1

Introduction to ASP.NET AJAX

Over the years, we developers have seen many changes in terms of how development occurs. We have gone from terminal-based programming to PC-based programming to Windows-based programming and to the Web. Now we are on the verge of another programming revolution — one that will bring about more interactive user interfaces to web applications. This programming revolution is brought to developers courtesy of a set of technologies that are generally known as AJAX (Asynchronous JavaScript And XML). No longer will users see the annoying flash with the clicking of a button to submit data. No longer will users lose the context of where they are located and be thrown back up to the top of a page. With AJAX, developers can build applications that step out of the traditional postback model of the Web, provide an improved user interface to users, and allow developers to develop applications that are much friendlier to use.

This chapter looks at the following:

- ASP.NET development and how it led to AJAX
- What AJAX is and a high-level overview of some of its base technologies
- The advantages of AJAX
- What ASP.NET AJAX is
- Some things that it might not make sense to do with AJAX

Development Trends

If you have been developing for a while, like us old guys, you have gone through several iterations of development — from terminals connected to mainframes and mini-computers to personal computers and then to client-server development. Client-server development allowed for the minimization of back-end resources, network resources, and the front-end PC by sending only the necessary data between back end and front end. Intelligent client-server development allowed for building applications that were responsive to the user and made efficient use of network and back-end resources. As the web development methodology took off in the late 1990s, we unfortunately

returned to the terminal-style development. In this methodology, any major operation between the client and server requires that all data be sent in what is called a *round trip*. With a round trip, all data from the form is sent from the client to the web server. The web server processes data and then sends it back to the client. The result of a round trip is that a lot of data is sent back and forth between the client and server. Given the circumstances, these operations may result in more data transfer and CPU utilization than a web application and server can really tolerate.

ASP.NET Development

ASP.NET is a set of web development technologies produced by Microsoft that is used to build dynamic web sites, web applications, and XML-based web applications. ASP.NET is a part of the .NET framework and allows for developers to build applications in multiple languages, such as Visual Basic .NET, JScript .NET, and C#.

Design Methodology

ASP.NET attempts to make the web development methodology like the GUI development methodology by allowing developers to build pages made up of controls similar to a GUI. Server controls in ASP.NET function similarly to GUI controls in other environments. Buttons, text boxes, labels, and datagrids have properties that can be modified and expose events that may be processed. The ASP.NET server controls know how to display their content in an HTML page just like GUI-based user controls know how to display themselves in their GUI environment. An added benefit of ASP.NET is that the properties and methods of the web server controls are similar, and in some cases the same as the comparable controls in the Windows GUI/Windows Forms environment.

Problems ASP.NET Solves

Microsoft has released various web application development methodologies since the shipment of IIS in Windows. Why do developers need ASP.NET? What problems does ASP.NET solve that the previous development methodologies did not solve?

Microsoft's first popular web development technology was the Internet Database Connector (IDC). The IDC methodology provided only database access; it did not provide access to any other resource programmatically. For example, there was no way to programmatically send email or do other non-database operations. Another issue was that it seemed to be somewhat different from the traditional programming languages that most developers were used to (Visual Basic and C++ being two popular ones). Along with this problem was the fact that the development experience was not very attractive within Microsoft FrontPage. Along with the development experience, IDC had no debugging experience worth mentioning. Overall, IDC was nothing more than a stopgap measure to get to an improved environment.

The next web development methodology from Microsoft was Active Server Pages (ASP). ASP was a scripting environment that allowed developers to work with a Visual Basic–like or JavaScript-type environment. Unfortunately, this type of environment came with several problems:

- **Prevalence of spaghetti code** — ASP code does not provide a structured development environment, often contributing to the creation of twisted and tangled "spaghetti code." ASP code is literally a file with some basic configuration information at the top of every page. Each page is executed from the top of the page to the bottom of the page. Although it is possible to use Component Object Model (COM) objects to eliminate some of the spaghetti code, this introduces more complexity in the form of another development tool.

- **Lack of code separation** — The code tends to be intermixed with display code. Intermixing the code and the display logic requires that the tools developers and designers use work well together. This was often not the case. For example, it was well known that various visual development tools could take a properly running ASP page, rearrange some of the code, and render the ASP page broken.
- **Lack of code reusability** — There is very little ability to reuse code within the ASP environment. Code reusability in classic ASP is a function of providing logic in the form of COM objects, as opposed to something within the ASP environment.
- **Lack of debugging support** — Debugging an ASP application typically involves the use of `Response.Write`. This is in sharp contrast to an integrated development environment (IDE) developed within a GUI environment.
- **Problems of COM** — ASP is based on the Component Object Model and suffers from many of the problems associated with COM. There were two major problems with COM:
  - The first was that updating COM objects tended to overwrite one object with the new one. This could be problematic if a programming method call changed or any other new behavior was introduced.
  - The second major problem with COM was that it was a binary standard. This binary standard was based on a 32-bit programming model. As a result, COM objects would not scale up to run natively within an environment that was an Intel-based, 64-bit environment. Although this might not have been a big deal in the early to middle 1990s when COM was designed and built, by the early 2000s and the introduction of inexpensive 64-bit systems, this was seen as a possible bottleneck.
- **Problems with being interpreted** — ASP is interpreted. Each time an ASP file is loaded, the ASP environment parses the ASP file, compiles the code, and then executes the file. This process is repeated on each call to an ASP file. The result is wasted processing on the server.
- **Presence of the state machine** — ASP applications typically have a state machine at the top of every ASP page that processes the state of the user and then displays code. (In software code, a state machine is a section of code that depends on both its direct inputs and inputs made during previous calls.) Given that most client-side applications are built based on events, which is a similar concept to a state machine, this is an unfamiliar way to develop for those not well versed in ASP.

After getting feedback from developers, Microsoft developed ASP.NET, which greatly simplifies the web development methodology:

- Developers no longer need to worry about processing state. With ASP.NET, actions are performed within a series of events that provide state machine-like functionality.
- With the use of a code-behind/beside model, code is separated from display. By separating code and display files, there is less of a chance of designer and developer tools interfering with each other.
- A single development tool may be used for building the application and business logic. By having a single integrated development suite, developers are able to more easily interact with the application logic. This results in more code reuse and fewer errors.
- With the Visual Studio 2005 IDE, ASP.NET supports many methods to debug and track a running ASP.NET application.

- Because ASP.NET is based on the common language runtime (CLR) and .NET, ASP.NET does not suffer from the versioning problems of COM. The .NET framework allows for multiple versions of components to be on a system without their interacting with each other.
- ASP.NET is compiled. The first time that a file is loaded, it is compiled and then processed. The compiled file is then saved into a temporary directory. Subsequent calls to the ASP.NET file are processed from the compiled file. The execution of the compiled file on requests is faster than the interpreted environment of classic ASP.

All in all, ASP.NET is a dramatic improvement over ASP and has become widely accepted in the development community.

So, What's the Problem?

Based on what you have just read regarding ASP.NET, it may sound really good to you. You may be asking yourself, "Why is there a need for something else? What's the problem?"

The truth is that ASP.NET has several issues that need to be addressed:

- **Round trips** — The server events in ASP.NET require round trips to the server to process these events. These round trips result in all form elements being sent between client and server as well as images and other data files being sent back to the client from the server. Though some web browsers will cache images, there can still be significant data transfer.
- **Speed/network data transfer** — Because of the `ViewState` hidden form element, the amount of data that is transferred during a postback is relatively large. The more data and controls on the page, the larger the `ViewState` will be and the more data that must be processed on the server and transmitted back to the client.
- **Waiting on the result** — When a user clicks a button or some other visual element that posts back data to the server, the user must wait for a full round trip to complete. This takes time when the processing is done on the server and all the data, including images and `ViewState`, are returned to the client. During that time, even if the user attempts to do something with the user interface, that action is not actually processed on the client.
- **User context** — Unless an application is able to properly use the `SMARTNAVIGATION` feature of ASP.NET, the user is redirected to the top of a page by default on a postback. Though there are ways around this issue, this is the default behavior.
- **Processing** — The number of server round trips, amount of data that is transferred, and the `ViewState` element's size result in processing on the server that is not really necessary.

Users typically do something, data is sent to the server, the web server processes it, and the result is finally sent to back to the user. While the server is processing the data, the user interface is "locked" so that additional operations don't happen until a result is returned to the user.

Improving the User Experience

Based on the preceding issues, several options are available for improving the user experience:

- **Java** — Java applets are cross-platform applications. While being used as a cross-platform mechanism to display data and improve the user experience, Java development on the client has not been accepted with open arms by the development community and is primarily used for user interface gee-whiz features as opposed to improving the experience of the user application. (As a side note, Java has been widely accepted for building server-side applications.)
- **XML-based languages** — XML User Interface Language (XUL) and Extensible Application Markup Language (XAML) are two of several languages that can provide an improved user experience. The problem with XUL is that it has been used only in the Mozilla/Firefox line of browsers. Silverlight (formerly WPF/e), an associated product, is an interpreter for a subset of XAML. Currently, there is support for Silverlight on Windows and the Apple Macintosh.
- **Flash** — Although Flash has been used and there are cross-platform versions, the product has been used only in the area of graphic UI needs and has not been accepted by the development community as a whole for building line of business applications. Recently, Adobe has released a pre-release version of an Internet technology referred to as Apollo. Apollo is a runtime that allows web skillsets to be used to develop rich desktop applications.
- **AJAX** — AJAX is a set of client technologies that provide for asynchronous communication between the user interface and the web server, along with fairly easy integration with existing technologies.

Given the amount of recent discussion among developers regarding AJAX, it appears that AJAX has the greatest chance among these technologies of gaining market acceptance.

Current Drivers

Interest in web-based development has grown over the past few years. With that interest, Microsoft has gone from classical ASP to ASP.NET development. ASP.NET development has grown to the point that it is the most popular development platform for web-based applications. Even with its popularity, it has to continually improve or it will get left in the dust of a more modern technology.

Over the past few years, building client-side web-based applications has grown in popularity. Users have liked the applications because of the increased client-side functionality, such as keeping a common user context during a "post" to the server and drag-and-drop features common to typical client applications. This functionality was popularized by several applications from Google, including Gmail, Google Suggest, and Google Maps.

In February 2005, this functionality got the name Asynchronous JavaScript And XML (AJAX) thanks to an essay by Jesse James Garrett. At about this time, several .NET libraries started to show up. These libraries hid many of the complexities of interfacing with web services and allowed developers to concentrate on the application as opposed to creating the plumbing to talk to the web services.

ASP.NET needs to add this functionality. The question becomes, how does one add client-side functionality to a development methodology that is mostly a server-side technology?

From a network standpoint, these applications are more efficient because they communicate back only the necessary pieces of information and get back only the necessary updates from the server. From a web server standpoint, these applications tend to use less CPU on the server. As a result, these types of applications are highly desirable.

What Is AJAX?

So into this development environment comes a set of technologies that are collectively referred to as AJAX. If you are an "old guy" developer like me, then AJAX represents a similar concept to the client-server development mentioned earlier in the chapter. With client-server development, the amount of data transferred is minimized over a terminal application by transferring only the necessary data back and forth. Similarly, with AJAX, only the necessary data is transferred back and forth between the client and the web server. This minimizes the network utilization and processing on the client.

Advantages of AJAX

The advantages of AJAX over classical web-based applications include:

- **Asynchronous calls** — AJAX allows for the ability to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.
- **Minimal data transfer** — By not performing a full postback and sending all form data to the server, network utilization is minimized and quicker operations occur. In sites and locations with restricted pipes for data transfer, this can greatly improve network performance.
- **Limited processing on the server** — Along with the fact that only the necessary data is sent to the server, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the `ViewState`, send images back to the client, or send a full page back to the client.
- **Responsiveness** — Because AJAX applications are asynchronous on the client, they are perceived to be very responsive.
- **Context** — With a full postback, users may lose the context of where they are. Users may be at the bottom of a page, hit the Submit button, and be redirected back to the top of the page. With AJAX there is no full postback. Clicking the Submit button in an application that uses AJAX will allow users to maintain their location. The user state is maintained, and the users are no longer required to scroll down to the location they were at before clicking Submit.

History of AJAX

For all its perceived newness and sexiness, the technologies that make up AJAX are really not new. The ability to communicate back to the server through a hidden frame without posting the main page back to the server has been around for a long time. Communication between client and server has been available — back to the release of Internet Explorer's ability to script ActiveX controls on the client browser and to the MSXML component, both of which date back into the late 1990s. Personally, I saw the first formal usage of

client script and MSXML in 2003. The problem with the technology at that time was the need to manually create the necessary client-side JavaScript. In 2003, there was too much code overall that had to be written and too much custom code that had to be written to get this to work. Only since the second half of 2005 have client-side libraries and server-side support for ASP.NET started to make their presence felt and been used significantly.

The mainstream development community has only recently started using the technique. The release of Google's Suggest and Maps are what really opened the eyes of the users to the development technologies. These applications sent a shockwave through the development community.

Technologies That Make Up AJAX

AJAX is a general umbrella term. AJAX itself stands for Asynchronous JavaScript And XML. The term was coined by Jesse James Garret of Adaptive Path in an essay published in February 2005 (`http://www.adaptivepath.com/publications/essays/archives/000385.php`) and was quickly accepted by the development community.

Based on this general umbrella term, take a look at the specific items that make up AJAX:

- **XMLHttpRequest** — `XMLHttpRequest` allows the browser to communicate to a back-end server. This object allows for the browser to talk to the server without requiring a postback of the entire web page. With Internet Explorer 5 and 6, this capability is provided by the MSXML ActiveX component. With the Mozilla Firefox, IE 7, and other web browsers, this capability is provided by an object literally called `XmlHttpRequest`. The `XmlHttpRequest` object is modeled after the MSXML component and defined by the XMLHttpRequest standard from the W3C. The ASP.NET 2.0 AJAX client-side JavaScript libraries hide the differences between the various browsers.
- **JavaScript** — JavaScript provides the capability to communicate with the back-end server. The version of JavaScript must be version 1.5 or later. Though JavaScript is not specifically required, it is needed from the standpoint that JavaScript is the only client-side scripting environment supported across the major modern web browsers. There are other client script languages; however, these are not supported across all browsers.
- **DHTML/DOM support** — The browser must support the ability to dynamically update form elements, and the ability to do this in a standard way comes through the support for the Document Object Model (DOM). By supporting the DOM, it becomes easy for developers to write a single piece of code that targets multiple browsers.
- **Data transport with XML or JSON** — Using XML allows for the ability to communicate with the web server in a standard mechanism. The default data format with ASP.NET AJAX is JSON.

What Is ASP.NET 2.0 AJAX?

On June 28, 2005, Microsoft announced "ASP.NET 2.0 AJAX." ASP.NET 2.0 AJAX is an AJAX-oriented .NET library that runs on .NET 2.0. Though ASP.NET 2.0 AJAX is an AJAX library and can be used to perform AJAX operations, it is really much more. ASP.NET 2.0 AJAX offers many of the same types of features of the server-side ASP.NET, but it is directed at the client side. Because ASP.NET 2.0 AJAX is fully integrated with ASP.NET, it provides rich integration with the services provided by ASP.NET.

ASP.NET 2.0 AJAX provides the following features (and much more):

- AJAX-style communications between client and server. This communication is over web services.
- Asynchronous communication. All client-to-server communication in the ASP.NET 2.0 AJAX framework is asynchronous.
- A set of server-side controls that enable rich client-side functionality.
- A set of client-side controls and classes that further enable client-side functionality.
- A framework for encapsulating client-logic through the creation of namespaces and classes.
- Cross browser support. Although there is no official matrix of web browsers that ASP.NET 2.0 AJAX supports, the latest versions of Internet Explorer, Firefox, and Safari are supported. In addition, Opera is thought to be supported; however, we have not been able to find an official statement from Microsoft regarding this.

Running ASP.NET AJAX Applications

Unfortunately, not all web browsers ever produced will support ASP.NET AJAX. To run an ASP.NET AJAX application, a web browser must:

- **Be relatively modern** — ASP.NET AJAX applications are not available in all versions of all web browsers. Though Internet Explorer version 6 and later, Firefox version 1.5 and later, and Safari provide support for these applications, older versions may be problematic because of their support for different versions of the other requirements.
- **Support the DOM** — The capability to update form elements on a page based on new data is important. Accessing the controls in a standard way means that writing code that runs over a majority of web browsers is easier than having code that has a large number of `if/then/else` statements that are dependent on the browser version.
- **Support JavaScript** — ASP.NET AJAX requires some amount of actions to occur out on the client. These actions are done using the JavaScript programming language. Because the major web browsers support JavaScript, it makes sense for JavaScript to be used for the client-side programming language.
- **Possibly have ActiveX enabled on the client** — If you are using the Internet Explorer 6 browser while running on Windows, you may have problems if ActiveX is not enabled.

Who's Using AJAX?

Great, now that you have seen that there is this technology called AJAX, are you alone in not having seen or talked about this before? Absolutely not! AJAX has just recently taken off in the second half of 2005 from a mindshare standpoint. As discussions have gone on with counterparts in the development community, many developers are just now looking to what AJAX can do for their applications and ultimately their customers. So, just who is using AJAX publicly?

- **Google Suggest** — Google Suggest features a dynamic drop-down list box that provides possible items to search on along with the approximate number of search results.

- **Google Maps** — The ability to grab a map and zoom around without requiring a postback is just amazing. This app/service took the development world by storm when it came out.
- **Google Gmail** — Google Gmail is a web-based email system.
- **Live.com** — The local.live.com service from Microsoft is actively using the ASP.NET AJAX framework, as is nearly the entire Live.com service. Hotmail, the email service for Live.com, has updated its service and uses AJAX.
- **Outlook web access** — The web interface into Microsoft Exchange 2000 was one of the early AJAX applications.
- **Easy Search Component** — The ASP.NET Easy Search Component provides support for searching a single web site similar to the Google Suggest service.
- **Other component vendors** — Component vendors such as ComponentArt, Dart, and others are providing controls that provide a rich user experience without forcing a full postback.

To go along with the third-party interest, the amount of developer interest is tremendous. For example, one only has to put the word *AJAX* into a blog title to receive an increase in the number of web views. Based on the amount of third-party support and the interest of developers, it is only a matter of time before everyone is doing it.

Currently

At this time, the first version of the ASP.NET 2.0 AJAX framework is an add-on to the existing .NET 2.0. It runs on top of the framework with no changes to the underlying "bits."

ASP.NET 2.0 AJAX falls into four areas:

- **Server-side controls** — Server-side controls generate the appropriate client-side markup and script to perform client-side operations without the need for a postback. These controls provide a fairly easy environment to debug. For example, debugging with the `UpdatePanel` is fairly easy. Besides the `UpdatePanel`, other controls that work similarly are the ASP.NET AJAX Control Toolkit.
- **Client-side classes** — These classes provide additional functionality to the client-side browser. This type of functionality is similar in concept to the base class libraries included in the .NET framework. An example would be the whole `Sys.Net` namespace along with the extensions to the base JavaScript objects.
- **Web services integration** — This functionality allows a developer to use web services as the communication channel between the web browser and the web server without having to understand the differences between the MSXML component in IE and the `XmlHttpRequest` object in Firefox.

Packaging

The packaging of ASP.NET 2.0 AJAX can be fairly confusing. The basics of the packaging are:

- **ASP.NET 2.0 AJAX Extensions 1.0** — The ASP.NET 2.0 AJAX Extensions 1.0, also referred to as the RTM/Core code, is an independent download. This contains the functionality that will receive support from Microsoft in the initial release of the product. The source code is available.

- **ASP.NET AJAX Futures Community Technology Preview (CTP)** — The ASP.NET 2.0 AJAX framework contains a set of functionality that is experimental in nature. This functionality will eventually become integrated with the RTM/Core code. During the initial release, the Futures CTP functionality will be a separate download from the RTM/Core bits. This will not receive specific support from Microsoft beyond community-based support. The CTP bits require that the RTM/Core bits already be installed for the CTP bits to be installed. The CTP is also referred to as *Value-Added Bits*.
- **Microsoft AJAX Library** — The Microsoft AJAX Library is a set of JavaScript client libraries that make up the standard download to a web browser and provide much of the support for AJAX in the client. These libraries will work with a non-IIS server and are available as a separate download. This library is included in the ASP.NET 2.0 AJAX Extensions 1.0 download as well as being available as a separate download.
- **ASP.NET AJAX Control Toolkit** — The AJAX Control Toolkit is a separate download that provides a set of client-side GUI widgets that integrate with the ASP.NET 2.0 AJAX framework. The toolkit is licensed separately from the framework and includes the source code for developers who would like to review the source.

Futures

It has already been announced that ASP.NET 2.0 AJAX will be integrated into the .NET framework and Visual Studio in the Orcas versions of these products. Future versions will undoubtedly have more integration between ASP.NET 2.0 AJAX, ASP.NET, and Visual Studio.

Summary

AJAX provides developers a foundation to build web-based applications with an improved user experience. In this introductory chapter, you have looked at the following:

- Development from a historical perspective
- Web development methodologies
- Some of the features that ASP.NET AJAX provides, such as improved user responsiveness and decreased load on the web server
- Multiple technologies that can improve the user experience
- The general components in the ASP.NET 2.0 AJAX packaging
- The problems ASP.NET AJAX solves, and who is using it

The next chapter looks at creating an ASP.NET AJAX application.