telerik

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Enabling AJAX in ASP.NET with No Code

telerik's r.a.d.ajax enables AJAX by simply dropping a control on a Web page, without otherwise modifying the application or writing a single line of code

By Don Kiely

Source code from this paper is available for download here:
www.telerik.com/documents/code.zip

Even though the Web enables some amazing interactions between people and organizations around the world, the development of the Web has not been smooth or easy. Some of the earliest enhancements provided ways to serve up dynamic content and allow execution of code on both the server and the client. Yet after 15 years of evolution, the browser is still a stunted alternative to rich desktop applications, constantly flickering and sending redundant information across the network as each user interacts with Web pages. Even though Web applications accessed through browsers are a great cross-platform way to deploy sophisticated applications, they still hearken back to the days of applications hosted on a mainframe that users access with dumb terminals. The next version of the Web requires some fundamental changes in how applications interact with the Web server.

Asynchronous JavaScript and XML (AJAX) could finally be the technology that nudges the next evolution of the Web into existence. Using widely available technologies, it provides a design pattern for elements in a Web page to communicate with a Web server without requiring that the entire page be rebuilt from scratch if only a small part of the page has to change in response to user action. AJAX can be complicated to implement. Some of the complexity arises because at this early stage of its development, there are a number of ways to put it to use.

This paper will discuss AJAX and the problems it solves as well as the problems it causes. It will then look at the AJAX features provided by **t**elerik r.a.d.**ajax** and show how to implement AJAX in a sample application that needs to update multiple controls asynchronously without a complete page refresh. Finally, it will describe the magic behind the product and finish with a summary of the benefits that r.a.d.**ajax** can bring to both new and existing ASP.NET applications.

Asynchronous JavaScript and XML (AJAX)

One problem with Web development prior to using AJAX is that with both static HTML pages and dynamic ASP.NET pages, every time a the user does something to cause an interaction with the server—click a link or a button, sort a grid, etc.—the server must regenerate the entire page and send it back to the browser. The browser renders the HTML and then waits to repeat the cycle. Figure 1 shows this process. On a complex page, this can cause a considerable amount of unnecessary overhead and wastes bandwidth by sending an entire page over the network, including portions that haven't changed along with the small amount that has changed. Worst of all, it keeps Web applications from delivering the clean, efficient interface that users have come to expect from desktop applications.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

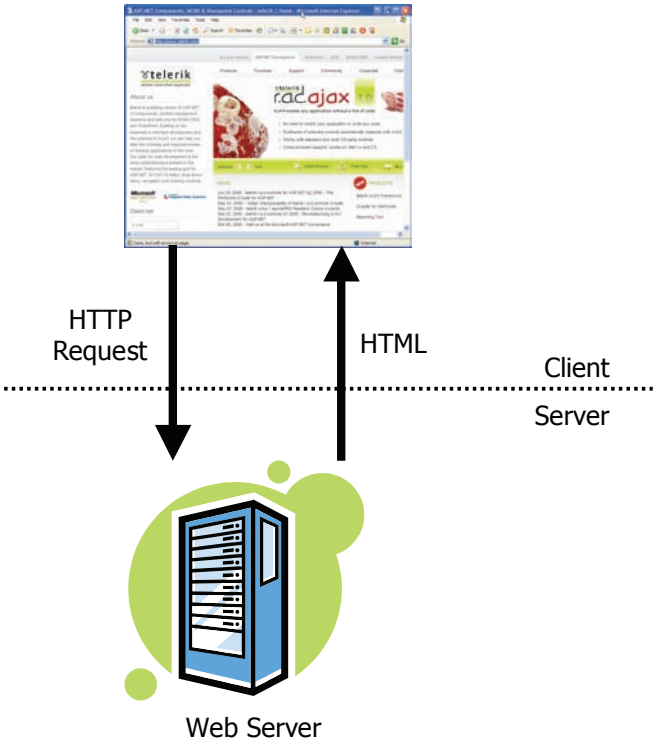

Figure 1. Basic HTTP request and response for a Web page.

AJAX solves this problem by bundling technologies that have been around since late in the last millennium: JavaScript, XML, and a means of making asynchronous calls to the server out of band from the usual HTTP request/response.

When a client initially requests an AJAX-enabled page, the entire page is requested by the client and sent to the browser as usual as shown in figure 1, and the browser renders the page normally. The differences begin when the client clicks a page element that is AJAX-enabled. The client doesn't perform a full postback that once again requests the entire page from the server. Here's how the AJAX-enabled process works:

1. Through client-side JavaScript, the AJAX request uses the browser's XMLHttpRequest object to request only the changed data for one or more page elements that need to be updated in response to the user's action. As part of the server request, AJAX provides a callback function to execute when the response is returned from the server.
2. The server receives and processes the request asynchronously, meaning that other code on the page is not blocked, and the page remains responsive to the user. The response is formatted as XML.
3. Upon receipt of the XML response, XMLHttpRequest executes the callback function. This callback function parses the response data and uses it to update the appropriate page elements.

Figure 2 illustrates these steps.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

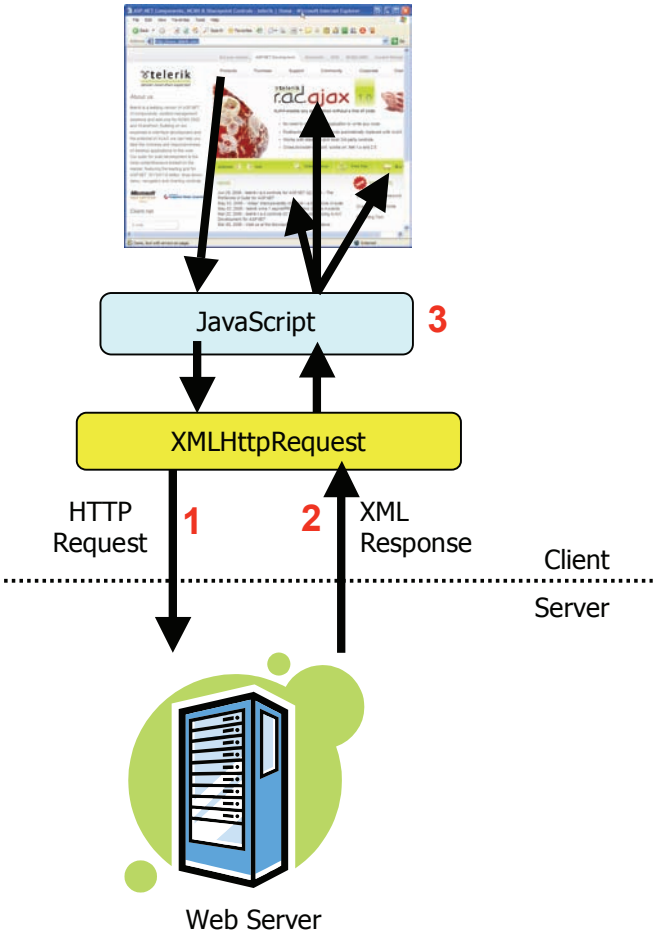

Figure 2. Anatomy of an AJAX request.

AJAX offers a fundamental paradigm shift in how the Web works. Once the Web server delivers the initial page, the HTML and content live at the client instead of the server. Each AJAX request only delivers raw data formatted as XML that contains updated information for only a portion of the page. The client uses that data to update the page that is cached on the client. In short, **AJAX is a universal framework for creating highly interactive, cross-browser applications.**

AJAX-enabled applications offer several key benefits:

- **No page refreshes.** Using AJAX eliminates the flickering of the browser window that commonly accompanies a full page postback, giving the application a more responsive feel to the user.
- **Much better performance.** Non-trivial Web pages can incorporate complicated logic, perhaps requiring several complex database queries. If only a single part of the page needs to be updated, the response to user action can be much faster. Because the server has to transmit less data, using AJAX will reduce network traffic.
- **Desktop-like behavior.** AJAX makes Web applications feel and appear much more like desktop applications, which don't do full refreshes each time the user interacts with the application.

Although at first glance AJAX sounds like a panacea that can bring Web applications all the benefits of their desktop brethren, up to now implementing AJAX in a Web page has had several drawbacks:

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

- AJAX can **require advanced interactivity** between various page elements. This often means that you must write custom JavaScript functions for each pair of elements that interact through AJAX. Client-side code on complicated pages can quickly turn into a quagmire of unmaintainable, untestable code.
- AJAX often results in **broken page lifecycles**. Most Web development platforms, including ASP.NET, have a complex, ordered set of events that occur for every page request in order to build the page returned to the browser. By updating only parts of the page, the state of various elements can easily get out of sync, causing the application to break in mysterious ways that are hard to debug.
- **Form data is not sent to the server** as part of a normal AJAX request. As a result, sometimes the server doesn't have the full set of information that it needs to process the request, resulting in an erroneous response sent to the client. This is another great way to break an application unless you write the code to send all required page data with each request.
- AJAX used on ASP.NET pages can result in **corrupted view state**. View state in an ASP.NET page is contained in a hidden form field and sent to the server with every postback. It is not automatically transmitted with an AJAX request. In addition, the server may update the view state incorrectly since it doesn't have the full set of state information for the page.
- AJAX has a **steep learning curve**. This new paradigm for building Web pages involves writing lots of client-side JavaScript to perform dynamic HTML actions. Developers need to know XML, JavaScript, DHTML, XMLHttpRequest, and the HTML document object model, at least until the tools catch up to the technology.

In short, AJAX offers very promising technology that can make Web applications as responsive as desktop applications. But implementing AJAX in a page means that developers must overcome a number of limitations, often leading to complicated, error-prone, masses of code.

But **t**elerik offers a solution.

The telerik r.a.d.ajax Framework

Even though the technologies that support AJAX have been available in browsers for years, AJAX as a formal design pattern is new and evolving as developers explore the most effective ways to put it to use. There are at least five common ways to implement AJAX in Web applications:

- **Manually.** Some early AJAX solutions required developers to implement the complete AJAX infrastructure by hand. Developers had to write complex JavaScript on the client to respond to user actions that cause an AJAX request to the server and then handle the response using a callback function. Manual AJAX implementations also require server code to receive an AJAX request and produce the chunk of XML that provides the updated data used to update portions of the Web page. As a page grows more complex, it is nearly impossible to manage and debug the complexity. And you have to start over for each new page.
- **Internally.** A Web page component can implement AJAX internally for its own use to communicate with the server and receive updates. This implementation allows little to no interaction with other elements on the page, making this a kind of proprietary solution. Data-intensive controls, such as r.a.d.**grid** use AJAX this way.
- **AJAX widgets.** An AJAX widget is typically a small control that adds slick user interface features with transitions and other responsive behaviors to implement limited AJAX features on a Web Form. Microsoft's ASP.NET "ATLAS" takes this approach.
- **AJAX-enhanced controls.** An AJAX-enhanced control can inherit from a regular base control, with AJAX features implemented in the inherited control. This can be an effective way to implement AJAX for specific controls, but requires replacing all instances of the base control with the new AJAX version.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

- **AJAX framework.** An AJAX framework hooks into the page lifecycle and state features of the Web platform to AJAX-enable any existing application. This is the most comprehensive way to implement AJAX, generally requiring very few changes to the existing application.

telerik r.a.d.**ajax** takes the last approach, providing a comprehensive framework for implementing AJAX in existing applications. r.a.d.**ajax** expands on AJAX to become a universal framework for creating highly interactive, cross-browser ASP.NET applications without writing a single line of code. Developers who use r.a.d.**ajax** don't need to write and debug complex, client-side JavaScript code.

r.a.d.**ajax** consists of several controls used to support various scenarios and types of interactions between controls. This paper will focus on just two of the controls: the AJAX Manager and Loading Panel controls. This paper will briefly describe these controls and then look at an example that uses both controls.

AJAX Manager
You can use the AJAX Manager control at the Web Form level to convert full page postbacks, replacing them with AJAX calls. By making AJAX calls instead of postbacks, AJAX can update portions of the page. AJAX Manager provides centralized management of the AJAX behavior on the page, preserving the page lifecycle and view state, as well as handling validation events. It works on both .NET 1.x and 2.0 with recent versions of virtually all the major browsers.

AJAX Manager keeps track of the controls on a page that are capable of causing a postback and lets you define AJAX relationships that determine what user actions on one control cause other controls to be updated. AJAX Manager is not visible at runtime but implements the r.a.d.**ajax** Property Builder dialog box to let you define the AJAX relationships. It can handle complex scenarios and unlimited relationships between any and all of the controls on the page.

> **NOTE:** An AJAX relationship can be between one control and several others, or between one control and itself. An example of a recursive relationship is for a data grid view. You can define an AJAX relationship for the grid itself, for example, so that when you sort a column, only the grid itself is updated and there is no page refresh.

AJAX Manager makes it easy to define one or more AJAX relationships between controls. An AJAX relationship is between one *initiator control* and one or more *update controls*. An initiator control is an element on a Web Form that in the absence of AJAX causes a postback to the server for a complete page refresh. The controls that are updated by the initiator control are the update controls. In the sample application used later in this paper, you'll see how to use a calendar control to initiate updates to a grid with a list of e-mails received on the selected date and the body of the e-mail message contained in a regular element. Thus an AJAX relationship is defined between the calendar and both the e-mail list and message body elements.

AJAX Loading Panel
When you use AJAX Manager to define AJAX relationships between controls, a control being updated retains its old appearance and data until the new data arrives from the asynchronous call back to the server. This may be appropriate for some applications, but it doesn't give the user any feedback that something is happening. This lack of feedback often leads to problems when the user stops execution, hits the Back button on the browser, or attempts to refresh the page.

The AjaxLoadingPanel control lets you display text such as "Loading…" and an animated graphic to make it clear to the user that the request is being processed. By associating an AjaxLoadingPanel to each control on the page that can be updated using AJAX, the browser automatically displays the loading message when a control is waiting for data. r.a.d.**ajax** ships

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

with eight different animated .gif images or you can use your own that better fits the overall site design.

The AjaxLoadingPanel control is implemented as a template control. By default when you add it to a Web Form using the Visual Studio Web Page Designer, it contains a single <img> tag to display the animated .gif you select. You can use whatever controls in the template you want, such as a label, in addition to the image.

Enabling AJAX Using the r.a.d.ajax Manger Control

*Watch a video of how to use r.a.d.**ajax** here:*
http://www.telerik.com/public/presentation/ajax/AJAX-Manager1.html

Enabling an existing ASP.NET application to use AJAX is remarkably easy. For any page, it requires dropping an RadAjaxManager control on the page and using its Property Builder to define the AJAX relationships between page elements. That's all it takes!

To demonstrate how to enable AJAX on an existing page, I'll use a simple page with three controls that simulates an e-mail application, shown in figure 3. Dates with e-mails are highlighted on the Calendar control as read from a Microsoft Access database. The user can select a date to display a list of e-mails in a DataGrid in the middle section of the form, along with the text of the first message in the list in a control on the right. Selecting a different e-mail message displays the new text.

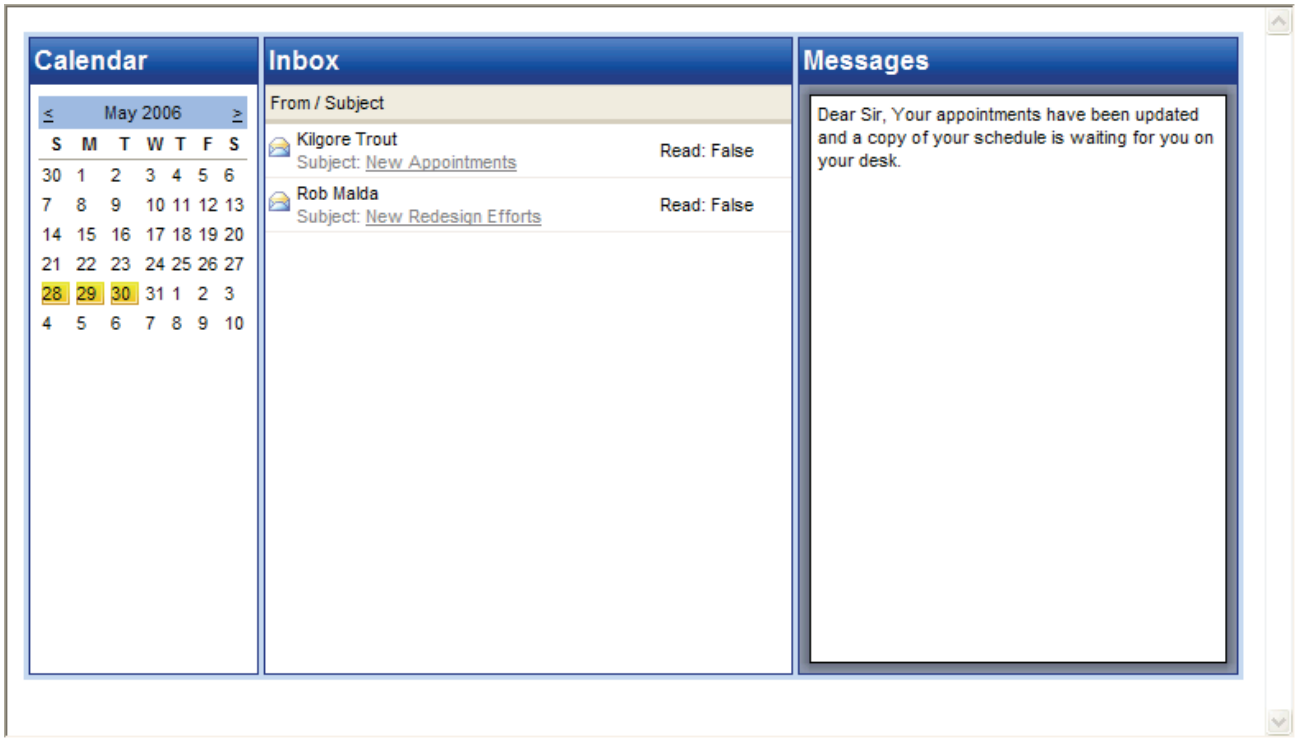

Figure 3. E-mail sample application, which updates various controls via a postback when dates and e-mails are selected.

Each time the user makes a selection in either the calendar or list of e-mails, it causes a postback to the server that causes the browser to download the entire page. Selecting a date on the calendar requires updating two elements, a data grid and . Selecting an e-mail updates just the element. There really is no need to refresh the entire page.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Here is how you can enable AJAX for the page using r.a.d.**ajax** so that only the controls that change have to be updated.

1. Open the application in Visual Studio and load the Web Page Designer for AjaxManager.aspx.
2. From the Visual Studio Toolbox, drag the RadAjaxManager control to the form, shown in figure 4. Since this control has no run time appearance, it doesn't matter where you put it. You only need to make this single modification to the existing page and its design.

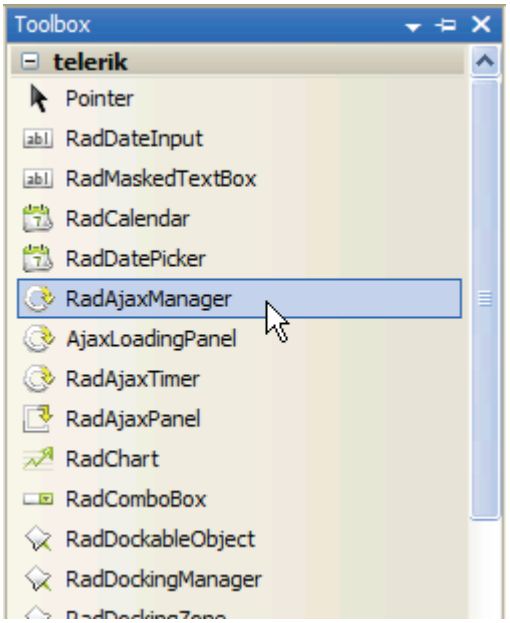

Figure 4. The AJAX Manager control is installed to the Visual Studio Toolbox when you install r.a.d.ajax.

3. Use the AJAX Manager's task list, shown in figure 5, to open the r.a.d.**ajax** Property Builder dialog box. It opens with the initiator controls list populated with all of the page elements.

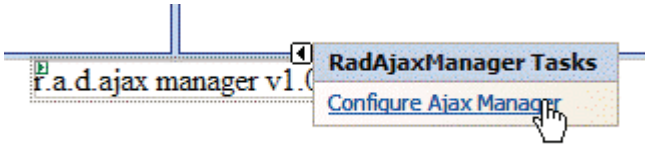

Figure 5. r.a.d.ajax is fully integrated into Visual Studio, complete with a visual configuration tool available from the control's task list.

4. Begin configuring the AJAX relationships by selecting an initiator control from the pane on the left. An initiator control causes other controls to update when a user makes a selection on it. For example, the calendar is an initiator control because when the user selects a date, the e-mail data grid and message element need to be updated.

   Select the checkbox for the Calendar1 control to mark it as an initiator control. Then select the checkboxes for the EmailGrid and MessageBody controls in the center panel of the Property Builder to indicate that these controls are updated in response

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

to a user selection on the calendar. Figure 6 shows the r.a.d.**ajax** Property Builder at this point. For the moment you don't need to make a selection in the Additional Properties panel.

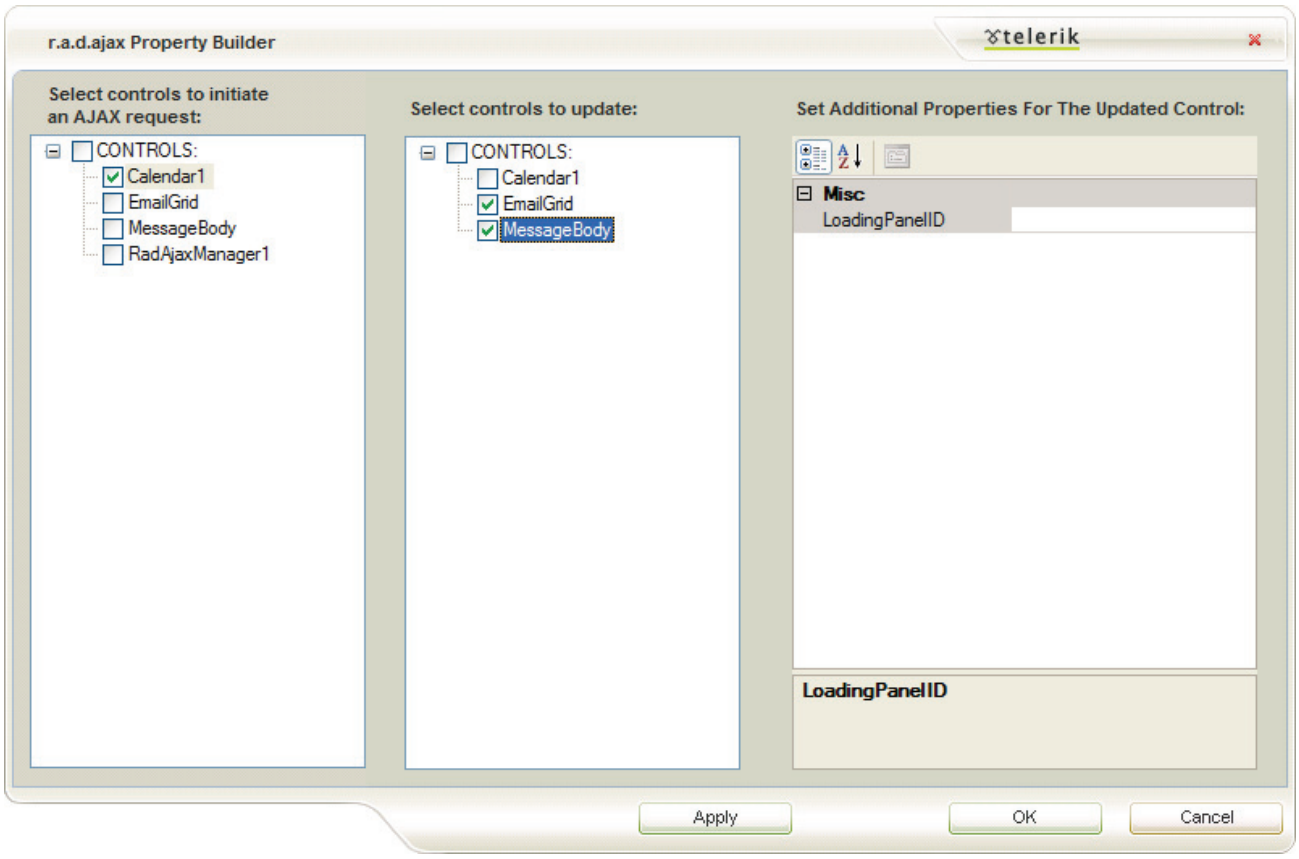

Figure 6, r.a.d.ajax Property Builder with the calendar configured as an initiator control and EmailGrid and MessageBody configured as update controls.

5. Repeat the process to make the EmailGrid an initiator control that updates the MessageBody element. In this case there is only one update control, MessageBody.
6. Click the OK button to close the r.a.d.**ajax** Property Builder and save the changes.
7. Save and run the project.

NOTE: Notice that a single control can be both an initiator and update control. In the sample, EmailGrid is updated when Calendar1 changes, and changes to EmailGrid causes an update to MessageBody. r.a.d.**ajax** lets you model just about any complex interrelationships between controls.

Now when the user makes selections from the calendar or list of e-mails the page uses AJAX to communicate with the server instead of a full page postback. You can see the effect by noticing that when the page first loads, Internet Explorer displays the page loading progress indicator (shown in figure 7) because it is loading (refreshing) the entire page. But in the AJAX-enabled version of the application, only a portion of the page is refreshed as the user selects dates and e-mails, so the progress indicator doesn't appear.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Figure 7. Progress indicator in Internet Explore, showing that the page is being loaded in its entirety.

That's all it takes to enable AJAX on an existing page with r.a.d.**ajax**. You don't need to write complex JavaScript on the client or complicated server-side code. Using this same approach, you can AJAX-enable much more complex applications so that it works entirely with AJAX, similar to how Outlook Web Access works.

Status Feedback

Even when using AJAX to retrieve updates to only a portion of a page, it can take some time for the server to respond. This is particularly true when the user has a slow Internet connection and when the update involves complicated database queries. As the sample application stands now, it displays the old data in the control until the new data arrives and is processed on the client, giving the user no feedback that anything is happening. r.a.d.**ajax** has another control, the Loading Panel, that can provide such feedback. This control enriches the user experience and keeps users from repeatedly clicking the refresh button when nothing seems to be happening!

I'll show you how to add two Loading Panels to the sample application.

1. Stop the application if it is running, and return to the Visual Studio Web Page Designer.
2. Drag two AjaxLoadingPanels from the Visual Studio Toolbox to the Web Form. It doesn't matter where you put them since at run time they will appear in place of the controls being updated.
3. Next you'll associate each Loading Panel with an update control on the page. Open the AJAX Manager's Property Builder. To specify the Loading Panel for the EmailGrid control, highlight EmailGrid in the center pane (being careful not to clear the checkbox). In the Additional Properties pane on the right, select the LoadingPanelID property and select AjaxLoadingPanel1 from the list. Do the same for the MessageBody control to link it to AjaxLoadingPanel2. The Property Builder should look like figure 8 for the MessageBody control.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

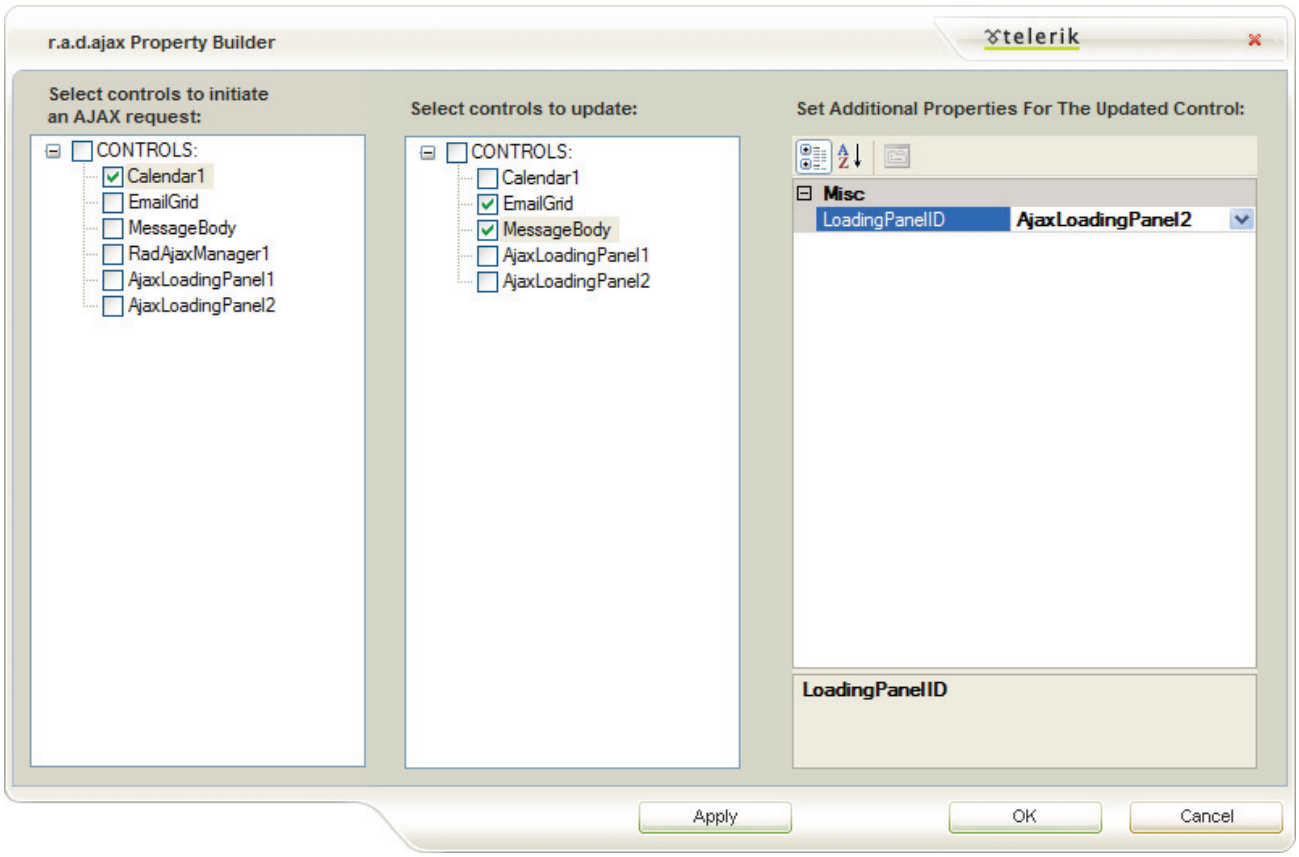

Figure 8. The LoadingPanelID property of an update control links an AJAX LoadingPanel control to display feedback to the user that something is happening.

4. Click OK to save the changes and close the Property Builder dialog box.
5. Save and run the application.

By using Loading Panels in the application, the contents of the update controls are immediately cleared and the user will see a *Loading…* message when they pick a date on the calendar. This makes the application much more user friendly, preventing the perception that it has frozen when it takes some time to get results back from the server.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

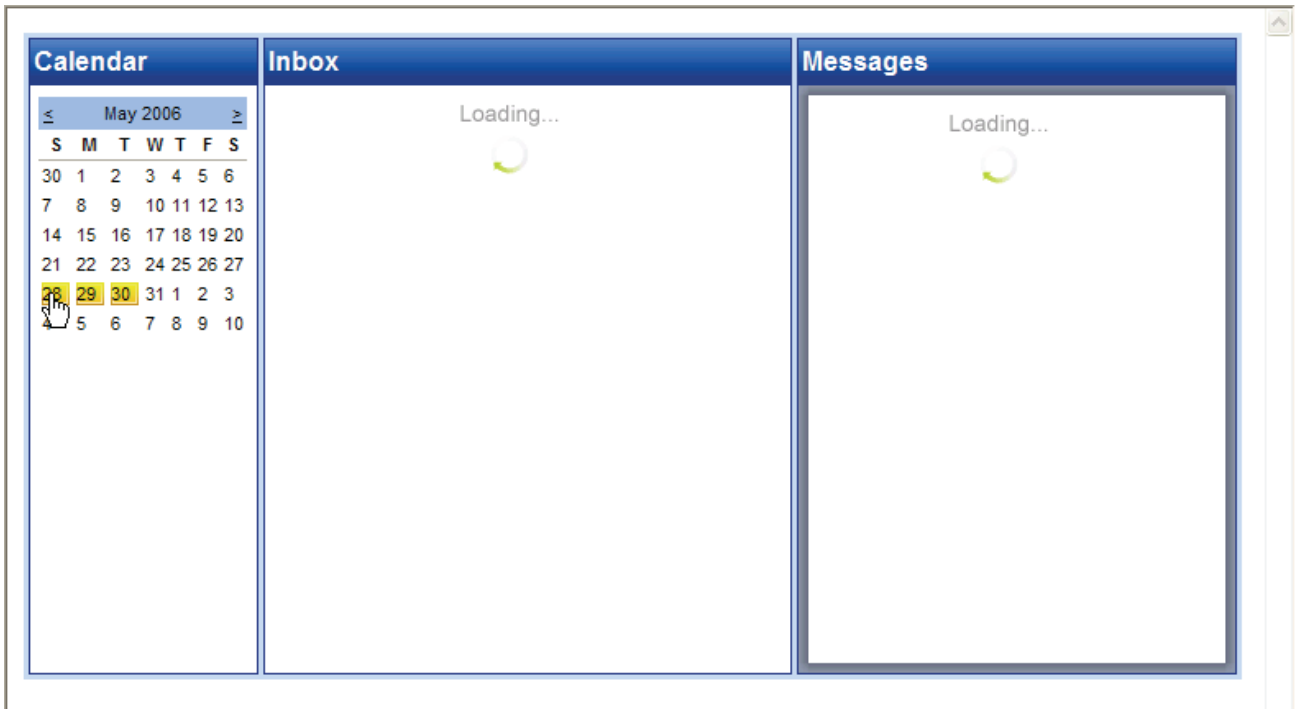

Figure 9. Using AJAX Loading Panels to let the user know that the page is being updated. The image is animated, giving the user something pretty to look at while waiting.

> **NOTE:** The Loading Panel control has several properties you can set to affect its run-time behavior, such as to change the text and image displayed, set an initial delay before display to avoid flashing when results are returned quickly, and specify the horizontal display position. You can also use the style attribute of the <rada:AjaxLoadingPanel> element to precisely position the panel on the page. The panels shown in figure 9 use a style attribute value of "padding-top:20px;" to move the panel down enough to not overlap the Inbox and Messages headers.

How r.a.d.ajax Works

It is often said that new technology is often magical because of what it does and how simple it seems to work. r.a.d.**ajax** is certainly magical, taking the complex technology of AJAX and making it almost trivial to enable AJAX on an ASP.NET page. But there really is no magic involved, just well-designed integration with ASP.NET and the browser.

The AJAX Manager defines and controls AJAX relationships between page elements: the initiator and update controls. It automatically replaces all postback requests with AJAX requests for the controls you define. This involves injecting a small amount of custom JavaScript into the page to turn ASP.NET postbacks into AJAX requests.

You can see a bit of the magic by examining the source behind an AJAX-enabled page. For example, the HTML below is the postback version of the of the sample application before it was enabled with AJAX, as generated by ASP.NET. This code displays the 28th day of May 2006 on the calendar, one of the days highlighted as having e-mails available (reformatted and elided to be more readable).

```
<td class="CalendarDay" style="width:14%;">
```

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

```
        <a href="javascript:__doPostBack('Calendar1','2339')" ...>28</a>
</td>
```

Below is the same element in the version of the page using the RadAjaxManager control:

```
<td class="CalendarDay" style="width:14%;">
        <a href="javascript:window['RadAjaxManager1'].AsyncRequest('Calendar1','2339')"
...>28</a>
</td>
```

Notice that r.a.d.**ajax** has replaced the __doPostBack call with an AsyncRequest call. This is the reason that AJAX works for this element. r.a.d.**ajax** makes this kind of replacement only for AJAX initiator controls, not for any other controls that do regular postbacks to the server. r.a.d.**ajax** does make other changes to the page that make the whole thing work. The r.a.d.**ajax** documentation describes the changes in detail, and the documentations provides information about how you can customize some of its behavior. The AJAX Manager also exposes server-side events that you can hook into in order to customize the response, letting you handle the gnarliest page logic.

r.a.d.**ajax** handles all of the necessary changes on the page for you, based on how you configure the AJAX relationships between controls. It also provides a Visual Studio designer with a convenient visual Property Builder to make it easy to configure. Even without the Property Builder, the HTML code is straightforward. Below is the complete <rada:RadAjaxManager> element that implements all the features in the .aspx page.

```
<rada:RadAjaxManager ID="RadAjaxManager1" runat="server">
  <AjaxSettings>
    <rada:AjaxSetting AjaxControlID="Calendar1">
      <UpdatedControls>
        <rada:AjaxUpdatedControl ControlID="EmailGrid"
          LoadingPanelID="AjaxLoadingPanel1" />
        <rada:AjaxUpdatedControl ControlID="MessageBody"
          LoadingPanelID="AjaxLoadingPanel2" />
      </UpdatedControls>
    </rada:AjaxSetting>
    <rada:AjaxSetting AjaxControlID="EmailGrid">
      <UpdatedControls>
        <rada:AjaxUpdatedControl ControlID="MessageBody"
          LoadingPanelID="AjaxLoadingPanel2" />
      </UpdatedControls>
    </rada:AjaxSetting>
  </AjaxSettings>
</rada:RadAjaxManager>
```

As you can see, it would be quite easy to write this code yourself. The magic is revealed.

Best of all, r.a.d.**ajax** takes care of all the infrastructure to hook into ASP.NET page processing on both the client and server to handle the AJAX requests, with no need to modify your application code or logic.

Key Benefits of r.a.d.ajax

r.a.d.**ajax** offers all of the benefits of AJAX with virtually none of the drawbacks.

The major benefit of r.a.d.**ajax** is that you don't have to learn the intricacies of AJAX to get its benefits. r.a.d.**ajax** does it all for you. You don't need to write a lot of messy client-side

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

JavaScript or run anything on the server, other than installing the r.a.d.**ajax** controls with your application. Deploying an application built with r.a.d.**ajax** requires simply copying the RadControls/Ajax subdirectory to the main application directory. The main contents of this directory are two JavaScript files and several .gif images that control the appearance of AJAX elements on the page.

To summarize, there are numerous benefits to using r.a.d.**ajax** to enable AJAX in your new and existing ASP.NET applications:

- You can AJAX-enable any new or existing application. There is no need to change existing server-side code in an existing application.
- You don't have to write cumbersome client-side JavaScript. In fact, you don't have to write even a single line of code on either the client or server.
- Form values are persisted because they are sent to the server for processing during the AJAX request, so that r.a.d.**ajax** can handle complex page logic.
- r.a.d.**ajax** is easy to learn and use. You don't have to learn the intricacies of AJAX.
- Any Web page component you specify as an initiator control changes from using a postback into a component that works with AJAX.
- r.a.d.**ajax** works with all standard ASP.NET controls and most third-party controls.
- ASP.NET page lifecycle, view state, and validation events are all preserved and handled normally.
- For new applications, you can easily enable AJAX without building pages in some specific way to prepare them for AJAX.

You can download a trial version of r.a.d.**ajax** at http://www.telerik.com/asp-net-controls/r.a.d.ajax/download-trial.aspx.

Resources

Videos that Demonstrate r.a.d.ajax

- You can see an overview of r.a.d.**ajax**, its components, and how to use them, including demos that show how to enable AJAX for a calendar control to eliminate full-page postbacks when selecting dates and how to define AJAX relationships to update multiple controls, at: http://www.telerik.com/public/presentation/ajax/ajax.html

Web Pages that Use AJAX

These sites use AJAX and can give you an idea of the possibilities.

- Microsoft Outlook Web Access was probably the first implementation of AJAX, although it didn't use the name.
- Google has been a major innovator with AJAX. Check out Google Maps at http://maps.google.com/, Google Groups at http://groups.google.com/, Google Spreadsheets at http://oqb.spreadsheets.google.com, and Google GMail at http://mail.google.com/mail/.
- Amazon's A9 site lets you search various parts of the Web and change options without refreshing the page. http://a9.com/
- Writely is a Web-based word processor that uses AJAX. Google purchased Writely to add to their growing stable of office applications. http://www.writely.com