HOUR 1

Getting Started with ASP.NET Ajax

What You’ll Learn in This Hour:

- Introducing Ajax
- Technologies that make up Ajax
- The pros and cons of using Ajax
- ASP.NET Ajax
- Goals of ASP.NET Ajax
- Installing Ajax
- Creating your first Ajax application

The solution to building applications with fast, user-friendly, and responsive user interfaces is here. Yes! Ajax is in. Ajax is an acronym for Asynchronous JavaScript and XML—a technology that can reduce web page postbacks significantly and yield better response times for your web applications. Using Ajax, the hits to the web server are reduced—thus, you have fewer page refreshes. Moreover, you can use Ajax to ensure that a specific portion of the page is refreshed and not the entire page content.

Usage of Ajax provides rich user experience with a responsive user interface, which eventually results in an awesome user experience. The ASP.NET 2.0 Server development platform is now integrated with the client-side libraries that incorporate cross-browser JavaScript and DHTML technologies. ASP.NET Ajax was available as a separate package in ASP.NET 2.0. With ASP.NET 3.5, you have the Ajax framework built in. In other words, you need not download and install the Ajax package separately in your system; you have built-in support for all Ajax features. There have been a lot of improvements to Ajax in ASP.NET 3.5. We discuss these improvements later in this book. In this hour, we take a look at ASP.NET Ajax, its ingredients, and how we can get started with it.
Things You Should Know

To understand the concepts covered in this book, you need a basic understanding of the following:

- JavaScript
- ASP.NET 2.0
- C#

Ajax—A Paradigm Shift

The advent of Ajax has put an end to the arduous struggle of web application development communities worldwide to find a technology that can not only improve response times, but also allow for asynchronous processing. Ajax is a technology with cross-platform, cross-architecture, and even cross-browser support. In fact, Ajax has already become recognized in Microsoft and Sun development communities for building lightning-fast web applications with improved response times, which results in awesome user experiences. Note that Ajax is a technology; it is not specific to ASP.NET or Java. You can use Ajax in both of the preceding technologies. Moreover, you can use Ajax in any web browser, such as IE, Mozilla, Firefox, and so on.

Prior to Ajax, we could register client-side scripts ASP.NET 2.0 Ajax Page level methods. Here is an example:

Page.ClientScript.GetPostBackEventReference(this, String.Empty);

There’s an old proverb that says, “The old order changeth yielding place to new.” With the introduction of Ajax, there has been a paradigm shift—we have moved away from the earlier trend in which we had to force a postback to retrieve data from the server. With Ajax, we can do the same even without a postback to the web server. The result is improved response times and better performance of the application as a whole.

Ajax uses the XMLHttpRequest object, a JavaScript object that can communicate directly with the web server to retrieve data, without the need to reload web pages each time data is requested. We discuss more on XMLHttpRequest object in Hour 3, “Working with the XMLHttpRequest Object.”
Technologies That Make Up Ajax

Ajax is, in itself, a combination of existing technologies. These include the following:

- XMLHttpRequest object
- JavaScript
- DHTML
- DOM
- XML

Let’s briefly discuss each of them.

JavaScript is a scripting language developed initially by Netscape, but having the capability to be used on all known browsers. It is an interpreted language that can be used both on the client and server side as a scripting language. JavaScript can be easily used to interact with the HTML elements, validate user input, and manage your settings, such as the color and background color of different controls on a form. Like any other programming language, JavaScript contains variables, arrays, loops, functions, operators, exception handling in the form of “try” and “catch” statements, and so on. You can place your JavaScript code directly on the same HTML page or even in a separate .js file and link your web page with it.

All the HTML elements in your web page are organized in a Document Object Model, a W3C recommendation that every browser follows. This model describes how all the elements in an HTML page, such as input fields, paragraphs, images, anchors, and so on, are related to the topmost structure: the “document.” This model defines the structure in a tree consisting of all the attributes and methods defined for an object in the document.

Fine, but what is DHTML? DHTML is the acronym for Dynamic Hypertext Markup Language, a technology that you can use to make your web page dynamic with the use of JavaScript. The word “dynamic” implies the capability of the browser to alter the look and style of HTML elements after the document has been loaded onto the browser. This dynamic content can be realized in several ways, either by applying properties to elements or by applying layers to a document.

CSS, or Cascading Style Sheets, are files that store the styles of your web page HTML elements. These files typically have the .css extension. Note that CSS is basically used to provide a customized look and feel to your HTML elements. You can use CSS
files to store the formatting and style information of elements at a common place
and then reuse it in your web forms to facilitate easy maintenance and enforce the
consistency of the look and feel of the user interface elements.

As an example, you can store all the headings in all the web pages of an applica-
tion by defining them as a class in the .css file. Later, if the heading style needs to
be changed, you can do this just in one place—the .css file. The changes would be
reflected across all web pages of your application wherever this class has been used.

The Pros and Cons of Using Ajax

Some of the many benefits of using Ajax in web-based applications include the fol-
lowing:

- Improved user experience
- Asynchronous processing
- Reduced server hits and network load
- Platform and architecture neutrality
- Multibrowser support
- Faster page renders and improved response times

We discuss each of these as we progress through the hours of this book.

The Downsides of Using Ajax

Now let’s discuss the drawbacks of using Ajax or, more precisely, areas where Ajax
can fit and those where it can’t. Although Ajax comes with a lot of advantages,
there are quite a few downsides to using Ajax in your web applications. The major
drawback is its massive usage and dependency on JavaScript. It should be noted
that JavaScript is implemented differently for various browsers, such as Internet
Explorer, Netscape, Mozilla, and so on. This becomes a constraint especially when
you need to make Ajax work across browsers.

Added to this, you do not have support for JavaScript in mobile browsers. So, taking
Ajax’s dependency on JavaScript into consideration, Ajax might not be well suited
for designing mobile applications.

Usage of Ajax makes your web page difficult to debug, increases the code size of
your web page, and makes your web page prone to potential security threats.
Moreover, its usage—and the asynchronous operations thereafter—tend to increase
the load on the web server. When using Ajax, making your application cross-browser compliant is rather difficult (although not impossible, of course), and the Back button of your web browser does not work.

Looking Back in Time

The technologies that make up Ajax are not new; they’ve been around for years. Netscape’s LiveScript (eventually called JavaScript) allowed for asynchronous processing some time ago.

Netscape came up with support for Dynamic XML and Microsoft with the XMLHttpRequest object within the browser that can be used to retrieve data from the server asynchronously. This phenomenon was later called Ajax by Jesse James Garrett of Adaptive Path in early 2005. Ajax was born. However, it was only in the fall of the same year that Ajax made its presence felt within development communities worldwide.

Google led the drive to make Ajax known to these communities by announcing the first public implementation of Ajax in Google Suggest. Because of these efforts, examples of the public use of Ajax can be found worldwide; a few of these examples are as follows:

- Google Maps
- Google Suggest
- GMail
- Live.com

And, the list goes on!

What Is ASP.NET Ajax?

ASP.NET Ajax, formerly known as Atlas, is an extension of ASP.NET 2.0. It allows you to leverage the power of Ajax while developing ASP.NET Ajax web applications. The MSDN states, “ASP.NET Ajax is a set of technologies to add Ajax (Asynchronous JavaScript and XML) support to ASP.NET. It consists of a client-side script framework, server controls, and more. Although Ajax is essentially a client-side technique, most of its real-world deployments call for server-side processing.”
The ASP.NET Ajax architecture has a framework developed for both client side and server side. The client-side framework comes in the form of the Microsoft Ajax Library. This Library includes a collection of client-side libraries that include support for creating client-side components, browser compatibility, managing asynchronous requests, web and application services, different core services in serialization, JavaScript base class extensions, and so on.

The ASP.NET Ajax server components consist of several web server controls and components to handle the flow and the user interface. It also shows the functionality of ASP.NET 2.0 Ajax server extensions, which include support for localization, globalization, debugging, tracing, web services, and application services.

Moreover, ASP.NET Ajax has come up with several server controls—namely, ScriptManager, UpdatePanel, UpdateProgress, and Timer—that enable a faster response. These controls are responsible for faster updates, better response times, and improved performance and efficiency. We’ll look at each of these controls in detail in the hours that follow.

ASP.NET Ajax also provides web services that can be used from the client script to work with different application services for forms authentication and user profiles. There are several ASP.NET application services provided with the Ajax server extensions, which can be accessed by web service calls from the client script. This data transfer can be handled by different network components that make it easy to return results of a web service call.

Other Ajax Frameworks

Apart from Microsoft’s ASP.NET Ajax, there are plenty of other Ajax frameworks. We discuss only the major ones—that is, those frameworks that are widely in use. Our list includes the following:

- Atlas
- AJAXPro.NET
- MagicAJAX.NET
- Anthem.NET

Atlas, as mentioned previously, is the older form of Microsoft Ajax Library. It is a framework that integrates the Client-Side JavaScript Library and is freely available and can be used with ASP.NET 2.0 to develop Ajax applications. It has cross-browser
support and exposes a number of object-oriented APIs, which can be used to develop web applications that minimize server hits or postbacks and perform asynchronous calls.

AJAXPro.NET is an Ajax library for use with the ASP.NET engine. The best part of AJAXPro.NET is that you can use it in the Internet Explorer event if the ActiveX object is disabled in your browser. Furthermore, it has certain distinct advantages over Atlas. It does not have any web service layer and is compliant with both versions of .NET—.NET 1.x and .NET 2.0—and is simple to understand and implement in your web applications.

MagicAJAX.NET, or the Magic Ajax engine for use with the ASP.NET engine, is also a freely available Ajax framework. It is a flexible Ajax framework for use in the ASP.NET platform. It was published in an article at www.codeproject.com. Since then, it was improved a lot and was later provided free at www.sourceforge.net.

Anthem.NET is an open source Ajax framework that is compliant with ASP.NET 1.x and 2.0 versions. This framework was developed by Jason Diamond and contains a rich set of Ajax-enabled controls that can be used to Ajax-enable your web applications. You have support for View State in Anthem.NET with a rich set of controls and server-side events.

**Goals of ASP.NET Ajax**

There are many goals for Ajax. The following are some of the most important:

- Improving performance and efficiency by negating page postbacks
- Introducing partial page updates to refresh only parts of a web page
- Reducing the network load
- Providing a framework with a collection of integrated server- and client-side components to ease development of web applications that can leverage the power of Ajax

The section that follows discusses the steps for installing Ajax.

**Installing Ajax**

To start developing ASP.NET Ajax applications using Visual Studio 2005, you first need to install and configure ASP.NET Ajax.
In discussing how to install Microsoft ASP.NET Ajax in your system, this section assumes that you have already downloaded the ASP.NET Ajax installer package from the download link provided later in this hour.

Setting Up Your Environment

Before you install ASP.NET Ajax in your system, make sure your system meets the minimum installation requirements as outlined in this section. The installation prerequisites are as follows:

- A Windows OS that can host the Microsoft .NET framework (Windows 2000, 2003, XP, Vista, and so on)
- Microsoft .NET framework 2.0 or Version 3.0
- Internet Explorer 5.01 or higher

By the Way

Before you proceed through the installation steps discussed here, make sure that you uninstall any previous versions of ASP.NET Ajax in your system.

Installing ASP.NET Ajax

The following are the steps for installing ASP.NET Ajax in your system:

1. Log in with the administrator's privileges and initiate the installation procedure.

2. Download the ASP.NET AJAX 1.0 from the ASP.NET Ajax downloads website:

3. Execute the ASPAJAXExtSetup.msi installation package to install ASP.NET Ajax in your system by double-clicking the .msi file and following the steps shown in Figures 1.1 through 1.5.
FIGURE 1.1
Installation Wizard—Step 1

FIGURE 1.2
Installation Wizard—Step 2
Alternatively, you can execute the ASPAJAXExtSetup.msi package from the command line:

```
msiexec /i ASPAJAXExtSetup.msi [/q] [/log <log file name>] [INSTALLDIR =<installation path>]
```
Creating Your First Ajax Application

The heart of any Ajax-enabled web page is the XMLHttpRequest object, an object that facilitates communication with the server without posting the web page back to the web server. This communication can be done synchronously or asynchronously.
without postbacks. In this section, we examine how we can implement a simple application using this object. The XMLHttpRequest object is discussed at length later in this book.

You have to instantiate the XMLHttpRequest object differently for different browsers, such as Mozilla and IE.

Let's now take a look at how you can use this object using JavaScript with different browsers. After we get an understanding of how the XMLHttpRequest object works, we can get into the details of the Microsoft Ajax Library and its usage. This Library encapsulates all the JavaScript code in the form of an API and presents us with a set of method calls that facilitate easier development in Ajax.

You need to follow specific steps to create your first sample application using the XMLHttpRequest object. These steps are discussed in this section and the next section.

1. Create an instance of the XMLHttpRequest object, as shown in the following code snippet:

   ```javascript
   var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP")
   ```

   In the next section we discuss how we can implement a generic function that we can use across all common browsers. Moving ahead, we will make use of this function to create an instance of XMLHttpRequest

By the Way

In Mozilla and Safari browsers, the XMLHttpRequest is a built-in native object. Creation of this object in these browsers is as follows:

   ```javascript
   var xmlHttp = new XmlHttpReuest();
   ```

Creating a Generic Function for Instantiating the XMLHttpRequest Object

In this section we implement a generic function that can be used to create an instance of the XMLHttpRequest object based on the type of the browser on which it is intended to be used. This function is as follows:

```javascript
var xmlHttp = false;
function getXmlHttpRequestObject() {
    try {
        //IE implementation
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch(e) {
        //Mozilla and Safari implementation
        xmlHttp = new XmlHttpRequest();
    }
}
```
Creating Your First Ajax Application

```javascript
{  
try  
{  
//Legacy implementation  
xmlHttp = new ActiveXObject("MsXm12.XMLHTTP");  
}  
catch(exp)  
{  
xmlHttp = false;  
}  
}  
if (!xmlHttp && typeof XmlHttpRequest != 'undefined')  
{  
//Mozilla based browsers  
//creating a native request object  
xmlHttp = new XmlHttpRequest();  
}
</script>

How Does It Work?

What does the previous code do? It illustrates the creation of the XMLHttpRequest object appropriate for the browser type making the request. It is done with the help of a simple exception-handling mechanism using the “try” and “catch” exception blocks. Note that this method can be called from any page in the application by simply placing this in a .js file and calling the getXmlHttpRequestObject method wherever it is required. If we place this in a .js file, we need not write the same code repeatedly in every web page.

2. Open Visual Studio 2005, and click File, New, Web Site to create a new web site. Name this web site and save it (see Figure 1.6).
3. Now place the following code in the Default.aspx HTML source:

```csharp
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<script language="javascript" type="text/javascript">
var xmlHttp = false;
function getXmlHttpRequestObject()
{
    try
    {
        //IE implementation
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch(e)
    {
        try
        {
            //Legacy implementation
            xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
        }
        catch(exp)
        {
            xmlHttp = false;
        }
    }

    if (!xmlHttp && typeof XMLHttpRequest != 'undefined')
    {
        //Mozilla based browsers
        //creating a native request object
        xmlHttp = new XMLHttpRequest();
    }
}
</script>
<head runat="server">
    <title>Creating your first ASP.NET Ajax Application</title>
</head>
<body>
<form id="form1" runat="server">
    <div>
        Your First AJAX application at work!
    </div>
</form>
</body>
</html>
```

When you execute the application, the output is similar to what is shown in Figure 1.7.
We now discuss how we can use Ajax to fetch data from the server asynchronously and display it in the client browser.

4. We have the JavaScript function in place. Now let's call and instantiate it, as shown in the following code snippet:

```javascript
getXmlHttpRequestObject();

Now, the instance is ready for use:

```xmlHttp.open("GET", "TestFile.txt", true);
xmlHttp.onreadystatechange = function()
{
  if (xmlHttp.readyState == 4)
  {
    alert(xmlHttp.responseText);
  }
}
xmlHttp.send(null);
```

**What Does the Previous Code Do?**

What does the previous code do? It uses the XMLHttpRequest object's GET/Open method to read a text file on the server and display its content at the client side in a message box using JavaScript.

Here is how the complete code inside the `<Script>` tag now looks like:

```html
<script language="javascript" type="text/javascript">
  var xmlHttp = false;
  getXmlHttpRequestObject();
  xmlHttp.open("GET", "TestFile.txt", true);
  xmlHttp.onreadystatechange = function()
  {
    if (xmlHttp.readyState == 4)
    {
      alert(xmlHttp.responseText);
    }
  }
  xmlHttp.send(null);
</script>
```
if (xmlHttp.readyState == 4)
{
    alert(xmlHttp.responseText);
}
xmlHttp.send(null);

function getXmlHttpRequestObject()
{
    try
    {
        //IE implementation
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch(e)
    {
        try
        {
            //Legacy object implementation
            xmlHttp = new ActiveXObject("MsXml2.XMLHTTP");
        }
        catch(exp)
        {
            xmlHttp = false;
        }
    }
    if (!xmlHttp && typeof XMLHttpRequest != 'undefined')
    {
        //Mozilla based browsers
        //creating a native request object
        xmlHttp = new XMLHttpRequest();
    }
}</script>

5. Add a file called TestFile.txt to your project, open it, and type some content, such as “Welcome to the world of Ajax.”

6. Save your work and start executing your project. You’ll be prompted to add the Web.Config file to your application to enable debugging.

7. Accept all the defaults, and click the OK button. This executes the application and opens up a browser, showing the contents of the text file retrieved from the server on an alert in your web page. We are done!

The output is captured, as shown in Figure 1.8.
In the code snippet shown earlier, the open method was used to communicate with the server, and “readyState” returned a status code of 4, which implies that a transaction completed successfully.

So, after reading this section, you now should have a basic idea of how Ajax can be put in action.

**Summary**

With Microsoft taking Ajax to new heights and quickly coming up with newer releases, Ajax is all set to become the next-generation technology of choice for building fast and responsive web applications. In this hour, we discussed the various set of technologies that make up Ajax, the key ingredients of ASP.NET Ajax, the major goals of ASP.NET Ajax, and why Ajax has become indispensable in web application development communities worldwide. Apart from this, we’ve also learned how to install ASP.NET Ajax and what comes with it. In the next hour, we’ll explore the internals of the ASP.NET Ajax architecture. So, stay tuned!
Workshop

Quiz

1. What are the basic goals of ASP.NET Ajax?
2. What are the constituent technologies in Ajax?
3. Name some of the server controls that are included as part of the ASP.NET Ajax framework.
4. What are the minimum requirements/prerequisites for installing and running ASP.NET Ajax applications?
5. Name some popular ASP.NET Ajax frameworks.

Answers

1. The basic goals of ASP.NET Ajax are as follows:
   - Reduced web server hits and network load
   - Rich and interactive user interface
   - Platform and architecture neutrality
   - Support for both synchronous and asynchronous communication
   - Provide a server- and client-side framework for seamless usage of Ajax in applications

2. The constituent technologies that make up Ajax are as follows:
   - XMLHttpRequest object
   - JavaScript
   - DHTML
   - DOM
   - XML
3. ASP.NET Ajax has come up with several server controls. These are the following:
   - ScriptManager
   - UpdatePanel
   - UpdateProgress
   - Timer

   These controls are responsible for faster updates, better response times, and improved performance and efficiency.

4. To install and run ASP.NET Ajax applications, you should have all of the following in your system:
   - A Windows OS that can host the Microsoft .NET framework (Windows 2000, 2003, XP, Vista, and so on)
   - Microsoft .NET framework 2.0 or Version 3.0
   - Internet Explorer 5.01 or higher

5. Some of the most popular ASP.NET Ajax frameworks include the following:
   - Michael Schwarz’s Ajax.NET
   - Jason Diamond’s Anthem
   - MagicAJAX.NET
   - ComfortASP.NET
   - Microsoft’s Atlas
   - ASP.NET Ajax from Microsoft
Index

A
adding
  scripts, ScriptManagerProxy control, 108
  services with
    ScriptManagerProxy control, 108
Address table (e-commerce sample application), 313-314
Admin.sitemap file (e-commerce sample application), 335
Administrator role (e-commerce sample application), 302
Ajax, 4
  AJAXPro.NET, 9
  Anthem.NET, 9
  ASP.NET Ajax, 8
  Atlas, 8
  creating applications, 13-14
    XMLHttpRequest object, 14-19
  CSS and, 5
  DHTML and, 5
  DOM and, 5
downside of, 6
  error handling, 137-140
  future of, 293-294
  history of, 7
  HTML and, 5
  JavaScript and, 5
  JSON and, 78-81
  MagicAJAX.NET, 9
  web parts and, 185-186
    client-side callbacks, 188-191
    UpdatePanel, 186-188
    XMLHttpRequest object, 4
Ajax Control Toolkit, 147-148, 152
  Ajax-enabled controls, 148-149
    building, 149-150
    extenders, 150-151
    script controls, 151-152
Ajax Control Toolkit

AutoComplete extender, 153-158
ConfirmButton extender, 161-169
DropDown extender, 169-172
Ajax control toolkit, registering, 333, 352
Ajax-enabled controls, 148-149
building, 149-150
extenders, 150
building, 151
script controls, 151
building, 152
AJAXPro.NET, 9
Anthem.NET, 9
Application class, Sys namespace, 196
application life cycle events, ASP.NET, 24-25
application services, ASP.NET
Ajax server extensions framework, 105
application services bridge, 32
ASP.NET Ajax server extension framework, 106
applications
creating Ajax applications, 13-14
XMLHttpRequest object, 14-19
e-commerce sample application, 299
adding/editing categories, 350-358
architecture and operational flow, 303-304
business objects, creating, 346-350
carts, updating, 366-371
creating categories, 343-345
customer registration page, creating, 327-334
database design, 305-314
home pages, creating, 327
master pages, creating, 317-322
modules in, 300-302
order history pages, creating, 388-391
order pages, creating, 376-382
order tracking, 382-384
pending order management, 384-387
product details pages, 363-365
product searches in, 359-363
products, adding to cart, 365-366
site navigation, 334-339
user and role management in, 322-326
architecture
ASP.NET, 24
application life cycle events, 24-25
page life cycle events, 25-28
ASP.NET Ajax, 28, 30
client framework, 32
server framework, 30-32
e-commerce sample application, 303-304
Microsoft Ajax Client Library, 90-91
core framework, 91-93
user interface framework, 93-94
Array type extensions, 96
arrays, storing multiple ordered items (JSON), 74-75
ASP.NET, 23
architecture, 24
application life cycle events, 24-25
page life cycle events, 25-28
ASP.NET 2.0, 105
Membership API, 105
Personalization API, 105
Profile API, 105
ASP.NET Ajax, 3, 7-8
architecture, 28, 30
client framework, 32
server framework, 30-32
goals for, 9
installing, 9-13
setting up your environment, 10
ASP.NET Ajax client-side event model, client life cycle events, 195-198
ASP.NET Ajax Futures, 285
ASP.NET Ajax Futures CTP, 286
dynamic data controls, 288-292
pure client-side controls in Sys.Preview.UI namespace, 286-287
support for returning DataSet, DataTable, or DataRow instances, 292-293
ASP.NET Ajax Pure Client-Side Library, 285
ASP.NET Ajax server controls, 106
ScriptManager, 107
ScriptManagerProxy, 108
Timer, 107
UpdatePanel, 109
UpdateProgress, 108
ASP.NET Ajax server extensions framework, 103-105
components of, 106
application services bridge, 106
server controls, 106-109
web services bridge, 106
ASP.NET Futures Ajax web sites, 288
ASP.NET SQL Server Registration Tool, 323
aspnet_Roles table (e-commerce sample application), 308
aspnet_Users table (e-commerce sample application), 308-309
aspnet_UsersInRoles table (e-commerce sample application), 309
assemblies
embedding script resources in, 267-270
resource assemblies, 270-271
asynchronous communication layer, 209-211
calling web service proxy classes
with HTTP GET verb, 217
with HTTP POST verb, 217
calling web services using client scripts, 214-216
page method proxy classes, 217-219
sending HTTP requests from clients, 211-214
asynchronous data retrieval, XMLHttpRequest object, 42-43
asynchronous postbacks, sequence of events, 196
AsyncPostBackTrigger, 127-128
Atlas, 8
attributes
profileService, 236
ScriptService attribute, 238
authentication, 222
ASP.NET Ajax server extensions framework, 104
custom authentication, 222
forms authentication, 222
passport authentication, 223
types of, 222
windows authentication, 223
authentication service, client-side scripts, 223-224
implementing sample applications, 224-232
AuthenticationService object, 224
AutoComplete extender
Ajax Control Toolkit, 153-158
properties of, 155-156
AutoCompleteExtender control, 285
avoiding PageRequestManager-ParserErrorException, 202-203

B
B2B (Business to Business), 299
B2C (Business to Consumer), 299
behaviors, 250
extending Microsoft Ajax Library, 256
simulating Ajax behaviors without XMLHttpRequest object, 50
binding

categories with drop-down list, 359
order history to GridView control, 389
pending orders to GridView control, 385
products list to DataList control, 361

Boolean type extensions, 96
Browser Compatibility Layer, 32
business objects for e-commerce sample application, creating, 346-350

Business to Business (B2B), 299
Business to Consumer (B2C), 299

C

callback operations, 190
Cart table (e-commerce sample application), 311-312
CartDetails table (e-commerce sample application), 311-312
CartDetailsTableAdapter (e-commerce sample application), methods in, 349
cart

defined, 365
in e-commerce sample application
adding products to, 365-366
updating, 366-371

Cascading Style Sheets (CSS), 54-56
Ajax and, 5
DIV tag, 57
making dynamic, 57-58
Style properties, 56-57
categories in e-commerce application
adding/editing, 350-358
binding with drop-down list, 359
creating, 343-345
Categories table (e-commerce sample application), 309-310

CategoriesTableAdapter (e-commerce sample application), methods in, 349
checking out, order page (e-commerce sample application), 376
Payment Details section, 379-382
Review the Order section, 376-378
Shipping Address section, 378-379

classes
Application class, Sys namespace, 196
Debug, 274-278
page method proxy classes, 217-219
PageRequestManager, 196
Provider abstract classes, 106
Sys.Application class, events, 198
in Sys namespace, 88
Sys.Component class, 92
Sys.Exceptions class, 92
Sys.Net.WebRequest class, 92
Sys.Net.WebRequestExecutor class, 93
Sys.Net.XMLHttpExecutor class, 93
Sys.Services.authentication-Service class, 244
Sys.UI.Behavior class, 93-94
Sys.UI.Control class, 93
Sys.WebForms.PageRequestManager class, 92
Sys._Application class, 91
System.UI.DomElement class, 94
System.UI.DomEvent class, 94

ClearTrace(), 276
client framework, ASP.NET
Ajax, 32
client life cycle events in ASP.NET
Ajax, 195-198
client scripts, calling web services, 214-216
client-side callbacks, web parts and Ajax, 188-191
client-side controls, pure
client-side controls in Sys.Preview.Ul namespace, 286-287
client-side scripting
CSS, 54
Ajax and, 5
DIV tag, 57
making dynamic, 57-58
Style properties, 56-57
DHTML, 53-54
Ajax and, 5
implementing, 63-66
client-side scripts, authentication service, 223-224
implementing sample applications, 224-232
clients, sending HTTP requests from, 211-214
code, debugging, 278-279
alert statements (JavaScript), 279
components, 250
of ASP.NET Ajax server
extensions framework, 106
application services bridge, 106
server controls, 106-109
web services bridge, 106
extending Microsoft Ajax Library, 250
creating custom client components, 250-252
using custom client components, 252-255
ConfirmButton extender, 161-169
properties of, 165
ContentTypePlaceHolder control, 371
ContentTemplate, UpdatePanel, 122-123
controls, 250
Ajax-enabled controls, 148-149
building, 149-150
extenders, 150-151
script controls, 151-152
ContentPlaceHolder, 371
CreateUserWizard, 327-329
DataList, binding products list to, 361
DetailsView, 344, 350-353
events, 353-354
dynamic data controls, 288-292
customizing display of, 290
mapping, 292
DynamicAutoData, 289, 291
DynamicDetails, 289
DynamicDetailsView, 291
DynamicFilter, 289
DynamicInsert, 289
DynamicList, 289
DynamicList control, 289
DynamicNavigator, 289
DynamicRssLink, 289
extending Microsoft Ajax Library, 255-256
FormView, 364
GridView, 166, 291, 344-345, 367, 376
binding order history to, 389
binding pending orders to, 385
events, 353-354
Login, 320
LoginStatus, 321
Menu, 335
loading sitemap files into, 336
ObjectDataSource, 345, 353, 368
ScriptManager, 329
overview, 135-137
Timer, 140-141
implementing partial page updates, 142-145
UpdatePanel, 356, 371
ContentTypePlaceHolder, 122-123
event model, 125
implementing partial page updates, 142-145
partial rendering, 119-122
render modes, 124-125
UpdateMode property, 125-127
UpdateProgress, 129-131, 356, 371
Core Component Layer, 32
core framework of Microsoft Ajax Client Library

- Sys.Component class, 92
- Sys.Exceptions class, 92
- Sys.Net.WebRequest class, 92
- Sys.Net.WebRequestExecutor class, 93
- Sys.Net.XMLHttpExecutor class, 93
- Sys.WebForms.PageRequestManager class, 92
- Sys._Application class, 91

Core Services Layer, 32

CreateUserWizard control, 327, 329

credit card payments for e-commerce sample application, 379-382

CSS (Cascading Style Sheets), 54-56

- Ajax and, 5
- DIV tag, 57
  making dynamic, 57-58
- Style properties, 56-57

custom authentication, 222

custom client components
- creating, 250-252
- using, 252-255

custom web parts, 182-183

customer registration page (e-commerce sample application), creating, 327-334

Customer role (e-commerce sample application), 302

Customer.sitemap file (e-commerce sample application), 335

customizing display of dynamic data controls, 290

data interchange formats, 71

- HTML content, 71
- JSON, 73-74
- plain text or string format, 71-72
- XML, 72-73

data retrieval
- asynchronous data retrieval, XMLHttpRequest object, 42-43
- synchronous data retrieval, XMLHttpRequest object, 40-41

database design for e-commerce sample application, 305-307

- tables and relationships, 308-314

databases
- mapping users and roles to, 322-326
- mapping users to, 306

dataList control, binding products list to, 361

dataRow, ASP.NET Ajax Futures CTP, 292-293

DataSet, ASP.NET Ajax Futures CTP, 292-293

DataSet objects, creating, 346

DataTable, ASP.NET Ajax Futures CTP, 292-293

Date type extensions, 97

dbg attribute, 279

Debug class, 274-278
debbuging, 274

- ASP.NET Ajax server extensions framework, 104
- code, 278-279
  - alert statements (JavaScript), 279
- Debug class, 274-278
  - enabling debugging support
    - in Internet Explorer, 280-282
    - in Visual Studio, 282
  - in web.config file, 280
- Sys.Debug, 275-278

delegates, 359
design of database, 305-307
detailsView control, 344, 350-353

- events, 353-354
developing web parts using user controls, 183-185

DHTML (Dynamic Hypertext Markup Language), 53-54

Ajax and, 5
display modes, web parts, 178-179
displaying. See also viewing
localized string in the client side, 260-264
progress data during partial updates, 108
DisplayText() method, 265
DIV, CSS, 57
Document object, 62-63
DOM (Document Object Model), 60
Ajax and, 5
Document object, 62-63
JavaScript and, 60-62
representation of
DOM tree, 61
downside of Ajax, 6
DropDown extender, 169-172
dynamic data controls, 288-292
customizing display of, 290
mapping, 292
dynamic HTML (DHTML), 53-54
Ajax and, 5
dynamic web sites, creating, 287
DynamicAutoData control, 289-291
DynamicDetails control, 289
DynamicDetailsView control, 291
DynamicFilter control, 289
DynamicInsert control, 289
DynamicList control, 289-291
DynamicNavigator control, 289
DynamicRssLink control, 289

E

e-commerce, defined, 299-300
e-commerce sample
application, 299
architecture and operational flow, 303-304
business objects, creating, 346-350
carts, updating, 366-371
categories
adding/editing, 350-358
creating, 343-345
customer registration page, creating, 327-334
database design, 305-307
tables and relationships, 308-314
home pages, creating, 327
master pages, creating, 317-322
modules in, 300
home page/registration/login, 300
order and product management, 302
order generation/online payment, 301
search/product display page, 301
shopping cart management, 301
user and role management, 302
order history pages, creating, 388-391
order pages, creating, 376-382
order tracking, 382-384
pending order management, 384-387
product details pages, 363-365
product searches in, 359-363
products, adding to cart, 365-366
site navigation, 334-339
user and role management in, 322-326
editing categories in e-commerce sample application, 350-358
Element object, 63
methods, 63
embedded scripts, 270-271
embedding script resources in assemblies, 267-270
EnableScriptGlobalization, 260
enabling
debugging support
in Internet Explorer, 280-282
in Visual Studio, 282
in web.config file, 280
page methods, 370
profile service, 235-236
error handling with Ajax, 137-140
Error type extensions, 97

How can we make this index more useful? Email us at indexes@samspublishing.com
errors, PageRequestManagerParserErrorException, 198-199
  avoiding, 202-203
  reproducing, 200-202
  event delegates, 359
  event handling with JavaScript, 58-60
  event model, UpdatePanel, 125
  event
    for GridView and DetailsView controls, 353-354
    of Sys.Application class, 198
    of Sys.Webforms.PageRequestManger, 197
  example application. See e-commerce sample application extenders
    Ajax-enabled controls, 150
      building, 151
    ConfirmButton extender, 161-169
    DropDownList extender, 169-172
  extending Microsoft Ajax Library, 249-250
    with behaviors, 256
    with components, 250-255
    with controls, 255-256
  extending JavaScript Library with Microsoft Ajax JavaScript Extensions framework, 95-97
  extensions
    Array type extensions, 96
    Boolean type extensions, 96
    Date type extensions, 97
    Error type extensions, 97
    Number type extensions, 96
    Object type extensions, 95
    String type extensions, 96
  forms authentication, 222
  FormView control, 364
  GetCallbackEventReference, 191
  getXmlHttpRequestObject( ) method, 39
  Global namespace, Microsoft Ajax Client Library, 87-88
  globalization, 259-260
    ASP.NET Ajax server extensions framework, 104
    JavaScript, 264-267
    script globalization, 267
  goals
    for ASP.NET Ajax, 9
    of Microsoft Ajax Client Library, 86
  Google, Ajax and, 7
  GridView, 166
  GridView control, 291, 344-345, 367, 376
    binding order history to, 389
    binding pending orders to, 385
    events, 353-354
  history
    of Ajax, 7
    of XMLHttpRequest object, 38
  home page (e-commerce sample application), creating, 327
  home page module (e-commerce sample application), 300
  HTML, Ajax and, 5
  HTML content, data interchange formats, 71
  HTTP GET verb, calling web services, 217
  Http handlers, 50
  HTTP POST verb, calling web services, 217
  HttpApplication objects, 24-25
  IFRAMES, 50
    partial page rendering, 119
  Images table (e-commerce sample application), 311
  ImagesTableAdapter (e-commerce sample application), methods in, 349
  implementing client-side scripting, 63-66
  partial page updates with UpdatePanel and Timer controls, 142-145
installing ASP.NET Ajax, 9-13
setting up your environment, 10
Internet Explorer, enabling debugging support, 280-282

J–K

JavaScript
Ajax and, 5
alert statements, debugging, 279
DOM and, 60-62
event handlers, 59
event handling, 58-60
globalization and localization, 264-267
JavaScript library, extending with Microsoft Ajax JavaScript Extensions framework, 95-97
JavaScript Object Notation. See JSON

JSON (JavaScript Object Notation), 33, 73-77
Ajax and, 78-81
arrays, storing multiple ordered items, 74-75
parsing, 77-78
storing name/value pairs with object literals, 75-76

L
layers, asynchronous communication layer, 209-211
calling web service proxy classes with HTTP GET verb, 217
calling web service proxy classes with HTTP POST verb, 217
calling web services using client scripts, 214-216
t_page method proxy classes, 217-219
sending HTTP requests from clients, 211-214

libraries
JavaScript Library, 95-97
Microsoft Ajax Client Library, 85-87
building blocks of, 90-94
goals of, 86
namespaces, 87-90
Microsoft Ajax Library with behaviors, 256
with components, 250-255
with controls, 255-256
Microsoft Ajax server reference library, namespaces, 110
System.Web.
Configuration, 112
System.Web.Handlers, 112

System.Web.Script, 113
System.Web.Script.Services, 113
System.Web.UI, 110-111
System.Web.UI.Design, 111-112

LoadPostBackData, 26
localization, 259-260
ASP.NET Ajax server extensions framework, 104
displaying localized strings in client side, 260-264
JavaScript, 264-267
script localization, 267
login, creating SQL login, 305
Login control, 320
login page module (e-commerce sample application), 300
LoginStatus control, 321

M
MagicAJAX.NET, 9
Manager role (e-commerce sample application), 302
mapping
dynamic data controls, 292
users and roles to databases, 322-326
users to databases, 306
master pages (e-commerce sample application), creating, 317-322

How can we make this index more useful? Email us at indexes@samspublishing.com
Membership API

Membership API, ASP.NET 2.0, 105
Menu control, 335

loading sitemap files into, 336

methods
in CartDetailsTableAdapter (e-commerce sample application), 350
in CategoriesTableAdapter (e-commerce sample application), 349
Document object, 62
Element object, 63
in ImagesTableAdapter (e-commerce sample application), 349
Node object, 62
in ProductsTableAdapter (e-commerce sample application), 349
of XMLHttpRequest, 44

Microsoft Ajax Client Library, 85-86

building blocks of, 90-91
core framework, 91-93
user interface framework, 93-94
goals of, 86
namespaces, 87

Global namespace, 87-88
Sys namespace, 88
Sys.Net namespace, 89
Sys.Serialization namespace, 89
Sys.Services namespace, 89
Sys.UI namespace, 89
Sys.WebForms namespace, 89-90

Microsoft Ajax JavaScript Extensions framework, extending JavaScript Library, 95-97

Microsoft Ajax Library, extending, 249-250

with behaviors, 256
with components, 250-255
with controls, 255-256

Microsoft Ajax server reference library, namespaces, 110

System.Web.Configuration, 112
System.Web.Handlers, 112
System.Web.Script, 113
System.Web.Script.Services, 113
System.Web.UI, 110-111
System.Web.UI.Design, 111-112

modules in e-commerce sample application, 300

home page/registration/login, 300
order and product management, 302
order generation/online payment, 301
search/product display, 301
shopping cart management, 301
user and role management, 302

Microsoft Ajax server reference library, 110

System.Web.Configuration, 112
System.Web.Handlers, 112
System.Web.Script, 113
System.Web.Script.Services, 113
System.Web.UI, 110-111
System.Web.UI.Design, 111-112

Sys.Preview.UI, pure client-side controls, 286-287
System.Globalization, 260
System.Resources, 260

name/value pairs, storing with object literals, 75-76

navigation, 334-339

Node object methods, 62
Node object properties, 61
Number type extensions, 96
object literals, storing name/value pairs, 75-76
Object type extensions, 95
ObjectDataSource control, 345, 353, 368

objects
  Document object, 62-63
  Element object, 63
online payment module (e-commerce sample application), 301
operational flow in e-commerce sample application, 303-304
order and product management (e-commerce sample application), 302
order generation module (e-commerce sample application), 301
order history page (e-commerce sample application), creating, 388-391
order management in e-commerce sample application, 384-387
order page (e-commerce sample application), creating, 376
  Payment Details section, 379-382
  Review the Order section, 376-378
  Shipping Address section, 378-379
Order table (e-commerce sample application), 312-313
order tracking (e-commerce sample application), 382-384

Partial Update group, 31
partial-page rendering, 117-118
IFrame, 119
reasons for using, 118
UpdatePanel, 119-122
XMLHttpRequest, 119
passport authentication, 222-223

Payment Details section (e-commerce sample application), 379-382
payment module (e-commerce sample application), 301
PayPal eCommerce Starter Kit, 380
pending orders in e-commerce sample application, managing, 384-387
personalization, 175-176
Personalization API, ASP.NET 2.0, 105
placing orders, creating order pages (e-commerce sample application), 376
  Payment Details section, 379-382
  Review the Order section, 376-378
  Shipping Address section, 378-379

@Page directive, debug attribute, 279
page life cycle events
  ASP.NET, 25-28
  UpdatePanel, 125
page method proxy classes, 217-219
page methods
  enabling, 370
  for price updates in e-commerce sample application, 369-370
PageHandlerFactory, 25
PageRequestManager class, 196
PageRequestManagerParserError Exception, 198-199
  avoiding, 202-203
  reproducing, 200-202
parsing JSON, 77-78
Partial Page Rendering group, 31
partial page updates, implementing with UpdatePanel and Timer controls, 142-145
partial rendering
  Sys.WebForms namespace, 89-90
UpdatePanel
  event model, 125
  render modes, 124-125
  UpdateMode property, 125-127

How can we make this index more useful? Email us at indexes@samspublishing.com
plain text format, data interchange formats

properties
- of AutoComplete extender control, 155-156
- of ConfirmButton extender, 165
- Document object, 62
- Node object, 61
- responseText, 48
- Style properties, CSS, 56-57
- UpdateMode property, 125-127
- of XMLHttpRequest, 43

Provider abstract classes, 106

pure client-side controls in
SystePreview.UI namespace, 286-287

Q–R

RaiseCallbackEvent method, 190

readyState property, status codes for XMLHttpRequest, 43

registering
- Ajax control toolkit, 333, 352
- customers in e-commerce sample application, 327-334

registration page module
(e-commerce sample application), 300

relationships for e-commerce sample application tables, 308-314

render modes, UpdatePanel, 124-125

rendering
- partial rendering, 124
- Sys.WebForms namespace, 89-90
- UpdatePanel, 124-127
- partial-page rendering, 117-118
- IFrame, 119
- reasons for using, 118
- UpdatePanel, 119-122
- XMLHttpRequest, 119

reproducing
- PageRequestManagerParserErrorException, 200-202

request and response cycle, 69-70

resource assemblies, 270-271

resource files, 260

Response object, 27

response times, 69-70

responseText property, 48

Review the Order section
(e-commerce sample application), 376-378

role management, 302

roles in e-commerce sample application, 308

mapping to users and databases, 322-326
sample application, e-commerce
sample application, 299
architecture and operational flow, 303-304
business objects, creating, 346-350
carts, updating, 366-371
categories
  adding/editing, 350-358
  creating, 343-345
customer registration page, creating, 327-334
database design, 305-307
tables and relationships, 308-314
home pages, creating, 327
master pages, creating, 317-322
modules in, 300
  home page/registration/login, 300
order and product management, 302
order generation/online payment, 301
search/product display page, 301
shopping cart management, 301
user and role management, 302
order history pages, creating, 388-391

order pages, creating, 376-382
order tracking, 382-384
pending order management, 384-387
product details pages, 363-365
product searches in, 359-363
products, adding to cart, 365-366
site navigation, 334-339
user and role management in, 322-326
SaveProfile() method, 246-247
schema for e-commerce sample application, setting up, 323
script controls, Ajax-enabled controls, 151
  building, 152
script descriptors, 148
script globalization, 267
script localization, 267
script references, 149
script resources, embedding in assemblies, 267-268, 270
scripting, client-side scripting, 54
CSS (Cascading Style Sheets), 54
  Ajax and, 5
DIV tag, 57
  making dynamic, 57-58
Style properties, 56-57
DHTML (Dynamic Hypertext Markup Language), 53-54
  Ajax and, 5
implementing, 63-66

ScriptManager, 31
  overview, 135-137
ScriptManager control, 107, 329
ScriptManagerProxy, 31
ScriptManagerProxy control, 108
scripts
  adding with ScriptManagerProxy control, 108
  embedded scripts, 270-271
Scripts tag, 271
ScriptService attribute, 238
search module (e-commerce sample application), 301
searches in e-commerce sample application, 359-363
sending HTTP requests from clients, 211-214
server controls, ASP .NET Ajax server extension framework, 106
  ScriptManager, 107
  ScriptManagerProxy, 108
  Timer, 107
  UpdatePanel, 109
  UpdateProgress, 108
server framework, ASP .NET Ajax
  application services bridge, 32
  server controls, 30-31
  web services bridge, 32
ServiceMethod, 154
services, adding with ScriptManagerProxy control, 108

How can we make this index more useful? Email us at indexes@samspublishing.com
Shipping Address section (e-commerce sample application)

order history pages, creating, 388-391
order pages, creating, 376-382
order tracking, 382-384
pending order management, 384-387
product details pages, 363-365
product searches in, 359-363
products, adding to cart, 365-366
site navigation, 334-339
user and role management in, 322-326
ShowTrace(), 276

simulating Ajax behavior without using XMLHttpRequest object, 50

site navigation for e-commerce sample application, 334-339

sitemap files, loading into Menu control, 336

SQL login, creating, 305

storing

multiple ordered items, arrays (JSON), 74-75
name/value pairs with object literals, 75-76

string format, data interchange formats, 71-72

String type extensions, 96

Style properties, CSS, 56-57

synchronous data retrieval, XMLHttpRequest object, 40-41

Sys namespace
Application class, 196
Microsoft Ajax Client Library, 88
Sys.Application class, events for, 198
Sys.Component class, 92
Sys.Debug, 275-278
Sys.Exceptions class, 92
Sys.Net namespace, Microsoft Ajax Client Library, 89
Sys.Net.WebRequestExecutor class, 93
Sys.Net.WebRequest, 211
Sys.Net.WebRequest class, 92
Sys.Net.XMLHttpExecutor class, 93

Sys.Preview.UI, pure client-side controls, 286-287

Sys.Serialization namespace, Microsoft Ajax Client Library, 89
Sys.Services namespace, Microsoft Ajax Client Library, 89
Sys.Services.authentication-Service class, 244

Sys.Services.Authentication-Service.logout method, 244
Sys.UI namespace, Microsoft Ajax Client Library, 89
Sys.UI.Behavior class, 93-94
Sys.UI.Control class, 93

Sys.WebForms namespace
Microsoft Ajax Client Library, 89-90
PageRequestManager, 196
table adapters in e-commerce sample application, creating, 346-350

tables for e-commerce sample application, 308-314

tags
  DIV tag, CSS, 57
  scripts, 271
  TargetControlID, 154
  TemplateField, 163
  Timer, 140-141
  implementing partial page updates, 142-145
  Timer control, 107
  trace messages, viewing, 275
  TraceConsole, 276
  tracing, 274
  tracking orders (e-commerce sample application), 382-384
  triggers, 127
    AsyncPostBackTrigger, 127-128
    PostBackTrigger, 129
  types of authentication, 222

T

table adapters in e-commerce sample application, creating, 346-350

tables for e-commerce sample application, 308-314

U

UpdateMode property,
  UpdatePanel, 125-127
  UpdatePanel control, 109, 356, 371

updates
  partial page updates,
    UpdatePanel control, 109
  partial updates, displaying progress data, 108

updating
  carts in e-commerce sample application, 366-371
  price in e-commerce sample application, 369-370

user and role management
  (e-commerce sample application), 302

user controls, developing web parts, 183-185

user interface framework of Microsoft Ajax Client Library
  Sys.Ui.Behavior class, 93-94
  Sys.Ui.Control class, 93
  System.Ui.DomElement class, 94
  System.Ui.DomEvent class, 94

users
  in e-commerce sample application, 308
    mapping to roles and databases, 322-326
  mapping to databases, 306
V–W

viewing
  product display page, 301
  trace messages, 275
Visual Studio, enabling debugging support, 282
web part zones, 177
web parts, 175-177
  Ajax and, 185-186
    client-side callbacks, 188-191
  UpdatePanel, 186-188
  creating, 179-182
  custom web parts, 182-183
  developing with user controls, 183-185
  features of, 177-179
web service proxy classes
  calling web services with
    HTTP GET verb, 217
  calling web services with
    HTTP POST verb, 217
web services, calling using client scripts, 214-216
web services bridge, 32
  ASP.NET Ajax server extension framework, 106
web sites
  ASP.NET Futures Ajax web sites, 288
  dynamic web sites, creating, 287
  web.config file, enabling debugging support, 280
  WebForm_DoCallback, 191
  WebPartManager, 179, 182
  WebRequestManager, 211
  WebResource, 268
  website navigation, 334-339
  windows authentication, 222-223

X–Z

XML, data interchange formats, 72-73
XmlHttpExecutor, 211
XMLHttpRequest
  methods of, 44
  properties of, 43
  status codes for readyState property, 43
XMLHttpRequest object, 4
  asynchronous data retrieval, 42-43
  creating, 38-40
  creating Ajax applications, 14-19
  overview of, 37-38
  synchronous data retrieval, 40-41
  working with, 44-49