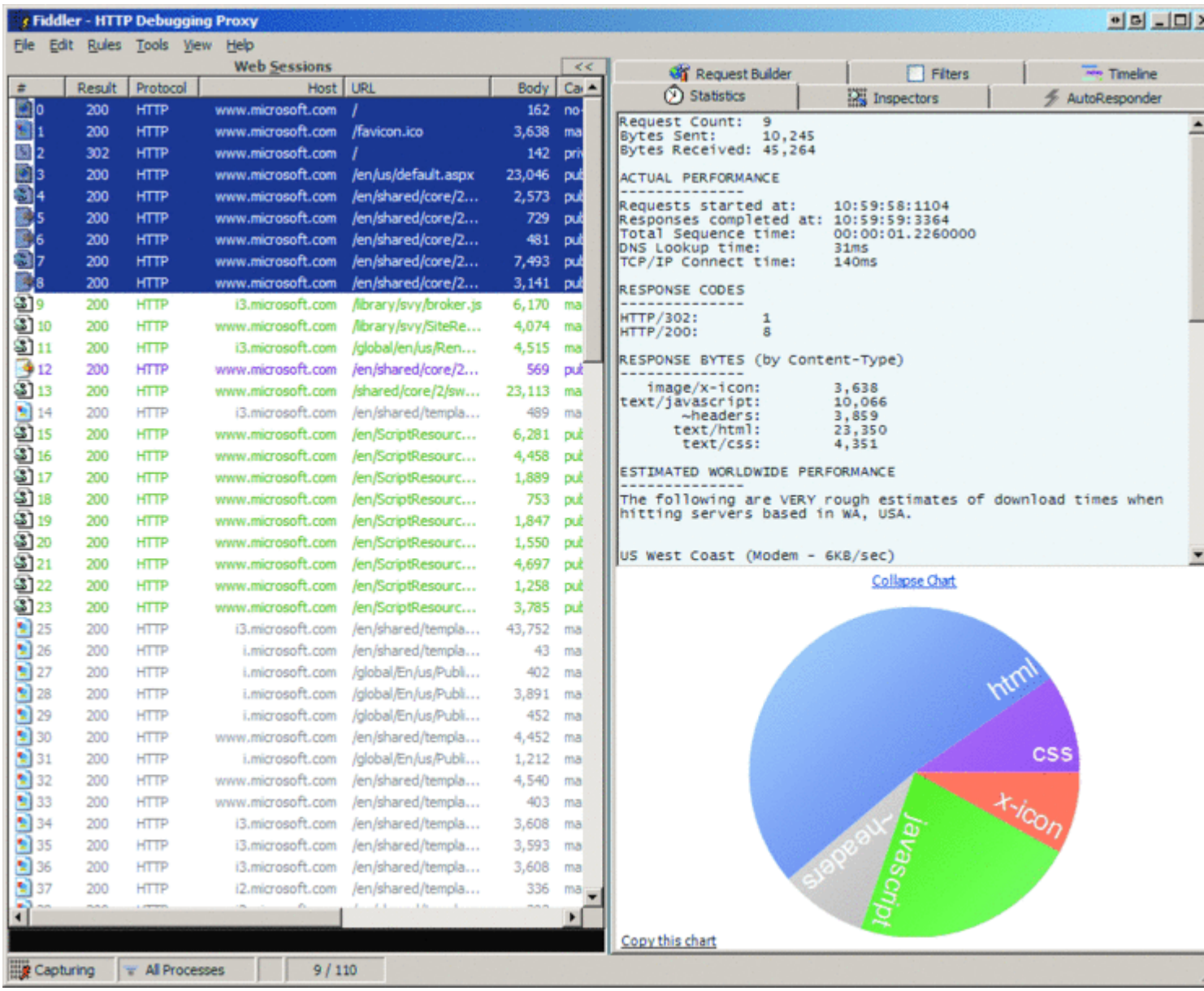


Figure 3 **Fiddler Analyzing Traffic**

The HTTP traffic logged by Fiddler is viewable through a two-paned user interface. The left pane lists each logged HTTP request/response pair. Selecting one or more HTTP request/response pairs from the left causes the details to load in the right pane.

The right pane includes a number of tabs with various kinds of information. The Statistics tab lists the total number of bytes sent and received, estimates on how long the selected request/response pairs would take to transmit in various settings, and a pie chart that breaks down the various types of requests and their sizes, relative to one another. The Timeline tab shows a graphical timeline of each selected request/response pair, illustrates which requests ran concurrently, and shows how long each request took to perform. These two tabs are the most useful ones for analyzing a Web site's performance.

The other tabs are useful for debugging client-side and server-side logic. The Inspectors tab offers formatted and raw views of the contents sent in the request and response. From the AutoResponder tab you can "fake" a response from the server by specifying a predetermined response, a useful technique for debugging client-side logic in Ajax applications. Also check out the Request Builder tab; from here you can construct a hand-crafted HTTP request and send it to a specified Web server.

Price: Free.

fiddlertool.com

The Bookshelf

JavaScript was invented in the mid-1990s as a client-side scripting language for the Netscape Navigator Web browser. For many years it was viewed as somewhat of a toy language— good for form field input validation and little else. Today, JavaScript is recognized as an important and powerful language. It is often used to dynamically modify the style and content of a Web page on the client side without requiring an expensive trip back to the Web server. And the complex interactions that occur in an Ajax-enabled Web page are possible because of this nifty language.

Due to JavaScript's increasing importance in Web development, several frameworks have been created. One of the most popular ones is jQuery, a free, open-source, cross-browser JavaScript framework created by John Resig. (In fact, Visual Studio 2010 will ship with the jQuery library, making it even easier for ASP.NET developers to get started with jQuery.)

In a nutshell, jQuery makes it easy to grab elements from the Web page and do stuff with them. For example, many Web pages that display a grid of data vary the style for each alternating row, which can be accomplished in JavaScript by applying a CSS class to each alternating table row. The following jQuery statement does just that, quite succinctly:

Copy Code

```
$("table tr:nth-child(even)").
   addClass("cssClassName");
```

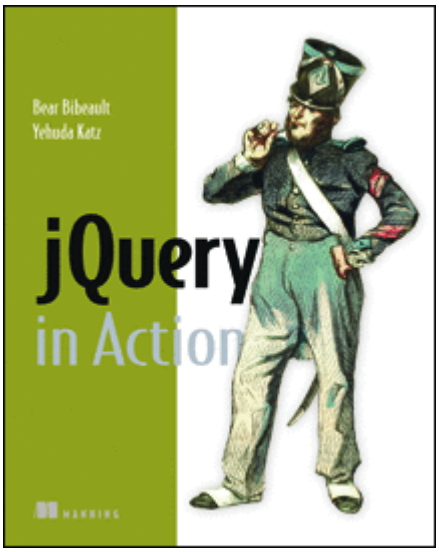
jQuery in Action (Manning, 2008), by Bear Bibeault and Yehuda Katz, is a great resource for leaning the ins and outs of jQuery and for mastering its terse and flexible syntax. *jQuery in Action* assumes the reader is already familiar with JavaScript and wastes no time covering the basics of the language. Instead, it starts with a quick introduction to the motivation behind jQuery and jQuery fundamentals and then moves on to using jQuery to accomplish common tasks. (Some of the more advanced JavaScript concepts that are used by jQuery are covered in an appendix.)

Next, the authors walk through jQuery's many features. They show how to select elements from a Web page, how to modify the style and content of elements, and how to add and remove elements from the DOM. They reveal how jQuery simplifies event handling, examine jQuery's utility functions, and illustrate how to use jQuery to communicate with the Web server. The authors provide this information in simple and straightforward English with just the right amount of code snippets and screenshots. The end result is a very informative and readable book.

One reason for jQuery's popularity is that it is very easy to extend the framework through plugins. With a few lines of JavaScript you can add your own functions to the framework. Best of all, developers can share the plugins they've created at the official jQuery site, where there are currently several hundred plugins available for download. *jQuery in Action* includes a chapter that shows you how to create your own plugins, and another chapter that examines four of the most popular and useful plugins available.

Price: $39.99.

manning.com



Send your questions and comments for Scott to toolsmm@microsoft.com.

Scott Mitchell, author of numerous books and founder of 4GuysFromRolla.com, is an MVP who has been working with Microsoft Web technologies since 1998. Scott is an independent consultant, trainer, and writer. Reach him at Mitchell@4guysfromrolla.com or via his blog at ScottOnWriting.NET.