Chapter 1

Introducing ASP.NET AJAX

Learning Objectives

After completing this chapter, you will be able to:

- *Understand the use and importance of AJAX.*
- *Know about different AJAX libraries.*
- *Work with ASP.NET AJAX.*



INTRODUCTION

AJAX stands for Asynchronous JavaScript and XML. It is a web technology that helps the users create dynamic, client-centric, user-friendly, and interactive web applications. Before the development of AJAX, most of the web applications performed slowly and were very complex. But, with the introduction of AJAX to the world of web technology, it has made a special impact on it. The AJAX-based applications look like Windows applications and they perform faster. Also, they are more efficient and user-friendly than previous web applications.

Earlier, a user had to wait for a long time to get the response of a request sent to the server. At that time, the user had to click on a link or a button and then wait to get the response from the server, while watching its progress on the status bar of the web page. This process was tedious and time-consuming. AJAX has not only changed this entire process of request and response, but also added various web controls to make a web application powerful.

In this chapter, you will get an overview of AJAX technology and understand its importance in developing web applications. Also, you will become familiar with various components used in AJAX to create interactive web applications. In addition to this, you will know about the advantages and limitations of AJAX.

Uses of AJAX

The introduction of AJAX has changed the concept of web application. Therefore, the AJAX-based applications are preferred over the simple web applications. This technology can be used mainly in the following conditions:

1. When a client or a customer wants his web application to be more interactive, customer-centric, and capable of performing maximum actions with least effort.

2. When the developers want their web pages to be more interactive with less resources.

3. If a client's application consists of data that needs to be changed or updated frequently.

AJAX is a technology that can be applied to a web application. It means it is not an interface, a platform, or a programming language that can be used for creating an application. Basically, it is an extension of the existing web technology that can be used with a platform to build an AJAX-based application. Some of the platforms, where you can use AJAX, are as follows:

1. ASP.NET

2. PHP

3. Java

There are many examples of AJAX-based web applications that are used today. For example, Google uses AJAX in most of its web pages. The Google Map is also based on AJAX technology. Another real example where some controls of AJAX have been used is Yahoo News

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

(http://news.yahoo.com/). In Yahoo's home page, you can find various news links. When you move the mouse pointer on any of these links, a pop-up window is displayed. This happens due to the use of **HoverMenu** control of AJAX. Figure 1-1 shows an example of AJAX-enabled page from Yahoo News.



Figure 1-1 Use of AJAX in a Yahoo page

Components of AJAX

AJAX has various components and some of the importants are listed below.

1. JavaScript
2. XML
3. CSS
4. DOM
5. XmlHttpRequest object
6. ScriptManager class

AJAX is mainly comprised of two languages: JavaScript and XML. In fact, AJAX is entirely written in JavaScript. It consists of various libraries of JavaScript that come into play when you implement AJAX into a web page. XML is mainly used for data transfer between client and server in AJAX.



Note

All components of AJAX will be discussed in detail in the later chapters.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

THE POSTBACK

When a request is sent to the server, the server sends a response of the request back to the client. This entire cycle of request and response is called postback. In other words, postback is a two-way round trip from client to server and vice versa. Postbacks are of two types: Synchronous and Asynchronous. These postbacks are discussed next.

Synchronous Postback

All traditional web applications use synchronous postback. It is also called normal postback. In synchronous postback, the entire contents of the web page are sent to the server, and in response, the server sends the required information to the client. In this type of postback, a user can observe the process of sending request to the server by the flickering of the web page and the progress shown on the status bar. Figure 1-2 shows the diagrammatic representation of a synchronous postback.

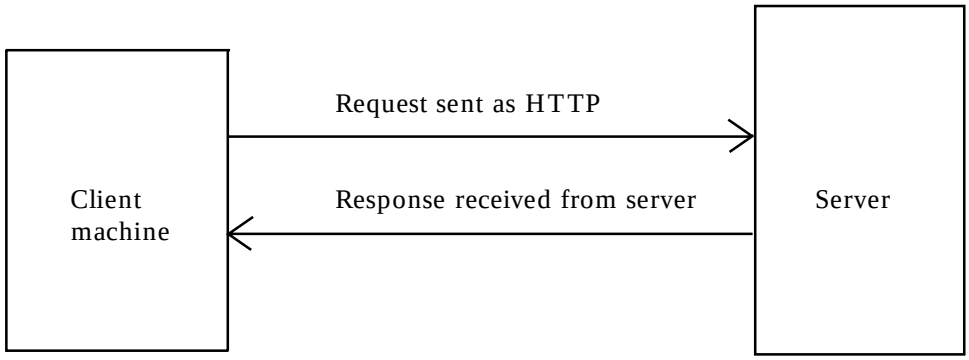

Figure 1-2 The diagrammatic representation of synchronous postback

Asynchronous Postback

You can perform asynchronous postback in a web page with AJAX. In an asynchronous postback, instead of sending the entire contents of a web page to the server, only the request for a specific information is sent to the server, and in response, the server sends the required information to the client. As a result, only a particular portion of the web page is updated with the new information and the rest of the web page remains the same. This process of sending the request for updating a particular portion of a web page, and not the whole page, is called partial page update. In this process, the web page uses the **XmlHttpRequest** object to send the request to the server. The other difference between the synchronous and asynchronous postbacks is that in asynchronous postback, you cannot see the progress on the status bar of a web page while updating it. Also, you cannot observe the flickering of the web page. All AJAX-enabled web pages use asynchronous postback. Figure 1-3 shows the diagrammatic representation of an asynchronous postback.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

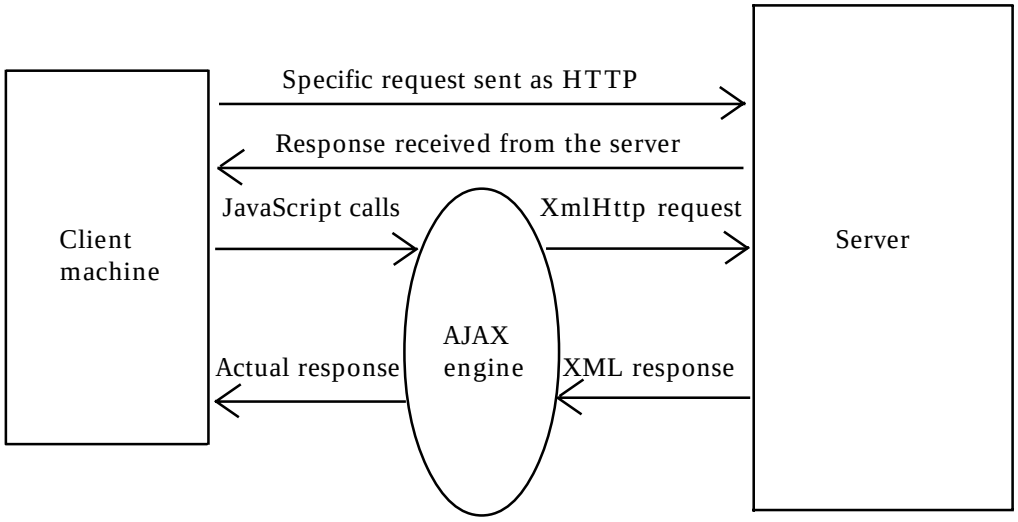

Figure 1-3 The diagrammatic representation of asynchronous postback

ASP.NET AND AJAX

Microsoft Visual Studio has added a new client-server technology in its environment called ASP.NET. This technology is used to create dynamic web applications. It is an excellent environment for creating architecture-based web applications. In this environment, you can create web applications based on high level programming languages such as C#, VB, J#, and so on. After the development of AJAX, Microsoft Visual Studio has added it in its own environment and now you can create AJAX-based applications in this environment as well.

Installation and Configuration Issues of ASP.NET AJAX

In the earlier versions of Visual Studio, you had to add as well as configure AJAX separately in it to develop AJAX-based applications. But, after the release of Visual Studio 2008 and its other new releases, you do not need to install or configure AJAX separately in the existing ASP.NET environment because AJAX is inbuilt in it.

Moreover, in the earlier versions of ASP.NET AJAX, there were some issues that have been addressed in the new versions. For example, in ASP.NET 2.0 AJAX, you would often face error in sys configuration while creating an AJAX-based application. To solve this problem and run the application, you need to add few lines of code in the *web.config* file. Otherwise, the AJAX-based web page will not run. However, with the launch of Visual Studio 2008 and ASP.NET 3.5, this problem has been solved. The ASP.NET 3.5 has included AJAX as an inbuilt component. Additionally, it has many JavaScript libraries to control the AJAX-based applications. Since AJAX is inbuilt in ASP.NET 3.5, you do not need to install or configure it. You can directly create an AJAX-based application in ASP.NET 3.5. Also, you can add more libraries separately to the AJAX environment as per your requirement to create applications.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

AJAX Libraries

AJAX libraries are a set of JavaScript libraries that are used to perform client-side actions to send requests to the server. These libraries have been primarily developed for creating AJAX-enabled web pages. Once you start working in an AJAX-enabled environment, you can use most of the AJAX libraries. Most of these libraries are inbuilt in ASP.NET AJAX and are loaded automatically when you use an AJAX-enabled page. However, you can also add other libraries in your web page to enhance the functionality of the application. Some of the AJAX libraries that are helpful in creating AJAX-based applications are discussed next.

AjaxAnywhere

This library helps you develop an AJAX-based application in JSP framework.

ajaxCFC

This library helps you create an AJAX-based application in ColdFusion framework.

Ajax.NET Professional

This library has been developed by Michael Schwartz. This library helps you create an interactive AJAX-based application.

Anna

This library does not support the general HTTP methods. It supports the text, XML, and JSON files.

Anthem.NET

This library helps you interact with the server in the ASP.NET 1.1 or ASP.NET 2.0 application. This library contains many server controls to help your application work like an AJAX-based application.

Dojo

This is a JavaScript toolkit. It contains client-side libraries that help you develop AJAX-based applications.

DWR

This library helps you create an AJAX-based application in Java framework. This library is highly integrated with Java.

JsHttpRequest

This library helps you create an AJAX-based application in PHP framework.

Prototype

This library works as a JavaScript framework and helps you build an interactive AJAX-based application. This library does not provide adequate support to the web developers for creating a user interface in an application.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Rialto
This library also works as a JavaScript framework and helps you create an AJAX-based application in ASP.NET, Java, or PHP framework.

Rico
The Rico library helps you create an effective interface to work with DOM elements.

SAJAX
The SAJAX library helps you create an AJAX-based application in PHP or Perl framework.

Sardalya
This library is browser-independent and helps you create AJAX-based applications. This library is freely available, but you are recommended not to use it for commercial purpose.

Visual WebGui
This library helps you create AJAX-based Winforms in ASP.NET framework. This library has also a functionality that can be used to create custom controls also.

WebORB
The WebORB library helps you create an AJAX-based application in ASP.NET and Java frameworks.

WORKING WITH ASP.NET AJAX
To start working with ASP.NET AJAX, you need to have Visual Studio 2008 installed on your system. But if you are working on a version that is prior to Visual Studio 2008, you need to install and configure AJAX on your system. Once you have fulfilled all minimum requirements to work with an AJAX-enabled web page, you can start working with it. In the next section, you will learn the basic steps to create an AJAX-enabled web page.

Creating an AJAX-enabled Web Page
The steps to create an AJAX-enabled web page are as follows:

1. Launch Visual Studio 2008.

2. Next, choose **New > Web Site** from the **File** menu, as shown in Figure 1-4. The **New Web Site** dialog box will be displayed, as shown in Figure 1-5.

3. Make sure the **ASP.NET Web Site** icon is selected in the **Templates** area of the **New Web Site** dialog box. Next, create a new folder in your hard drive (here, it is F drive) and name it, for example, *Ch_01*. Now, specify the path of this folder as *F:\AJAX\AjaxProject\Ch_01* in the **Location** drop-down list, as shown in Figure 1-5.

4. Next, choose the **OK** button; the *Default.aspx* page will be displayed with the source code, as shown in Figure 1-6. If the *Default.aspx* page is not displayed by default, you can open it from the **Solution Explorer** window.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

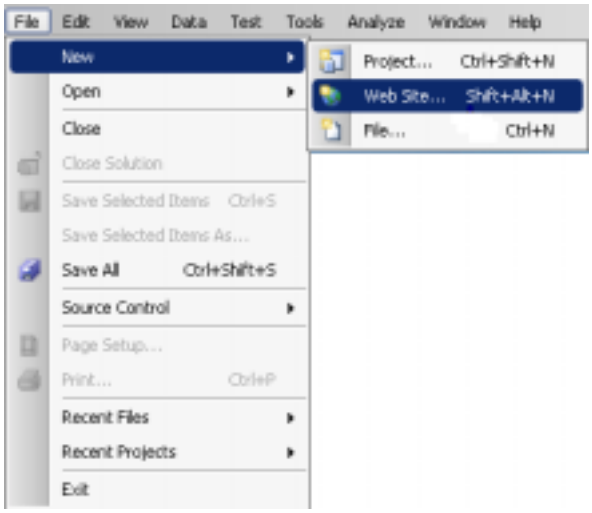

Figure 1-4 Choosing Web Site from the File menu

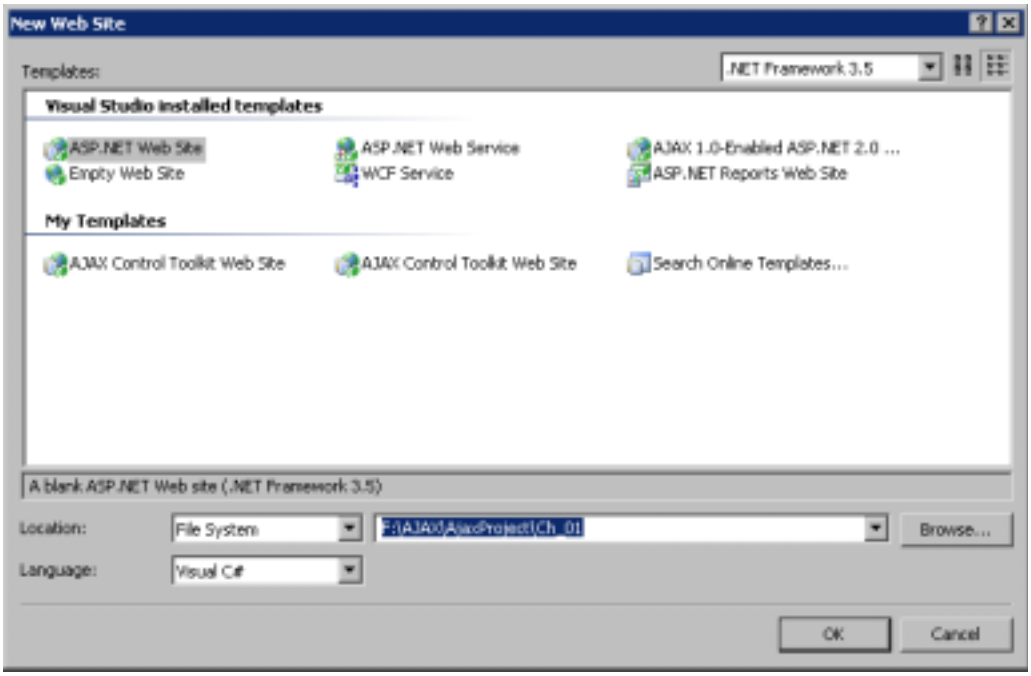

Figure 1-5 The New Web Site dialog box

5. Next, press the F5 key to run the application; the browser will be displayed, as shown in Figure 1-7.

6. Right-click on the web page and then choose the **View Source** option from the context menu displayed, as shown in Figure 1-8; the source code will be displayed in the **Notepad** editor, as shown in Figure 1-9.

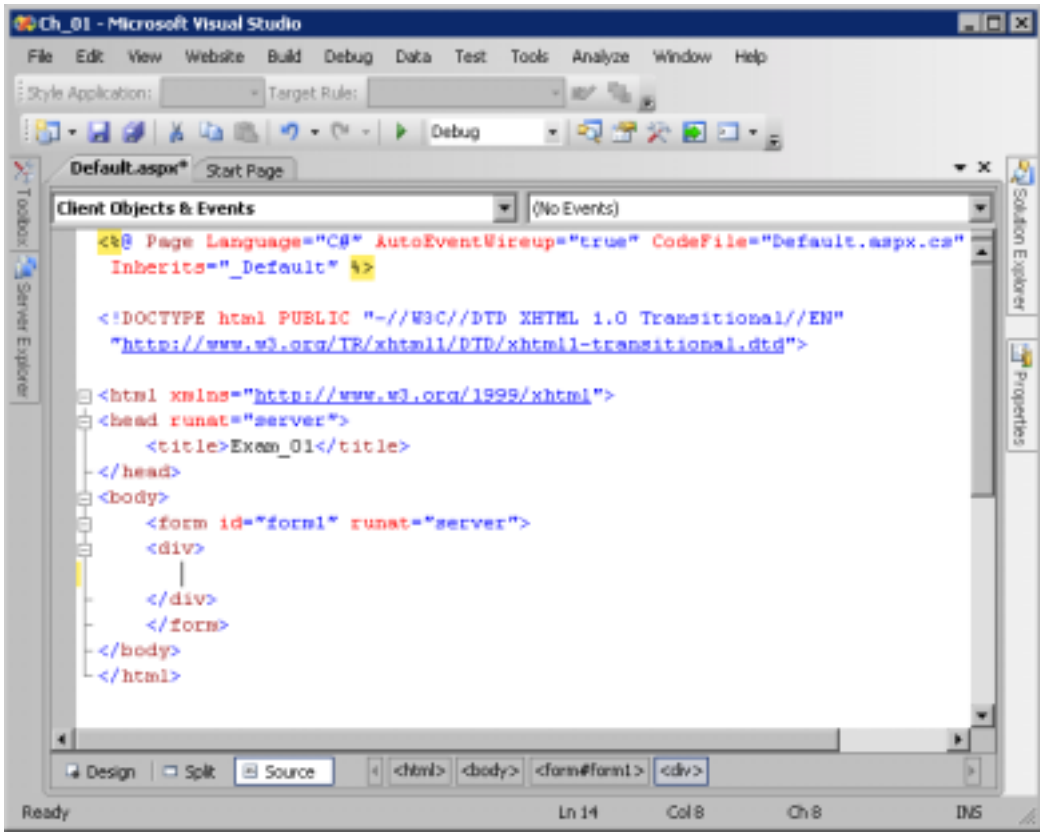

Figure 1-6 The Source window of the Default.aspx page

Note that you have not yet created any AJAX-enabled page. The *Default.aspx* page is a normal web page, which does not contain any AJAX-enabled feature. Also, you will notice that the source code displayed in the **Notepad** editor, refer to Figure 1-9, looks like the source code of a simple *aspx* page. This source code is exactly the same shown in the *Default.aspx* page, refer to Figure 1-6, except the code written for ViewState. Here, ViewState has been used to store information of the web page in the hidden form to maintain the session. Now, perform the following steps:

7. Close the **Notepad** editor, switch to the **Source** window of the *Default.aspx* page, and then write the text **Exam_01** between the starting and closing of the **<title>** tag.

8. Double-click on the **ScriptManager** control in the **Toolbox**, as shown in Figure 1-10, to add it to the *Default.aspx* form.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

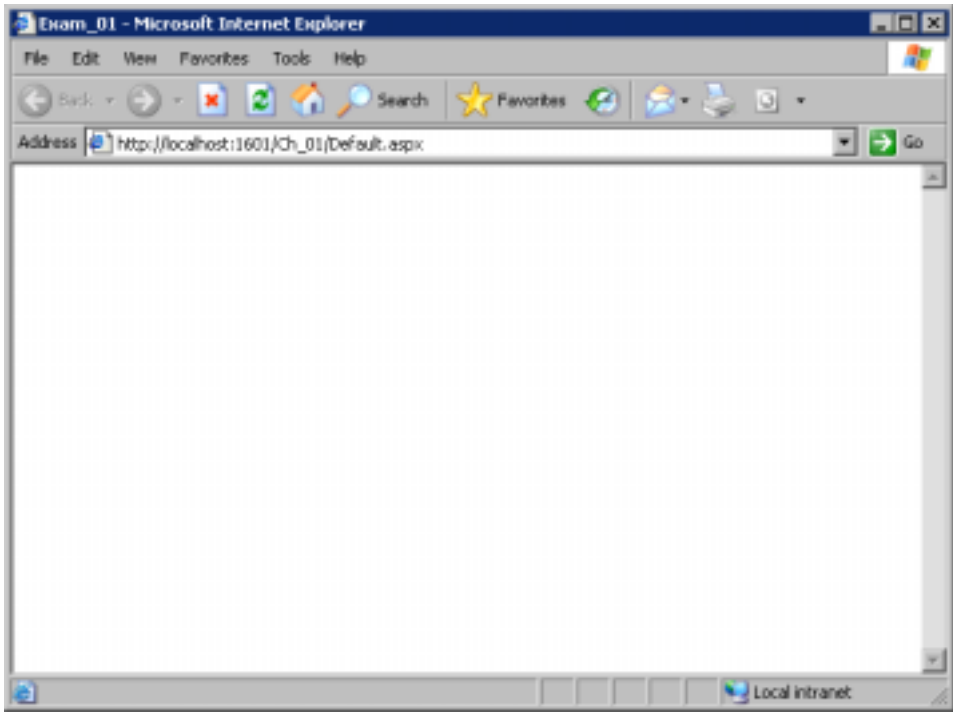

Figure 1-7 The browser after running the web page

9. Now, notice the source code shown in Figure 1-11. Repeat steps 5 and 6 and run the web page again. The new source code will be displayed in the browser as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">                    1
<html xmlns="http://www.w3.org/1999/xhtml">                                   2
<head><title>Exam_01</title></head>                                           3
<body>                                                                        4
<form name="form1" method="post" action="Default.aspx" id="form1">            5
<div>                                                                         6
<input type="hidden" name="__EVENTTARGET"
id="__EVENTTARGET" value="" />                                                7
<input type="hidden" name="__EVENTARGUMENT"
id="__EVENTARGUMENT" value="" />                                              8
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTE3OTkzOTYyMDBkZHQqIefPsw9xOTEdbGkPFD0zKmBZ"/>               9
</div>                                                                        10
<script type="text/javascript">                                               11
//<![CDATA[                                                                   12
var theForm = document.forms['form1'];                                        13
if (!theForm) {                                                               14
theForm = document.form1;                                                     15
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

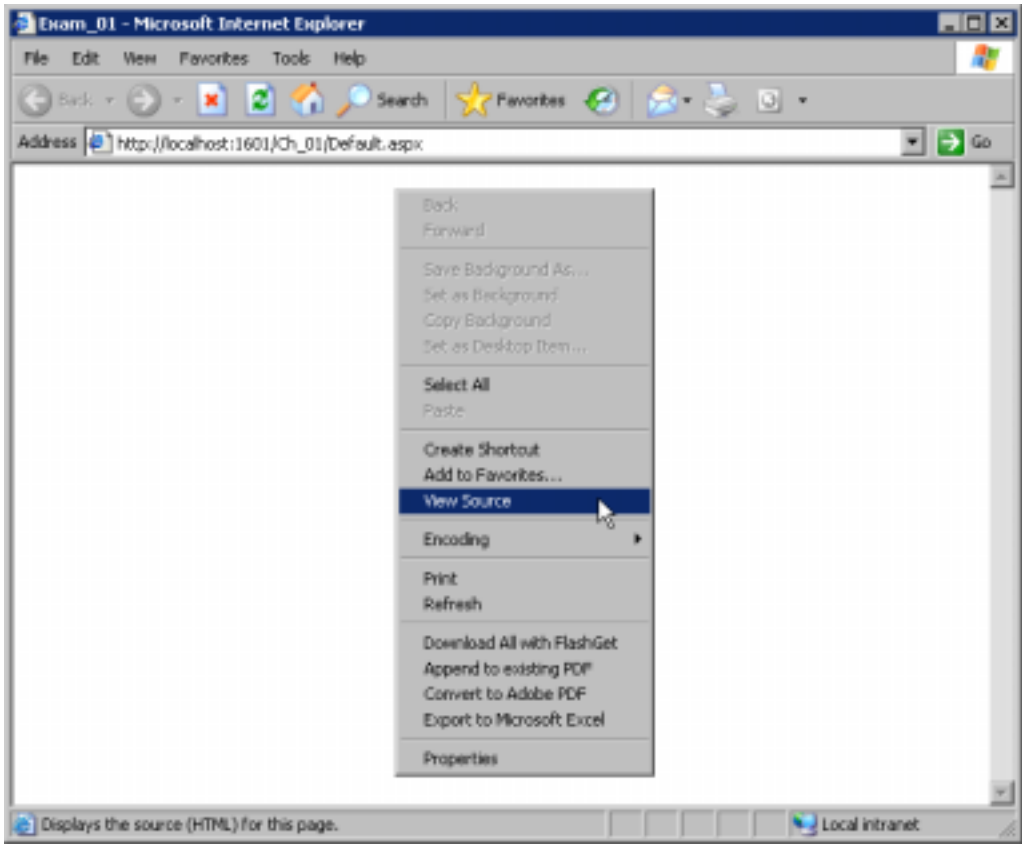

Figure 1-8 Choosing the View Source option from the context menu

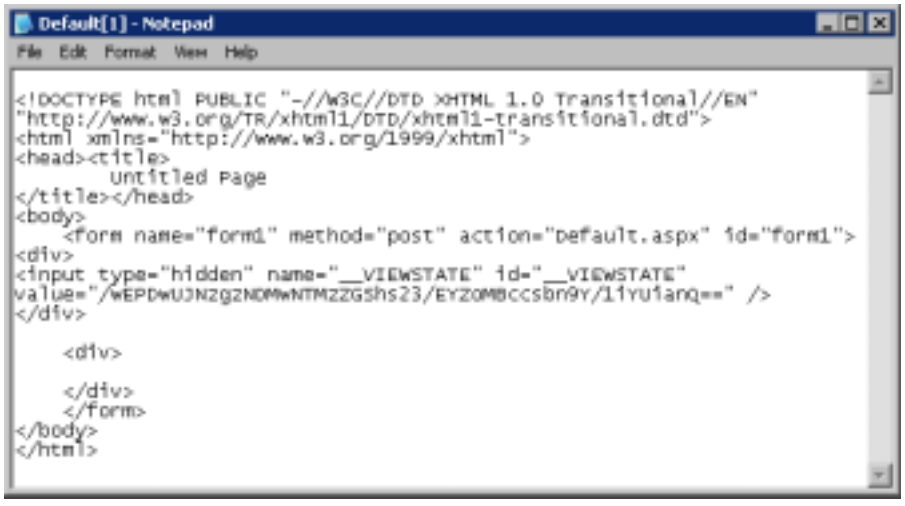

Figure 1-9 The source code displayed in the Notepad editor

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

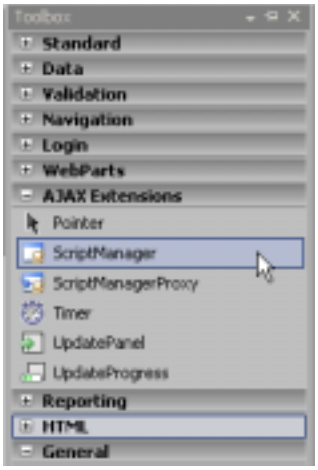

Figure 1-10 Adding the ScriptManager control

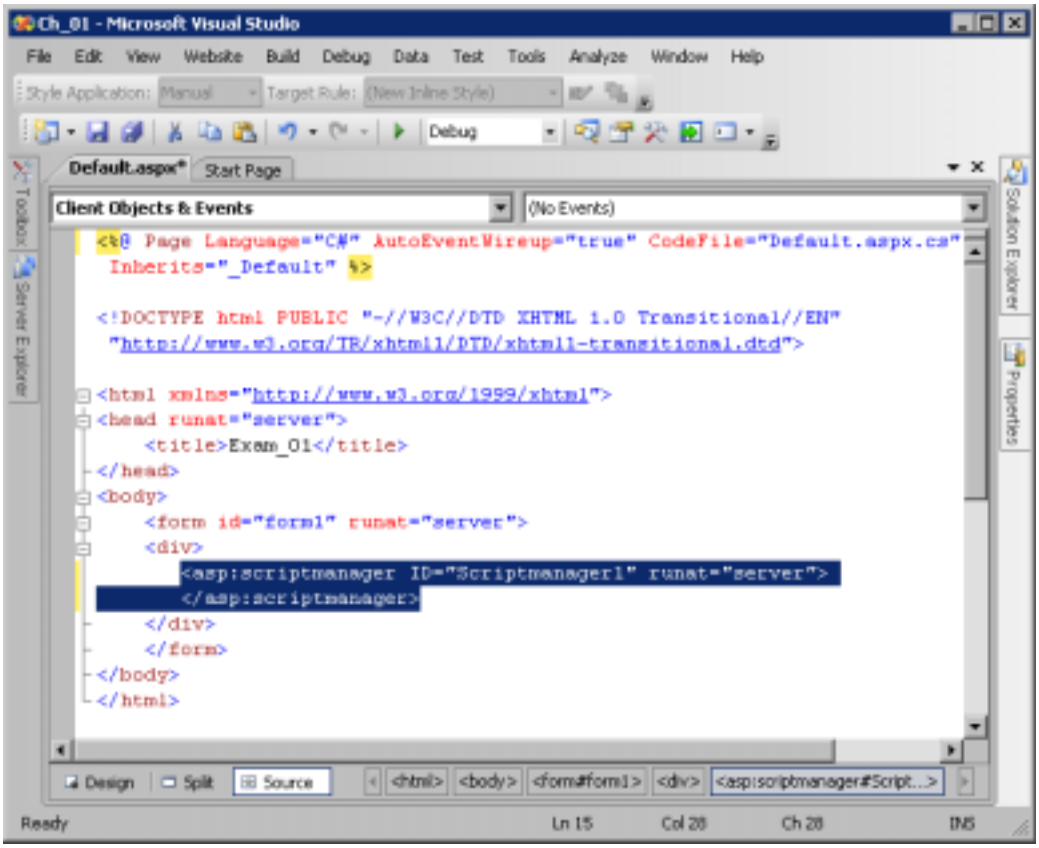

Figure 1-11 The Source window of the Default.aspx page

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
}                                                                          16
function __doPostBack(eventTarget, eventArgument) {                       17
if (!theForm.onsubmit || (theForm.onsubmit() != false)) {                  18
theForm.__EVENTTARGET.value = eventTarget;                                 19
theForm.__EVENTARGUMENT.value = eventArgument;                             20
theForm.submit();                                                          21
}                                                                          22
}                                                                          23
//]]>                                                                      24
</script>                                                                  25
<script src="/Ch_01/WebResource.axd?d=n_xIcrqArUtAzp_4KSLtfQ2&amp;
t=633525162180444495" type="text/javascript">                              26
</script>                                                                  27
<script src="/Ch_01/
ScriptResource.axd?d=sQ0QNid0J8P35MgW5lXdmvNVsHd9OPGsS0agDYDQFg
SR6XXsaCj2CPlk-_xAAi2vzNsKxmkjZdwbVT9IyZewxdTBKI8XbtXNAjpARUFW
7281&amp;t=633614862510000000" type="text/javascript">                     28
</script>                                                                  29
<script type="text/javascript">                                            30
//<![CDATA[if (typeof(Sys) === 'undefined') throw new Error('ASP.NET AJAX
client-side framework failed to load.');                                   31
//]]>                                                                      32
</script>                                                                  33
<script src="/Ch_01/
ScriptResource.axd?d=sQ0QNid0J8P35MgW5lXdmvNVsHd9OPGsS0agDYDQFg
SR6XXsaCj2CPlk-_xAAi2vIy4BjKqTCozeQB_3uVPIYgpd5o4IGpo101jpEFgElYL
N0jjFIlGflZwI_lws_U6v0&amp;t=633614862510000000" type="text/javascript">   34
</script>                                                                  35
<div>                                                                      36
<script type="text/javascript">                                            37
//<![CDATA[                                                                38
Sys.WebForms.PageRequestManager._initialize
('ScriptManager1', document.getElementById('form1'));                      39
Sys.WebForms.PageRequestManager.getInstance()._updateControls([], [], [], 90);  40
//]]>                                                                      41
</script>                                                                  42
</div>                                                                     43
<script type="text/javascript">                                            44
//<![CDATA[                                                                45
Sys.Application.initialize();                                              46
//]]>                                                                      47
</script>                                                                  48
</form>                                                                    49
</body>                                                                    50
</html>                                                                    51
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Explanation

Compare the above source code with the source code shown in Figure 1-9. You will find a lot of differences between them. The above code indicates that it is the code for an AJAX-enabled web page. When you add the **ScriptManager** control to the form, many AJAX libraries will automatically be added to it. Although this web page is a dummy page and cannot respond, this page is AJAX-enabled. It means you can implement this web page as an AJAX-enabled page. If you notice the source code written above carefully, you will see a number of lines of code are written in JavaScript. From the above code, it is clear how the addition of the **ScriptManager** control to the code can make a difference between an ASP.NET page and ASP.NET AJAX page. The **ScriptManager** control will be discussed in detail in the later chapters. The purpose of this example is to explain how an AJAX-enabled web page can be created rather than implementing it.

Basic Controls of ASP.NET AJAX

There are some basic controls used in ASP.NET AJAX such as **ScriptManager**, **ScriptManagerProxy**, **Timer**, **UpdatePanel**, **UpdateProgress**, refer to Figure 1-10. Among these controls, the **UpdatePanel** control is the most important control. It is used to implement partial page updates in a web page. It is discussed in detail in the later chapters.

ADVANTAGES AND LIMITATIONS OF AJAX

Although AJAX has brought a drastic change in the area of web technology yet it has a few drawbacks. You will learn about the advantages and limitations of AJAX in the next section.

Advantages

The major advantages of AJAX are as follows:

Usability

AJAX technology has made the web application quite easy to use. An AJAX-enabled web application looks like a Windows application as it does not make a user wait for a page update or refresh.

Support of JavaScript to AJAX

AJAX has full support of JavaScript. Since JavaScript supports asynchronous programming, it is very useful for creating AJAX-based web applications.

Support of XML to AJAX

XML is the most efficient way to transmit data between a client and server. In AJAX, the data transmitted between client and server is mostly in the XML form. Therefore, this is also an advantage of using AJAX.

Data Manipulation

Using AJAX, you can easily manipulate the data in a web page. For example, in an AJAX-enabled application, you can retrieve, update, delete, or insert data into a web page easily, quickly, and efficiently.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Browser Independency

Apart from Internet Explorer, AJAX supports many browsers such as Mozilla Firefox, Opera, Safari, and so on. The request object for different browsers may be different.

Dynamic Interface

AJAX brings many interactive controls under AJAX Control Toolkit. These controls are used to create dynamic interfaces on a web page. By using these interactive controls, a developer can save a lot of time while designing a web page. You will learn in detail about AJAX Control Toolkit in the later chapters.

Auto Refresh

In many websites, web pages need constant update; for example, updating the score of a live cricket match or updating inputs of stock exchange. AJAX has made creating such applications really easy as the user needs to use less coding than other applications.

Limitations

Some of the limitations of AJAX are as follows:

Browser Issues

One of the limitations of AJAX is that it does not support earlier versions of browsers.

ViewState Management

The postbacks in ASP.NET AJAX affect the ViewState management, therefore, you need to manage it separately.

Back Button and Bookmark Problem

An AJAX-enabled page does not bookmark the previous page. So, you cannot switch back to the previous page by clicking the back button.

Self-Evaluation Test

Answer the following questions and then compare them to those given at the end of this chapter:

1. AJAX stands for _____ .

2. Which of the following is not a component of AJAX?

   (a) JavaScript (b) DOM
   (c) DWR (d) XML

3. Which of the following applications can be created using AJAX?

   (a) Window-based applications (b) Web-based applications
   (c) Console-based applications (d) Database applications

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

4. In which of the following environments can you create an AJAX-based application?

(a) ASP.NET
(b) Java
(c) PHP
(d) All the above

5. AJAX is a web technology that is used to develop web applications. (T/F)

Review Questions

Answer the following questions:

1. _____ is a two-way round trip from client to server and vice-versa in a web page.

2. Which of the following controls is not an AJAX control?

(a) The **UpdatePanel** control
(b) The **UpdateProgress** control
(c) The **GridView** control
(d) The **ScriptManager** control

3. Which of the following components belongs to AJAX?

(a) JavaScript
(b) XML
(c) The **XmlHttpRequest** object
(d) All the above

4. Which of the following is not an advantage of using AJAX?

(a) Support of JavaScript
(b) Support of XML
(c) ViewState Management
(d) Auto Refresh

5. An AJAX-enabled web page follows an asynchronous postback. (T/F)

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Answers to Self-Evaluation Test

1. Asynchronous JavaScript and XML, **2.** c, **3.** b, **4.** d, **5.** T