

Robert Foster

# ASP.NET 3.5 AJAX

**UNLEASHED**

**SAMS**

## ASP.NET 3.5 AJAX Unleashed

Copyright © 2009 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-32973-9

ISBN-10: 0-672-32973-5

*Library of Congress Cataloging-in-Publication Data:*

Foster, Robert Hill.

ASP.NET 3.5 Ajax unleashed / Robert Foster.

p. cm.

ISBN 0-672-32973-5

1. Active server pages. 2. Ajax (Web site development technology) 3.

Application software—Development. 4. Microsoft .NET. I. Title.

TK5105.8885.A52F67 2008

005.2'76—dc22

2008014763

Printed in the United States of America

First Printing December 2008

### Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

### Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or programs accompanying it.

### Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

**U.S. Corporate and Government Sales**

**1-800-382-3419**

**corpsales@pearsontechgroup.com**

For sales outside of the U.S., please contact

**International Sales**

**international@pearson.com**

**Editor-in-Chief**  
Karen Gettman

**Executive Editor**  
Neil Rowe

**Development Editor**  
Mark Renfrow

**Managing Editor**  
Kristy Hart

**Project Editor**  
Meg Shaw

**Copy Editors**  
Lori Lyons  
Karen Annett

**Indexer**  
Erika Millen

**Proofreader**  
Anne Goebel

**Technical Editor**  
Todd Meister

**Publishing  
Coordinator**  
Cindy Teeters

**Book Designer**  
Gary Adair

**Composition**  
Jake McFarland

**Manufacturing Buyer**  
Dan Uhrig

# Introduction

As an ASP.NET web developer in today's market, you need countless skills to distinguish yourself from the next person. One of those skills that you can quickly acquire is developing AJAX-enabled applications. As your users begin to utilize AJAX-enabled sites on the web, such as Live.com and Google Maps, they are beginning to expect the same rich functionality in the applications that you deliver.

Although these applications took many man hours to create, the process is made easier for you as a developer with the AJAX library that's packaged with ASP.NET 3.5. You can save many hours and lines of code by learning and leveraging ASP.NET AJAX to create and deliver a very rich user experience in your applications.

In this book, you learn how to make the most of ASP.NET AJAX by building on the knowledge that you already have as an ASP.NET developer and extend that knowledge so that you can easily create AJAX-enabled applications. The book has been divided into the following chapters:

## **Chapter 1—Introduction to AJAX Technologies**

In this chapter, you learn the fundamentals of AJAX by first understanding the XMLHttpRequest JavaScript object, and then building a simple AJAX library for use in your applications.

## **Chapter 2—Introduction to ASP.NET AJAX**

Chapter 2 introduces you to the controls and capabilities that are available in ASP.NET AJAX and serves as a springboard for technologies that are discussed in future chapters.

## **Chapter 3—The ScriptManager and ScriptManagerProxy Controls**

Chapter 3 discusses the core object of the ASP.NET AJAX library: the ScriptManager control. In this chapter, you learn the capabilities of the ScriptManager control and how you can best utilize it in your AJAX-enabled pages.

## **Chapter 4—The ASP.NET AJAX Client API**

In Chapter 4, you learn how to make the most of the ASP.NET AJAX client-side objects that are made available via the ScriptManager control.

## **Chapter 5—The UpdatePanel and Timer Controls**

In Chapter 5, you learn how to use the UpdatePanel control to quickly and easily add AJAX functionality to your ASP.NET web pages. You also learn how to use the Timer control to make asynchronous callbacks from your pages at a specified time interval.

**Chapter 6—Advanced Techniques with the ASP.NET AJAX PageRequestManager Object**

In Chapter 6, you learn some essential techniques that can be used with the ASP.NET AJAX PageRequestManager object to save you many lines of code as well as help you create richer user experiences for your AJAX-enabled applications.

**Chapter 7—Using the ASP.NET AJAX Control Toolkit**

In Chapter 7, you learn about the controls that are available in the ASP.NET AJAX Control Toolkit, which is an open-source, community-based suite of controls for ASP.NET AJAX.

**Chapter 8—Building an ASP.NET AJAX Extender Control**

Chapter 8 first introduces the concept of creating AJAX-enabled extender controls, and then shows you how to build an extender control that can be used in your applications.

**Chapter 9—ASP.NET AJAX and SharePoint 2007**

In Chapter 9, you learn how to enable ASP.NET AJAX in SharePoint 2007 and Windows SharePoint Services (WSS) 3.0, and then you learn how to build an AJAX-enabled WSS web part.

**Chapter 10—Creating ASP.NET AJAX-Enabled Vista Sidebar Gadgets**

In Chapter 10, you learn how to build ASP.NET AJAX-Enabled Gadgets for the Windows Vista Sidebar.

## CHAPTER 1

# Introduction to AJAX Technologies

If you've purchased this book, you probably are interested in AJAX technologies. If you are not familiar with the technology or are new to AJAX, it is important that you take some time and understand where AJAX fits into the big picture of web development. This section helps bring you up to speed with AJAX's place in your web development toolbox.

One problem with the design of web pages (especially ASP.NET web pages) is that to refresh the data on the page, it must postback to the server. This causes the page to flicker while the page is posting, and then the page is reloaded in the browser with the results of the post. You can view the amount of data with tools such as IEHTTPHeaders and HTTPWatch. You will quickly notice that the amount of information getting posted is quite sizeable because ASP.NET applications not only postback the controls, but also post the page's ViewState. Although the technique of a postback works, it creates a lot of traffic over the wire and inherently reduces the overall scalability of your application.

Asynchronous JavaScript and XML (AJAX) is a development pattern that you can use to provide your users with a much richer user experience in your web applications. Simply stated, AJAX allows you to asynchronously load data into pieces of a page on demand instead of loading the whole page each time data is requested.

An example of where a user experience can be enhanced by AJAX is a web page that contains related dropdown boxes. For example, say that you have a web page that supplies information on cars in which a user must select the year, make, and model of his car from dropdown boxes on the

### IN THIS CHAPTER

- ▶ AJAX and Web 2.0
- ▶ Why Use AJAX?
- ▶ AJAX: An Example
- ▶ Using the AJAX Library

page. When a user selects a year, she populates all makes for a selected year in the make dropdown box. Additionally, when a user selects a make of car, the models dropdown is populated with models for a given year and make.

If you use traditional ASP.NET without AJAX, you will probably set the `AutoPostBack` property of your dropdowns to true and the page will postback (and flicker) when your user makes selections on the page.

Conversely, if you use AJAX, data can be loaded asynchronously into the list boxes as the user makes selections on the page. This is much more efficient because only the data being requested will travel in the request to the server, which could be as simple as a query string appended to the end of a page request. Also, the page will not flicker as the user makes selections in the dropdowns because the post actually is happening in the background.

#### NOTE

An example of AJAX is described later in the chapter.

## AJAX and Web 2.0

Web 2.0 is a term (or rather buzzword) that you often hear when describing most “modern” web sites; however, it shouldn’t be a new concept to web developers. Web 2.0 is actually a consolidation of many existing technologies that allows you to provide a rich interactive user experience over the web. Examples of Web 2.0 technologies include, but aren’t limited to, the following areas:

- ▶ Rich Internet Applications (RIAs), which include AJAX, Adobe Flash, Silverlight, and Moonlight
- ▶ Web services
- ▶ Blogs
- ▶ Wikis
- ▶ Social networking
- ▶ Social bookmarking
- ▶ RSS/Atom

Before the Web 2.0 movement began on the Internet, web pages often focused solely on providing the user with data. The user would simply request a page, view the page, request another page, view that page, and so on.

In contrast, the patterns and techniques behind Web 2.0 are all about the user experience with the web: AJAX and web services for rich, efficient user experiences, blogs, wikis, social networking, and social bookmarking for collaboration, and RSS/Atom so that users can “subscribe” to data.

As technologies such as AJAX evolve and are adopted in large scale on the web, Web 2.0 techniques are quickly becoming the expected user experience for the web. Mainstream examples of AJAX include the Google-based applications, such as Google's Maps, Docs, and Calendar, as well as Microsoft-based applications, such as Hotmail, and Windows Live-based applications. As users start utilizing these types of applications in their everyday lives, they will come to expect the same type of functionality in the applications you develop.

## Why Use AJAX?

As stated previously in this chapter, you can use AJAX to help provide a rich user experience. Of all the new, cool techniques and technologies that are available in Web 2.0 and rich Internet applications, AJAX is clearly the most widely used today. Should you use AJAX simply because it's cool? The short answer is "No," which is explained in the next section.

### AJAX Rationale

Although the user experience that results from AJAX development patterns is a much richer experience, the rationale for utilizing AJAX is the total amount of traffic that is reduced from users accessing your web pages. When you have a web application with a large user base, using AJAX significantly increases the scalability of the application due to the amount of web traffic that is reduced every time a page is loaded.

The following list gives advantages and disadvantages of AJAX:

#### Advantages

- ▶ Reduced page bandwidth
- ▶ Load page data on demand
- ▶ Service-based approach to web development
- ▶ Rich user experience

#### Disadvantages

- ▶ Unexpected browser functionality (when the user clicks the back or refresh button)
- ▶ Business logic can exist in the JavaScript
- ▶ Difficult to test with automated tools

Although this book highlights and focuses on the advantages of AJAX, several disadvantages should be noted. The first thing your users will notice is that when they click the refresh or back button in the browser, the page has a possibility of losing its state when the page reloads. This is a result of dynamically loading data on demand (which was one of the advantages of AJAX).

The second disadvantage of AJAX that you will see in the development world is actually JavaScript. You might wonder that if AJAX is initiated from client-side JavaScript, how can JavaScript be a disadvantage? The answer is that JavaScript eventually will be considered a disadvantage for AJAX in your web applications. Over time, you will be quite shocked at the amount of JavaScript that is required (which is an evolution throughout the lifecycle of your application) for AJAX functionality in your application. It will start out very small and clean, but can quickly spiral out of control.

From the architecture perspective, there is not a clean way to reverse-engineer JavaScript into an architecture model. An architecture document will probably define how a page should function, and there could be thousands of lines of JavaScript behind the scenes to enable this functionality. Depending on your environment and Software Development Lifecycle (SDLC), this could become a problem because it is very difficult to validate the JavaScript against the architecture models.

The third (and certainly not the last) disadvantage is the potential difficulty of testing AJAX functionality with automated tools. For example, Visual Studio Team Suite (VSTS) 2005 had limited support for AJAX when creating web tests. This problem was easily circumvented with tools such as Fiddler (<http://www.fiddlertool.com/fiddler/>), which helps you capture AJAX requests, which then can be loaded into the VSTS Web Test. It is important to note that these issues have been resolved in VSTS 2008.

This section is designed not to scare you away from AJAX, but to create an awareness of things that can affect you and your AJAX development experience. Best practices and warnings are highlighted in upcoming chapters so that you can be aware of the pitfalls that can show up in your applications.

In the next section, you learn the basics of implementing AJAX in your applications.

## AJAX: An Example

Now that you understand the advantages and disadvantages of AJAX, it is helpful to learn about the code behind AJAX requests. This section introduces an example of using AJAX and JavaScript Object Notation (JSON).

### The XMLHttpRequest Object

The main object behind AJAX is the XMLHttpRequest object. This object exposes functionality that allows you to send and receive data (usually XML-based, but not required) from client-side JavaScript code to a server page.

XMLHttpRequest has six methods, listed in Table 1.1, and seven properties, listed in Table 1.2.

Deep knowledge of the capabilities of the XMLHttpRequest object's properties and methods is required learning for any AJAX developer. At its core, every library that encapsulates AJAX functionality (ASP.NET AJAX, AJAX.NET, Yahoo YUI, and so on) uses the XMLHttpRequest object.

The next section defines a simple AJAX JavaScript library that can be used to make AJAX calls in your web applications.

TABLE 1.1 XMLHttpRequest Methods

Method	Parameters	Description
Abort		Aborts the request that has been made.
getAllResponseHeaders		Returns a string of all response headers.
getResponseHeader	headerName	Returns the value of a given response header.
Open	method, url method, url, async method, url, async, username method, url, async, username, password	Opens the request to a specified URL. method defines the HTTP method that will be used (typically GET or POST). url defines the location of the page that is being requested. async is a Boolean that defines whether the request will be made asynchronously or not.
Send	content DOMString Document	Sends the request to the server. DOMString is the XML that is to be sent as a string. Document is an XML DOM document
setRequestHeader	label, value	Allows you to add a label and value to the HTTP header that is being sent to the server.

TABLE 1.2 XMLHttpRequest Properties

Property	Description
onreadystatechange	Event handler that is fired when the readyState property changes. This will be a JavaScript function that handles the data that is returned to the client.
readyState	Returns the state of the XMLHttpRequest object. Possible values are 0 = not initialized 1 = open 2 = HTTP Headers received 3 = receiving 4 = loaded
responseText	Returns the complete response as a string.
responseXML	Returns the complete response (if it is XML) as an XML document object.
responseBody	Returns a binary encoded string of the response.
Status	Returns the HTTP status code of the request. HTTP status codes are defined at <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html</a> .
statusText	Returns the HTTP status description of the request.

## A Simple AJAX Library

As a developer, you probably have several (possibly hundreds of) scripts stuffed away somewhere that you reuse to accomplish specific tasks. One such script you can use for making AJAX calls is provided in Listings 1.1 through 1.3.

### NOTE

Although this library can be written many different ways, this example contains three JavaScript functions: `createAjaxRequest`, `createHttpRequestObject`, and `getResponse`. Each function is separated into its own listing and described in the next three sections.

LISTING 1.1 createAjaxRequest Function

```
//Section 1
var http_request = false;

function createAjaxRequest(url, parameters) {
```

```
http_request = createHttpRequestObject();

if (!http_request) {
    alert('Error creating XMLHttpRequest object');
    return false;
}
http_request.onreadystatechange = getResponse;
http_request.open('GET', url + parameters, true);
http_request.send(null);
}
```

---

This section contains the `createAjaxRequest` function. This method is called when you want to invoke an AJAX request. When this method is invoked, it first makes a call to `createHttpRequestObject` (described in Listing 1.2) to create an instance of the `XMLHttpRequest` object. After the object is returned, you need to do some error checking around this object to ensure that it was created correctly. Next (and most importantly), the object is configured by setting the `onreadystatechange` property to the `getResponse` function handler (described in Listing 1.3), and the request is opened (as asynchronous) and finally sent to the server.

#### LISTING 1.2 `createHttpRequestObject` Function

---

```
//Section 2
function createHttpRequestObject(){
    var request;
    if (window.XMLHttpRequest) { // IE7, Mozilla, Safari, etc.
        request = new XMLHttpRequest();
        if (request.overrideMimeType) {
            request.overrideMimeType('text/xml');
        }
    } else if (window.ActiveXObject) { // IE 6 and previous
        try {
            request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
    return request;
}
```

---

This section contains a helper function called `createHttpRequestObject`. Unfortunately, depending on the type of browser that you are using, there are many ways that you can

(or should) create an instance of XMLHttpRequest. This method creates and returns the object instance of XMLHttpRequest that your browser supports.

---

LISTING 1.3    `getResponse` Function

---

```
//Section 3
function getResponse() {
  if (http_request.readyState == 4) {
    if (http_request.status == 200) {
      var xmlDoc = http_request.responseXML;
      alert(xmlDoc.xml);
    } else {
      alert('There was a problem with the request.');
```

---

Although `createAjaxRequest` and `createHttpRequestObject` are somewhat boilerplate functions, the processing of the response occurs in the `getResponse` function, defined in Listing 1.3, `getResponse`.

The handler to execute this function was set up in the Section 1 `createAjaxRequest` method by setting the `onreadystatechange` property of the XMLHttpRequest object. This function gets fired *every time* the `readyState` property changes. It important to note that you will not be able to access the response until the `readyState` property is equal to 4, or loaded, so you first must check the status of the `readyState` before processing the response.

After your response has been loaded, you must perform another check on the HTTP status. If anything is wrong with your request (for example, 401-unauthorized, 404-page not found, and so on), then you can respond to that error here. In the example in Listing 1.3, the status is checked for a value of 200 (or OK), and then the function will process the request.

As soon as you make these two checks on the `readyState` and `status` properties of the XMLHttpRequest object, you can then process the response. In this example, the response is loaded into an XML Document object, and then the XML is displayed in a message box to help confirm the results.

Listing 1.4 is a complete code listing for all of the code described in this section.

---

LISTING 1.4 Simple AJAX Library—Complete Listing

---

```
var http_request = false;

function createAjaxRequest(url, parameters, callbackFunction) {
  http_request = createHttpRequestObject();
```

```
if (!http_request) {
    alert('Error creating XMLHttpRequest');
    return false;
}
http_request.onreadystatechange = getResponse;
http_request.open('GET', url + parameters, true);
http_request.send(null);
}

function createHttpRequestObject(){
    var request;
    if (window.XMLHttpRequest) { // IE7, Mozilla, Safari, etc.
        request = new XMLHttpRequest();
        if (request.overrideMimeType) {
            request.overrideMimeType('text/xml');
        }
    } else if (window.ActiveXObject) { // IE 6 and previous
        try {
            request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
    return request;
}

function getResponse() {
    if (http_request.readyState == 4) {
        if (http_request.status == 200) {

            var xmlDoc = http_request.responseXML;
            alert(xmlDoc.xml);
        } else {
            alert('There was a problem with the request.');
```



---

**NOTE**

You will most likely process the XML document and not display it in an alert. An example of processing the returned XML is explained in the next section.

---

## Using the AJAX Library

After you understand the basics of AJAX, you can start applying this knowledge to your projects. In this section, you learn how to use AJAX to connect two dropdown list boxes in ASP.NET.

This example allows you to select a make of car (BMW, Mercedes, or Porsche) and use AJAX to request the models for the selected car make. The very simple page is illustrated in Figure 1.1.

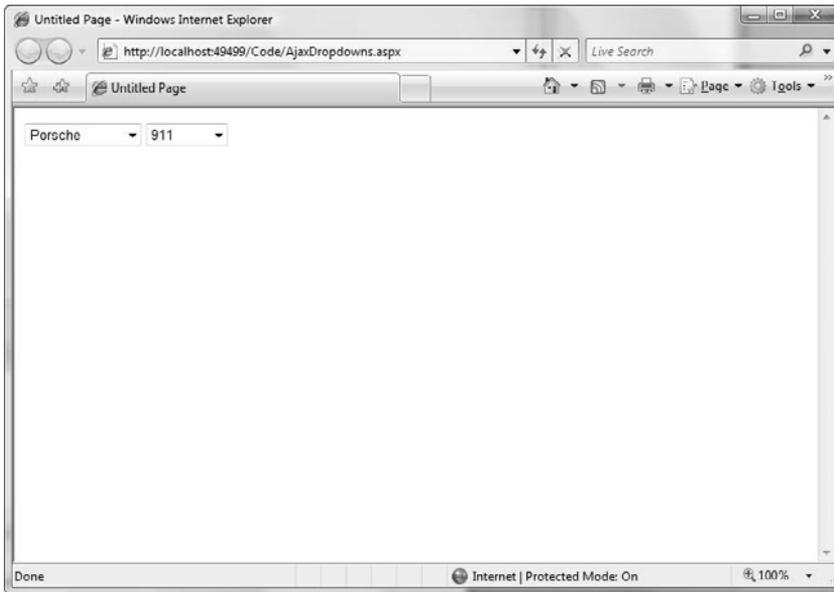


FIGURE 1.1 AJAX dropdowns

When you select a car from the first dropdown, an AJAX request is made to the server to get different car models. For this example, a call is made to a page called `getModels.aspx`, which is defined in Listing 1.4.

---

### LISTING 1.4 `getModels.aspx`

```
<%@ Page Language="C#" %>

<script runat="server">

    protected void Page_Load(object sender, EventArgs e)
    {
        Response.ContentType = "text/xml";
    }
</script>
```

```
string Make = Request.QueryString["make"];
switch (Make.ToUpper())
{
    case "BMW":
        Response.Write("<models><car>3-Series</car>
<car>5-Series</car><car>7-Series</car><car>M</car><car>X3</car>
<car>X5</car><car>Z-Series</car></models>");

        break;

    case "MERCEDES":
        Response.Write("<models><car>C-Class</car>
<car>E-Class</car><car>S-Class</car><car>AMG</car></models>");

        break;

    case "PORSCHE":
        Response.Write("<models><car>911</car><car>Cayman</car>
<car>Cayenne</car><car>Boxster</car></models>");

        break;

    case "BLANK":
        Response.Write("<models><car></car></models>");
        break;

    default:
        Response.Write("<models><car>Invalid selection</car></models>");

        break;
}
}
</script>
```

---

This page simply accepts a query string argument called `make` and will return an XML string of car models, depending on the `make` argument. A sample of the returned XML where the value `PORSCHE` is passed into the `make` query string argument is described in Listing 1.5.

## LISTING 1.5 Models XML Records

---

```

<models>
  <car>911</car>
  <car>Cayman</car>
  <car>Cayanne</car>
  <car>Boxter</car>
</models>

```

---

**NOTE**

Note that this syntax is important, because you will learn how to parse the XML document and populate the second dropdown with the values that are returned from this page.

---

Now that you have learned where the data is coming from, it is time to learn how to request and manipulate the data using AJAX. Listing 1.6 describes a page named `AjaxDropdowns.aspx`.

LISTING 1.6 `AjaxDropdowns.aspx`


---

```

<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

    protected void Page_Load(object sender, EventArgs e)
    {
        //setup the AJAX call on the dropdown
        ddlMakes.Attributes.Add("onchange",
"javascript:createAjaxRequest('getModels.aspx', '?make=' + " + ddlMakes.ClientID +
".value);");

    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>

</head>
<body>
    <form id="form1" runat="server">

```

```
<div>

    <asp:DropDownList ID="ddlMakes" runat="server">
        <asp:ListItem Value="blank">Select a Make</asp:ListItem>
        <asp:ListItem>BMW</asp:ListItem>
        <asp:ListItem>Mercedes</asp:ListItem>
        <asp:ListItem>Porsche</asp:ListItem>
    </asp:DropDownList>
    <asp:DropDownList ID="ddlModels" runat="server">
    </asp:DropDownList>
</div>
</form>
<script type="text/javascript">
//Section 1
var http_request = false;

function createAjaxRequest(url, parameters) {
    http_request = createHttpRequestObject();

    if (!http_request) {
        alert('Error creating XMLHttpRequest object');
        return false;
    }
    http_request.onreadystatechange = getResponse;
    http_request.open('GET', url + parameters, true);
    http_request.send(null);
}

//Section 2
function createHttpRequestObject(){
    var request;
    if (window.XMLHttpRequest) { // IE7, Mozilla, Safari, etc.
        request = new XMLHttpRequest();
        if (request.overrideMimeType) {
            request.overrideMimeType('text/xml');
        }
    } else if (window.ActiveXObject) { // IE 6 and previous
        try {
            request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
}
```

LISTING 1.6 Continued

---

```

        return request;
    }

    //Section 3
    function getResponse() {
        if (http_request.readyState == 4) {
            if (http_request.status == 200) {
                //get dropdown
                var ddl = document.forms[0].<%=this.ddlModels.ClientID %>;

                //clear items in dropdown
                ddl.options.length = 0;

                //get the XML response and loop elements
                var xmldoc = http_request.responseXML;
                var cars = xmldoc.getElementsByTagName('car');
                for(i=0; i<cars.length; i++){
                    //add items to the dropdown
                    try{
                        var opt = new
Option(cars[i].firstChild.nodeValue, cars[i].firstChild.nodeValue);
                        ddl.add(opt, i);
                    }
                    catch (e){
                        ddl.length=0;
                    }
                }
                http_request = null;
            }
            else {
                alert('There was a problem with the request.');
```

```

        }
    }
</script>
</body>
</html>
```

---

As you browse through the code in Listing 1.6, first notice that it is using the simple AJAX library that you learned about earlier in this chapter. You learn about the changes that were made to the `getResponse` function later in this chapter, but it is important to note because you will also see calls made to the `createAjaxRequest` function, which is discussed next.

This page contains two dropdowns, `ddlMakes` and `ddlModels`, and one server side event handler, `Page_Load`. When the user selects a car make from the `ddlMakes` dropdown, an AJAX call needs to be made to populate the `ddlModels` dropdown with car models. In this example, the AJAX call is set up on the `ddlMakes` dropdown in the `Page_Load` event by adding an `onchange` attribute to the server side dropdown list control and setting its value to call the `createAjaxRequest` JavaScript method defined in the simple AJAX library.

The big change to the AJAX library is in the `getResponse` method, which is highlighted in Listing 1.7.

---

**LISTING 1.7** `getResponse` Method

---

```
function getResponse() {
    if (http_request.readyState == 4) {
        if (http_request.status == 200) {
            //get dropdown
            var ddl = document.forms[0].<%=this.ddlModels.ClientID %>;

            //clear items in dropdown
            ddl.options.length = 0;

            //get the XML response and loop elements
            var xmlDoc = http_request.responseXML;
            var cars = xmlDoc.getElementsByTagName('car');
            for(i=0; i<cars.length; i++){
                //add items to the dropdown
                try{
                    var opt = new
Option(cars[i].firstChild.nodeValue, cars[i].firstChild.nodeValue);
                    ddl.add(opt, i);
                }
                catch (e){
                    ddl.length=0;
                }
            }
            http_request = null;
        }
        else {
            alert('There was a problem with the request.');
```

---

As you walk through the method, first, a reference is made to the `ddlModels` dropdown on the page, and all items are cleared by setting the `options.length` property to zero. If you

don't do this, items will simply get added to the dropdown list every time the user selects a new car make. Next, the XML response is returned from the `http_request` object. Finally, the code loops through the values of all the returned `<car>` elements and populates the `ddlModels` dropdown list.

When you execute the page, you will notice that the dropdown is populated with data very quickly with no page flicker. This is a great first example of using AJAX because linked (or related) dropdowns are a common scenario that you see in applications. Additionally, functionality such as this is often a developer's first experience with AJAX functionality in their applications.

Before ASP.NET AJAX (and other libraries), this was how we, as a development community, had to write AJAX functionality. One of the first things you will notice about this approach is the amount of JavaScript that you have to write and maintain. It quickly becomes very difficult to maintain and add or change the functionality of your pages without starting from scratch. There is nothing wrong with this approach, and libraries such as ASP.NET AJAX shield you from a lot of the coding and JavaScript maintenance (and the libraries *are* generating this code for you). However, it is important to note the maintenance cost associated with manually writing AJAX code.

## Summary

In this chapter, you learned about AJAX and its place in the Web 2.0 world. AJAX has a place in your web applications, and the rationale of utilizing AJAX was discussed in detail as well as the advantages and disadvantages that you will encounter by applying this type of functionality in your applications. Finally, you learned how to build AJAX from scratch using ASP.NET and JavaScript.

# Index

## A

- Abort method, 9
- Accordion control, 102, 106-110
- AccordionPane control, 102, 106-110
- activating SharePoint features, 237
- add\_abortPostBack method, 94
- add\_beginRequest method, 94
- add\_dispose method, 94
- add\_endRequest method, 94
- Add Gadgets dialog window, 187-188
- add\_getInstance method, 94
- add\_InitializeRequest method, 94
- add\_pageLoaded method, 94
- add\_pageLoading method, 94
- AddNumbers function, 41
- AddProperty method, 158-159
- \$addHandler function, 51
- \$addHandlers function, 51
- advantages of AJAX, 7-8
- AJAX Client Behavior web site item, 25
- AJAX Client Control web site item, 25
- AJAX Client Library web site item, 25
- AJAX Control Toolkit, 26-27
- AJAX-enabled WCF Service web site item, 26
- AJAX Master Page web site item, 25
- AJAX Web Form web site item, 25
- AJAX Web Forms, 32-34
- AjaxControlToolkit.dll file, 152
- AjaxDropdowns.aspx page, 16-18
- AJAXWebServiceProxy.aspx web page, 38-39
- AllowCustomErrorsRedirect property (ScriptManager), 34
- AlwaysVisibleControlExtender control, 102
- AnimationExtender control, 102
- Application class, 52
- Apply Styles (Expression Web), 197
- ASP.NET AJAX Control Toolkit. *See specific controls*
- ASP.NET AJAX Server Control Extender projects, 24
- ASP.NET AJAX Server Control projects, 24
- Assemblies element, 230

AssemblyDeployment element, 229  
 assert method, 53  
 AsyncPostBackError event, 36  
 AsyncPostBackErrorMessage property (ScriptManager), 34  
 AsyncPostBackTimeout property (ScriptManager), 34  
 AsyncPostBackTrigger, 78  
 authentication, Forms authentication, 62-65  
 AuthenticationService property (ScriptManager), 34  
 autoformatting Expression Web code, 202-203  
 AutoSize property (Accordion control), 110

## B-C

BeginRequest event, 68-71, 94  
 BeginRequestEventArgs type, 68  
 Behavior class, 66  
 BindGrid method, 89  
 Bounds class, 66

CalendarExtender control, 102, 110-112  
 CalendarExtender.aspx sample page, 110-112  
 CascadingDropDown control, 102  
 CasPermissions element, 230  
 ChildrenAsTriggers property (UpdatePanel control), 78

classes. *See specific classes*  
 \$clearHandler function, 51  
 clearTrace method, 53  
 client API namespaces, 49  
 client-side handlers, adding to server-side controls, 51  
 client-side properties (FocusExtender), 158-159  
 ClientBehavior1.js file (FocusExtender), 155, 159-161  
 ClientBehavior1.resx file (FocusExtender), 155  
 Code Formatting options (Expression Web), 202-203  
 CodePlex site, 213  
 CollapseControlID property (CollapsiblePanelExtender control), 117  
 CollapsePanel.aspx sample page, 112-116

CollapsiblePanelExtender control, 102, 112-117  
 completingRequest event, 54  
 configSettings section (SharePoint/WSS Web.Config file), 170-171  
 configuration files. *See Web.Config file*  
 configuring Visual Studio for ASP.NET AJAX Control Toolkit, 105-106  
 ConfirmButton.aspx sample page, 118-119  
 ConfirmButtonExtender control, 102, 118-119  
 ConfirmText property (ConfirmButtonExtender control), 119  
 Control class, 66  
 control extenders
 

- adding, 152-154
- FilteredTextBoxExtender, 152-153
- FocusExtender project, 154-161
- selecting, 153-154

Control Toolkit. *See specific controls*  
 controls. *See specific controls*  
 converting data types, 57  
 \$create function, 51  
 Create New Web Site dialog, 198-199  
 createAjaxRequest function, 10-11  
 CreateChildNode method, 219  
 createHttpRequestObject function, 11  
 CSS Properties (Expression Web), 196  
 CSS Schemas, 202

## D

data types, 57, 68  
 Debug class, 53  
 Default.aspx page, 87-89, 162-163  
 deploying .NET Framework 3.5 as SharePoint feature
 

- CreateChildNode method, 219
- feature activation, 237
- Feature.xml file, 216-217
- FeatureReceiver class, 214-216, 219-228
- Microsoft.SharePoint.targets file, 231-234
- ModificationEntry structure, 218-219
- scope, 214

- SharePoint Solution Management, 235
  - SolutionConfig.xml elements, 229-230
  - SolutionConfig.xml file, 217-218, 228-229
  - SolutionPackage.ddf file, 230-231
  - STSADM.EXE command-line utility, 234
  - STSDev tool, 213-214
  - deserialize method, 43**
  - disadvantages of AJAX, 7-8**
  - displaying modal windows**
    - ModalWizard.aspx page, 138-144
    - stylesheet, 134-138
  - Document Type Declaration (DTD), 202**
  - DomElement class, 66-67**
  - DomEvent class, 66**
  - downloading AJAX Control Toolkit, 26**
  - dragging XML files into Expression Web, 205-206**
  - DragHandleID property (DragPanelExtender control), 122**
  - DragPanel.aspx sample page, 120-121**
  - DragPanelExtender control, 102, 120-122**
  - dropdown list boxes, connecting**
    - AjaxDropdowns.aspx page, 16-18
    - getModel.aspx page, 14-15
    - getResponse method, 19-20
    - models XML records, 15
  - DropDownControlID property (DropDownExtender control), 124**
  - DropDownExtender control, 102, 122-125**
  - DropShadowExtender control, 103**
  - DTD (Document Type Declaration), 202**
  - DWT files, 200**
  - DynamicPopulateExtender control, 103**
- E**
- elements, 229-230**
  - Employee class, 54, 56**
  - EmployeeSales web part**
    - code listing, 171-174
    - EnsurePanelFix() method, 176-177
    - Monthly Sales list schema, 174-175
    - Monthly Sales sample data, 175
    - rendering in SharePoint Team Site, 174-176
  - Enabled property (Timer control), 89**
  - EnabledViewState property (Timer control), 89**
  - EnablePageMethods property (ScriptManager), 34**
  - EnablePartialPageRendering property (ScriptManager), 34**
  - EnableScriptGlobalization property (ScriptManager), 34**
  - EnableScriptLocalization property (ScriptManager), 34**
  - EnableViewState property**
    - ScriptManager, 34
    - UpdatePanel control, 78
  - EndRequest event, 68-71, 94**
  - EndRequestEventArgs type, 68**
  - EnsurePanelFix() method, 176-177**
  - enumerations, 66**
  - events. See specific events**
  - Exceptions type, 68**
  - ExpandControlID property (CollapsiblePanelExtender control), 117**
  - Expression Web**
    - Code Formatting options, 202-203
    - configuring for ASP.NET AJAX Control Toolkit, 145-150
    - IntelliSense, 204
    - overview, 191
    - tools, 145
    - user interface
      - Apply Styles, 197
      - CSS Properties, 196
      - Folder List, 194
      - illustration, 191
      - Manage Styles, 198
      - Tag Properties, 195-196
      - Toolbox, 196
    - Web pages
      - creating, 192-193
      - dragging XML files into, 205-206
      - standards-based web sites, 198-202
      - XSLT style sheets, generating, 207-209

**ExtenderControl class, 158**  
**ExtenderControl1.cs file (FocusExtender), 155**  
**extenders. See control extenders**

## F

**FadeTransitions property (Accordion control), 110**  
**fail method, 53**  
**Feature.xml file, 216-217**  
**FeatureReceiver class, 214-216, 219-228**  
**Fiddler, 8**  
**files. See specific files**  
**FilteredTextBoxes.aspx page, 125-127**  
**FilteredTextBoxExtender control, 103, 125-127, 152-153**  
**FilterType property**  
    control extenders, 154  
    FilteredTextBoxExtender control, 127  
**\$find function, 51**  
**FocusCSSClass property, 158, 161-163**  
**FocusExtender project, 154-155**  
    client-side properties, 158-159  
    ClientBehavior1.js file, 155, 159-161  
    ClientBehavior1.resx file, 155  
    Default.aspx, 162-163  
    ExtenderControl class, 158  
    ExtenderControl1.cs file, 155  
    TextBoxFocusExtender.cs file, 156-158  
**Folder List (Expression Web), 194**  
**formatting Expression Web code, 202-203**  
**form templates, 32-34**  
**Forms authentication, 62-65**  
**functions. See specific functions**

## G

**Gadget.xml file, 186**  
**Gadgets**  
    adding to Vista Sidebar, 179-181  
    creating, 181-187  
    Gadget.xml file, 186  
    GetRandomNumber.html file, 187

    JavaScript proxy, 183-185  
    web service, 181-183  
    definition, 179  
    GetRandomNumber Gadget  
        Gadget.xml file, 186  
        GetRandomNumber.html file, 187  
        JavaScript proxy, 183-185  
        web service, 181-183  
    testing, 187-188  
**Get Employee Information WCF service, 41-43**  
**\$get function, 51**  
**getAllResponseHeaders method, 9**  
**getModels.aspx page, 14-15**  
**GetRandomNumber Gadget**  
    Gadget.xml file, 186  
    GetRandomNumber.html file, 187  
    JavaScript proxy, 183-185  
    web service, 181-183  
**GetRandomNumber.html file, 187**  
**getResponse method, 12, 19-20**  
**getResponseHeader method, 9**  
**GetScriptDescriptors method, 158**  
**GetScriptReferences method, 159**  
**getSerializedEmployee() method, 61**  
**global namespace, 49-51**

## H-I

**HeaderCssClass property (Accordion control), 110**  
**HoverMenu.aspx sample page, 127-130**  
**HoverMenuExtender control, 103, 127-130**  
    HoverMenu.aspx example, 127-130  
    properties, 130  
**ID property**  
    Timer control, 89  
    UpdatePanel control, 78  
**Init event (Application class), 52**  
**initializeRequest method, 98**  
**initializeBase function, 161**

initializeRequest event, 94  
 InitializeRequestEventArgs type, 68  
 IntelliSense, 204  
 Interval property (Timer control), 89  
 invokingRequest event, 54  
 isAsyncPostBack property (PageRequestManager object), 94

## J

JavaScript Object Notation (JSON), 41  
 JavaScript proxy classes, generating, 183-185  
     AJAXWebServiceProxy.aspx web page, 38-39  
     Get Employee Information WCF service, 41-43  
     JSON (JavaScript Object Notation), 41  
     MyMathService.cs file, 37-38  
     MyMathService.svc file, 37  
     overview, 37  
 JavaScriptSerializer class, 56  
 JSON (JavaScript Object Notation) objects, 41  
     processing with Sys.Serialization, 57-61

## K-L

Key enumeration, 66  
 libraries, simple AJAX library, 11-13  
 lists, connecting dropdown list boxes  
     AjaxDropdowns.aspx page, 16-18  
     getModels.aspx page, 14-15  
     getResponse method, 19-20  
     models XML records, 15  
 ListSearchExtender control, 103  
 load event (Application class), 52  
 LoadPostData method, 35  
 LoadScriptsBeforeUI property (ScriptManager), 34  
 Login function, 65  
 Logout function, 65

## M

Manage Styles (Expression Web), 198  
 MaskedEditExtender control, 103  
 MaskedEditValidator control, 103  
 master/detail UpdatePanel controls, 82  
 methods. *See specific methods*  
 Microsoft Expression Web. *See Expression Web*  
 Microsoft.SharePoint.targets file, 231-234  
 modal windows, displaying  
     ModalWizard.aspx page, 138-144  
     stylesheet, 134-138  
 ModalPopupExtender control, 103, 134. *See also*  
     modal windows, displaying  
 ModalWizard.aspx page, 138-144  
 models XML records, 15  
 ModificationEntry structure, 218-219  
 MouseButton enumeration, 66  
 multiple UpdatePanel controls  
     Default.aspx page, 87-89  
     master/detail UpdatePanel controls, 82  
     ucCustomerOrders.ascx page, 85-87  
     ucCustomers.ascx page, 82-85  
     user control sequence diagram, 81  
 MutuallyExclusiveCheckBoxExtender control, 103  
 MyMathService.cs file, 37-38  
 MyMathService.svc file, 37

## N

namespaces. *See specific namespaces*  
 .NET Framework 3.5, deploying as  
 SharePoint feature  
     CreateChildNode method, 219  
     feature activation, 237  
     Feature.xml file, 216-217  
     FeatureReceiver class, 214-216, 219-228  
     Microsoft.SharePoint.targets file, 231-234  
     ModificationEntry structure, 218-219  
     scope, 214

- SharePoint Solution Management, 235
- SolutionConfig.xml elements, 229-230
- SolutionConfig.xml file, 217-218, 228-229
- SolutionPackage.ddf file, 230-231
- STSADM.EXE command-line utility, 234
- STSDev tool, 213-214

**New Page dialog, 192**

**NoBot control, 103**

**NoFocusCSSClass property, 158, 161-163**

**NumericUpDownExtender control, 103**

## O-P

objects. *See specific objects*

**OnAsyncPostBackError method, 35**

**OnButtonClick function, 40**

**onGetEmployeeClick function, 43**

**onreadystatechange property (XMLHttpRequest object), 10**

**OnResolveScriptReference method, 35**

**OnSucceeded function, 40**

**OnTick property (Timer control), 89**

**Open method, 9**

**pageLoad function, 51**

**pageLoaded event, 94**

**PageLoadedEventArgs type, 68**

**pageLoading event, 94**

**PageLoadingEventArgs type, 68**

**PageRequestManager object**

- events, 68-71, 93-94

- methods, 93-94

- overview, 93

- properties, 93-94

- sample application, 95-99

**PagingBulletedListExtender control, 103**

**partial-page rendering, 43-47**

- definition, 23

- PartialPageRendering.aspx page, 44-46

- products\_sel stored procedure, 44

- Timer control, 90-92

**PartialPageRendering.aspx page, 44-46**

**PasswordStrength control, 103**

**Point class, 66**

**PopupControlExtender control, 103**

**PopupControlID property (HoverMenuExtender control), 130**

**PopupPosition property (HoverMenuExtender control), 130**

**PostBackTrigger, 79**

procedures, products\_sel, 44

processClientClick function, 51

processing JSON objects with Sys.Serialization, 57-61

Products.xml file, 205

Products.xsl file, 207-209

products\_sel stored procedure, 44

ProfileService property (ScriptManager), 34

projects, 24

## Q-R

**Rating control, 104**

**readyState property (XMLHttpRequest object), 10**

refreshing UpdatePanel controls with triggers, 79-81

**RegisterArrayDeclaration method, 35**

**RegisterClientScriptBlock method, 35**

**RegisterClientScriptInclude method, 35**

**RegisterClientScriptResource method, 35**

**RegisterDataItem method, 36**

**RegisterDispose method, 36**

**RegisterExpandAttribute method, 36**

**RegisterExtenderControl method, 36**

**RegisterHiddenField method, 36**

**RegisterOnSubmitStatement method, 36**

**RegisterPostBackControl method, 36**

**RegisterScriptControl method, 36**

**RegisterStartupScript method, 36**

**\$remove function, 51**

**rendering, partial-page, 43-47**

definition, 23

PartialPageRendering.aspx page, 44-46

products\_sel stored procedure, 44

Timer control, 90-92

**Render method, 36****RenderMode property (UpdatePanel control), 78****ReorderList control, 104****ResetWebServer element, 229****ResizableControlExtender control, 104****ResolveScriptReference event, 36****responseBody property (XMLHttpRequest object), 10****responseText property (XMLHttpRequest object), 10****responseXML property (XMLHttpRequest object), 10****RoleService property (ScriptManager), 34****RoundedCornersExtender control, 104****S****SafeControlSettings element, 230****saveEmployee() method, 61****scope of SharePoint solutions, 214****ScriptManager control**

AJAX Web Form template, 32-34

events, 36

JavaScript proxy classes, generating

AJAXWebServiceProxy.aspx web page, 38-39

Get Employee Information WCF service, 41-43

JSON (JavaScript Object Notation), 41

MyMathService.cs file, 37-38

MyMathService.svc file, 37

overview, 37

methods, 35-36

overview, 22-23, 31-32

partial-page rendering, 43-47

properties, 34-35

Services element, 40

**ScriptManager method, 35****ScriptManagerProxy control, 23****ScriptMode property (ScriptManager), 35****ScriptPath property (ScriptManager), 35****Scripts property (ScriptManager), 35****SDLC (Software Development Lifecycle), 8****SelectedIndex property (Accordion control), 110****selecting control extenders, 153-154****Send method, 9****server-side controls, 51****services, web, 181-183****Services property (ScriptManager), 35, 40****setLocation method, 68****setRequestHeader method, 9****SharePoint**

and AJAX, 165

deploying .NET Framework 3.5 as SharePoint features

CreateChildNode method, 219

feature activation, 237

Feature.xml file, 216-217

FeatureReceiver class, 214-216, 219-228

Microsoft.SharePoint.targets file, 231-234

ModificationEntry structure, 218-219

scope, 214

SharePoint Solution Management, 235

SolutionConfig.xml elements, 229-230

SolutionConfig.xml file, 217-218, 228-229

SolutionPackage.ddf file, 230-231

STSADM.EXE command-line utility, 234

STSDev tool, 213-214

EmployeeSales web part, 171-177

overview, 211-213

Solution Management, 235

Web.Config file, 166-171

**SliderExtender control, 104****SlideShowExtender control, 104****Software Development Lifecycle (SDLC), 8**

**Solution Management (SharePoint)**, 235  
**SolutionConfig.xml** file, 217-218, 228-230  
**SolutionId** element, 229  
**SolutionName** element, 229  
**SolutionPackage.ddf** file, 230-231  
 standards-based web sites, creating with Expression Web, 198-202  
**Status** property (XMLHttpRequest object), 10  
**statusText** property (XMLHttpRequest object), 10  
 stored procedures, **products\_sel**, 44  
 structures, **ModificationEntry**, 218-219  
**STSADM.EXE** command-line utility, 234  
**STSDev** tool, 213-214  
 style sheets (XSLT), generating with Expression Web, 207-209  
**Sys** namespace, 50-53  
**Sys.Net** namespace, 50, 53-54  
**Sys.Serialization** namespace  
     definition, 50, 54  
     Employee class example, 54-56  
     JavaScriptSerializer class, 56  
     processing JSON objects with, 57-61  
     serialization data type conversions, 57  
**Sys.Services** namespace, 50, 62-65  
**Sys.UI** namespace, 50, 66-68  
**Sys.WebForms** namespace, 50, 68-71

## T

**TabContainer** control, 104  
**TabPanel** control, 104  
**Tag Properties (Expression Web)**, 195-196  
**TargetControlID** property  
     CollapsiblePanelExtender control, 117  
     ConfirmButtonExtender control, 119  
     DragPanelExtender control, 122  
     DropDownExtender control, 124  
     FilteredTextBoxExtender control, 127  
     HoverMenuExtender control, 130  
     TextBoxWatermarkExtender control, 133

**TargetControlType** attribute (control extenders), 158  
 templates, AJAX Web Form templates, 32-34  
 testing Vista Sidebar Gadgets, 187-188  
**TextBoxFocusExtender** control. See FocusExtender project  
**TextBoxFocusExtender.cs** file, 156-158  
**TextboxWatermark.aspx** page, 131-133  
**TextBoxWatermarkExtender** control, 104, 130-133  
**Timer** control, 23, 89-92  
**ToggleButtonExtender** control, 104  
**Toolbox (Expression Web)**, 196  
**ToolkitScriptManager** control, 104  
**trace** method, 53  
**traceDump** method, 53  
**triggers**  
     adding to UpdatePanel controls, 79  
     AsyncPostBackTrigger, 78  
     PostBackTrigger, 79  
     refreshing UpdatePanel controls, 79-81  
**Triggers** property (UpdatePanel control), 78  
**types**, 57, 68

## U

**ucCustomerOrders.aspx** page, 85-87  
**ucCustomers.aspx** page, 82-85  
**unload** event (Application class), 52  
**UpdateCustomerOrders** method, 89  
**UpdateMode** property (UpdatePanel control), 78  
**UpdatePanel** control  
     multiple UpdatePanel controls, 81  
     Default.aspx page, 87-89  
     master/detail UpdatePanel controls, 82  
     ucCustomerOrders.aspx page, 85-87  
     ucCustomers.aspx page, 82-85  
     user control sequence diagram, 81  
 overview, 23, 73-74  
 performance problems, 73  
 properties, 78

- simple example, 74-77
- triggers, 78-81
  - adding, 79
  - AsyncPostBackTrigger, 78
  - PostBackTrigger, 79
  - refreshing UpdatePanel control, 79-81

**UpdatePanelAnimationExtender control, 104**

**UpdateProgress control, 23**

## V

**ValidatorCalloutExtender control, 104**

**Visibility Mode enumeration, 66**

**Visible property (UpdatePanel control), 78**

**Vista Sidebar Gadgets**

- adding, 179-181
- creating, 181-187
- definition, 179
- testing, 187-188

**Visual Studio**

- configuring for ASPNET AJAX Control Toolkit, 105-106
- controls. *See specific controls*
- projects
  - ASPNET AJAX Server Control Extender projects, 24
  - ASPNET AJAX Server Control projects, 24
  - overview, 24
- web site items, 25-26

## W

**WatermarkCSSClass page**  
(TextBoxWatermarkExtender control), 133

**WatermarkText page** (TextBoxWatermarkExtender control), 133

**WCF services, Get Employee Information, 41-43**

**Web 2.0, 6-7**

**web pages**

- adding controls to, 151-152
- creating with Expression Web, 192-193

**web parts, EmployeeSales**

- code listing, 171-174
- EnsurePanelFix() method, 176-177
- Monthly Sales list schema, 174-175
- Monthly Sales sample data, 175
- rendering in SharePoint Team Site, 174-176

**web services**

- creating, 181-183
- Get Employee Information WCF service, 41-43
- JavaScript proxy classes, generating
  - AJAXWebServiceProxy.aspx web page, 38-39
  - Get Employee Information WCF service, 41-43
  - JSON (JavaScript Object Notation), 41
  - MyMathService.cs file, 37-38
  - MyMathService.svc file, 37
  - overview, 37

**web sites**

- creating with Expression Web, 193, 198-202
- web site items, 25-26

**Web.Config file**

- Forms authentication, 62
- SharePoint/WSS Web.Config file, 166-171

**WebRequest class, 53**

**WebRequestManager class, 53-54**

**WebServiceError class, 53**

**WebServiceProxy class, 53**

**windows**

- Add Gadgets dialog window, 187-188
- modal windows, displaying
  - ModalWizard.aspx page, 138-144
  - stylesheet, 134-138

## X-Y-Z

**XML, dragging into Expression Web, 205-206**

**XMLHttpExecutor class, 53**

**XSLT style sheets, generating with Expression Web, 207-209**