Quick answers to common problems

ASP.NET jQuery Cookbook

Over 60 practical recipes for integrating jQuery with ASP.NET

Sonal Aneel Allana

[PACKT] PUBLISHING

ASP.NET jQuery Cookbook

Over 60 practical recipes for integrating jQuery with ASP.NET

Sonal Aneel Allana

[PACKT] PUBLISHING

BIRMINGHAM - MUMBAI

ASP.NET jQuery Cookbook

First published: April 2011

Production Reference: 1180411

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-849690-46-1

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author
Sonal Aneel Allana

Reviewers
Tim Cromarty
Buu Nguyen
Aleš Rosina
Hajan Selmani

Acquisition Editor
Steven Wilding

Development Editor
Neha Mallik

Technical Editor
Pallavi Kachare

Indexer
Rekha Nair

Editorial Team Leader
Vinodhan Nair

Project Team Leader
Lata Basantani

Project Coordinator
Leena Purkait

Proofreader
Jonathan Todd

Production Coordinator
Arvindkumar Gupta

Cover Work
Arvindkumar Gupta

About the Author

Sonal Aneel Allana is a Senior Technical Consultant with Accentiv' SurfGold, Singapore. She has over a decade of experience in building web and desktop solutions in Microsoft technologies. She has worked with government agencies, credit bureaus, banks, financial institutions, and multinationals, delivering high-end customized solutions to meet their needs. She has worked on applications ranging from core banking software and B2B links to content management, workflow, and loyalty engines.

From time to time, she has also taken up the role of project lead and has carried numerous projects from conceptualization through implementation successfully. In addition to her experience in Microsoft technologies, she is well versed in J2EE and LAMP platforms.

She holds a Master's degree in Computer Science from the National University of Singapore and a Bachelor's degree in Computer Engineering from Mumbai University. She is also passionate about network and data security and has a certificate in Security Technology and Management from the Institute of Systems Science, National University of Singapore.

I would like to dedicate this book to my father for always believing in me and loving me unconditionally. Dad, I am sorry that you passed away before this book could become a reality.

Special thanks to my mother for her love, sacrifices, and encouragement.

My husband, Aneel, for his constant guidance, support, and patience. Without him this book would be impossible.

Thanks to Salim, Minaz, Rehan, and Yasmeen. There are many things I owe to you.

Thanks to the wonderful team at Packt Publishing for their hard work, dedication, and patience. Special thanks to Steven for being a constant guide throughout the process. Thanks to my reviewers for meticulously evaluating the content and for helping to make the book better.

About the Reviewers

Tim Cromarty has been working in IT for well over 20 years in a variety of positions and responsibilities. In the late '80s he was an Oracle developer, and worked on solutions for a variety of industries including Engineering, TV, Finance, Banking, and Government. Then, in the late '90s he became involved in Java and web development working for the European offices of U.S.-based organizations. For the past 10 years, Tim has held a number of director-level positions in application development, consulting, and operations.

He has been using .NET since it was introduced and is an expert in web development (especially MVC and jQuery), application architecture, solution design, and application development. Being both modest and reliable, as a developer he enjoys learning and promoting new programming languages, techniques, and practices.

Based in Hampshire, England, Tim is currently an application architect working on healthcare solutions, and in his spare time develops websites for local communities and non-profit groups. His blog can be found at `http://blog.clicktricity.com`.

Tim also reviewed *ASP.NET MVC Cookbook* (Packt Publishing).

I am married with an amazing young son. My ambition is to stop lying about my age when I reach 33... My wife thinks I'm a nerd because my idea of relaxing is to read a programming manual! I would like to thank my wife for feeding me during my involvement in this brilliant book.

Buu Nguyen is a Microsoft MVP in ASP.NET and the Vice President of Technology at KMS Technology, a global software services company. Prior to joining KMS Technology, Buu worked as an architect and senior manager in both startup and established software development companies.

A passionate software practitioner, Buu has been involved in developing many types of software ranging from Java and .NET enterprise applications, Ruby on Rails web applications to iOS and Windows Phone 7 mobile applications. He is also an active community contributor who has conducted training for Microsoft, published technical articles, developed open source software, and spoken at technical conferences, including those organized by Microsoft.

In addition to his industrial and community engagement, Buu is a part-time university lecturer who teaches undergraduate courses in .NET web development technologies, Java programming, software architecture, and enterprise system development.

Buu can be contacted through his website at `http://www.buunguyen.net/blog/`.

Aleš Rosina is a freelance developer, devoted to new technologies. Working on mobile and web area specialized in .NET technologies for about 10 years. He is also Microsoft Certified Professional for five years, active in community with writing his own blog (`http://shelastyle.net`), and lecturing on local events and conferences about developing with new technologies. Currently living in Ljubljana, Slovenia, where he works on various projects with different clients.

Hajan Selmani is a known ASP.NET technology expert and Microsoft technologies enthusiast. He works as a Software Engineer in Seavus Group. He has several years of experience with web technologies including client and server side such as ASP.NET, AJAX, jQuery/JavaScript, Web Services, and so on. He holds B.Sc. in Computer Sciences from South East European University and is soon expected to have defense of his Master thesis in the same field. He is an active blogger, speaker and forums contributor in three Microsoft ASP.NET/.NET communities. Currently, he is living and working in Skopje, Macedonia. You can find his blog on `http://weblogs.asp.net/hajan`.

I would like to express my sincere gratitude to my wife Amina and my family for the support they always give me, especially while working on this great book.

---

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

The jQuery library has become increasingly popular with web application developers because of its simplicity and ease of use. The library is supported by an active community of developers and has grown significantly over the years after its inception in 2006 by John Resig. Using this library eases complicated tasks and adds to the interactive experience of the end user. Its extensible plugin architecture enables developers to build additional functionalities on top of the core library.

With Microsoft's contribution of Templates, DataLink, and Globalization plugins to the jQuery library and the distribution of jQuery with Visual Studio 2010 onwards, the library has gained popularity with ASP.NET developers. jQuery can be very easily interfaced with ASP.NET controls as well as custom user controls. It can be used to validate controls using client side scripts, thus giving us an alternative for server side Validation Controls. It can be used to incorporate cool animation effects as well as to create graphic-rich pages. It can be used to post AJAX requests to web services, page methods, and HTTP handlers. The possibilities with jQuery are endless.

This cookbook aims to cover some of the day-to-day tasks faced by ASP.NET developers and how jQuery can be applied to work out the same. We will be focusing on interfacing jQuery with ASP.NET applications and some of the amazing tasks that can be accomplished using this powerful library. The recipes described in this book give fast and easy solutions to some of the common encountered problems in web applications.

What this book covers

Chapter 1, *Working with ASP.NET Controls*, describes the interfacing of standard ASP.NET controls with the jQuery library. Controls such as TextBox, CheckBoxList, DropDownList, and Hyperlink are covered in this chapter.

Chapter 2, *Validation of ASP.NET Controls*, explains different client side validation techniques that can be used with jQuery for validation of ASP.NET controls. These validation techniques provide a substitute for server side Validation Controls.

Chapter 3, *Working with GridView Control*, explores different manipulations that can be applied to the ASP.NET GridView control. You will learn to highlight rows/cells, remove rows/cells, use plugins, and apply animation effects using scripts.

Chapter 4, *Working with Image Control*, describes interesting applications of jQuery with the ASP.NET Image control. The recipes covered demonstrate handy utilities such as zooming, cropping, swapping, and changing the opacity of images. You will also learn to build a photo gallery using jQuery.

Chapter 5, *Animations in ASP.NET*, looks into various interesting animation effects that can be achieved using jQuery. You will learn to apply different types of animation effects to ASP.NET Label, Image, and Panel controls. Various effects such as fading, sliding, enlarging, and so on are covered in this chapter. The chapter also demonstrates animation chaining and prevention of animation queue buildup.

Chapter 6, *AJAX and ASP.NET (Part 1)*, demonstrates the making of AJAX calls using jQuery. The chapter describes three basic techniques of sending AJAX request to the server: via page methods, web services, and HTTP handlers. The chapter also demonstrates the use of the Firebug add-on in Mozilla Firefox browser to investigate the AJAX request/response.

Chapter 7, *AJAX and ASP.NET (Part 2)*, explores more advanced applications of jQuery AJAX. The recipes in this chapter show how to work with HTML, XML, script, JSON, JSONP, and complex data types such as objects using AJAX. You will also learn to build a search engine using Yahoo! Search API.

Chapter 8, *Client Templating in ASP.NET*, explains the use of Microsoft's jQuery Templates plugin in ASP.NET websites. The chapter covers both inline as well as external templates.

What you need for this book

You will need to use the jQuery library and Visual Studio (2005 and above) to work out the examples described in this book. Some of the recipes also require the download of jQuery plugins. The source code included with the book is based in C#.

Who this book is for

This book is for ASP.NET developers wanting to learn the nitty-gritty of interfacing jQuery with ASP.NET applications. The expertise level ranges from basic to advanced.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Create a new ASP.NET website `Chapter1` in Visual Studio 2010."

A block of code is set as follows:

```
<body>
    <form id="form1" runat="server">
       <!—-INCLUDE ASPX MARKUP HERE-- >
    </form>
</body>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
.defaultText
    {
      font-style:italic;
      color:#CCCCCC;
    }
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Add a TextBox field with a **Search** button to the form".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or e-mail `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.PacktPub.com`. If you purchased this book elsewhere, you can visit `http://www.PacktPub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt Publishing, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Working with ASP.NET Controls

In this chapter, we will cover:

- Creating default text for TextBoxes
- Auto focus on the first TextBox and tab on *Enter* key
- Disallowing cut/copy/paste operations on a TextBox
- Highlighting text in a TextBox and copying to the clipboard
- Displaying selected items of a CheckBoxList
- Selecting/deselecting all items in CheckBoxList
- Getting selected text and value from DropDownList
- Appending items at runtime to DropDownList
- Creating 'Back to Top' ASP.NET Hyperlink
- Updating URL of ASP.NET Hyperlink at runtime

Introduction

jQuery is the most widely used JavaScript library in web applications today. Developed by John Resig in 2006, it is supported by an active community of developers apart from the core jQuery team. It is designed to be lightweight, cross browser, and CSS3 compliant. It simplifies the way DOM elements are selected, making it easier to navigate a page. The library has powerful Ajax capabilities, event handling, and animation effects, thus enabling rapid web application development. By allowing the creation of plugins, it enables developers to build powerful capabilities on top of the core jQuery library.

jQuery can be interfaced with numerous web platforms such as ASP, PHP, .NET, and Java. With the official support of Microsoft, it is now distributed with Visual Studio (Version 2010 onwards). Microsoft is also committed to contributing to the library and its jQuery Templates, Data Link, and Globalization plugins are now accepted as official jQuery plugins.

In this chapter, we will be specifically exploring how jQuery can be used in various ways to manipulate ASP.NET controls, the result being a better interactive experience for the web user without causing additional server-side overhead.

Downloading jQuery

jQuery is available from `http://jQuery.com` in different types of builds based on the requirements of the application:

- **Minified version**: This version is preferred for production environments. It has a smaller file size requiring shorter time to download.
- **Packed version**: This is the compressed version of jQuery. Though it has a smaller file size compared to the minified version, it requires non-trivial time to unpack when downloaded by the browser.
- **Uncompressed version**: This version is preferred for development environments. It has a larger file size compared to the minified and packed versions, but is suited for debugging in the development phase.

Including jQuery library

There are two ways of including the jQuery library on the web page:

- Using publicly hosted copies from CDNs (Community Development Networks) such as Google and Microsoft
- Downloading and using a local copy

Both methods are widely used today, the trend being more towards using CDNs. CDNs have performance benefit in terms of improving caching and parallelism. However, in applications deployed on the intranet or applications that have limited or no Internet access, it might be better to host a local copy on the web server or the local file system.

But where exactly does jQuery fit in an ASPX form? To answer that, let's take a look at the basic structure of an ASPX page:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Sample.aspx.
cs" Inherits="Sample" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
```

```
<title>RECIPE TITLE</title>
    <script src="js/jquery-1.4.1.js" type="text/javascript"></script>
    <script type="text/javascript">
        /* INCLUDE JQUERY MARKUP HERE */
    </script>
</head>
<body>
    <form id="form1" runat="server">
       <!—-INCLUDE ASPX MARKUP HERE-- >
    </form>
</body>
</html>
```

From the above sample ASPX form, note the following:

- The jQuery library is included in the page header using <script> tags as follows:

  ```
  <script src="js/jquery-1.4.1.js" type="text/javascript"></script>
  ```

- The actual jQuery magic is applied in another script block in the page header:

  ```
  <script type="text/javascript">
          /* INCLUDE JQUERY MARKUP HERE */
      </script>
  ```

- The web form with the required ASPX markup is included in the page body as follows:

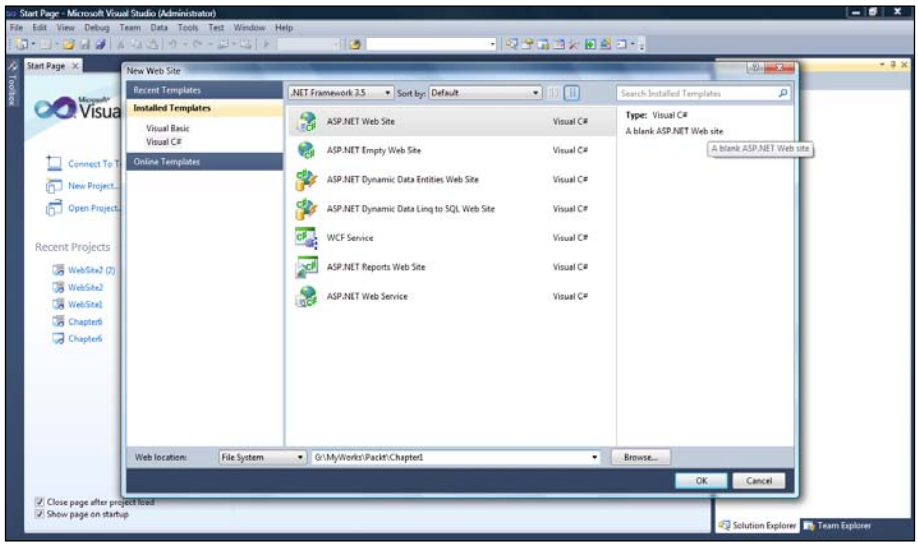
  ```
  <body>
      <form id="form1" runat="server">
         <!—-INCLUDE ASPX MARKUP HERE-- >
      </form>
  </body>
  ```

For the purpose of this book, we have downloaded the uncompressed version of jQuery 1.4.1 on the local system. All recipes and code samples are based on this version of jQuery. The future releases of the library are expected to be compatible backward.

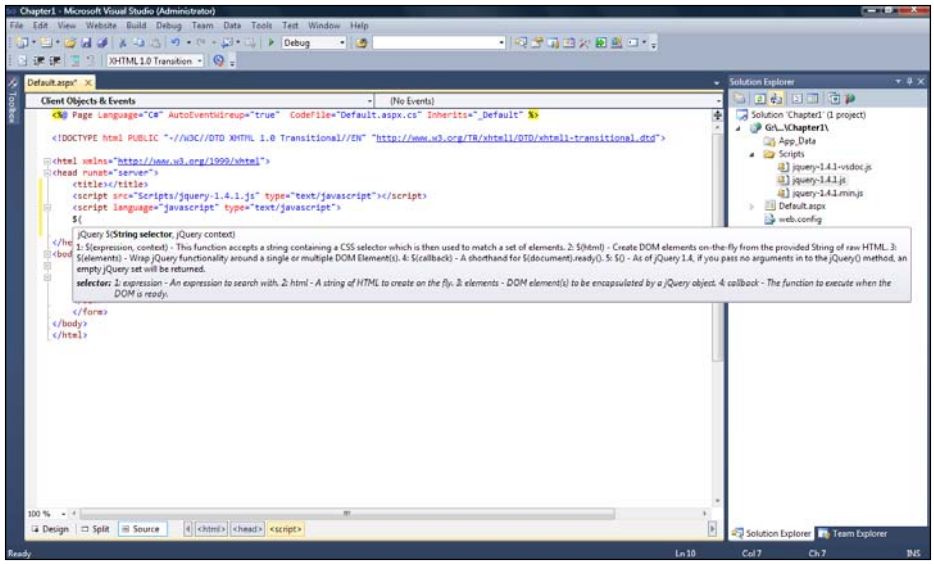
Getting started

Using Visual Studio 2010:

- Create a new ASP.NET website `Chapter1` in Visual Studio 2010 as shown below:

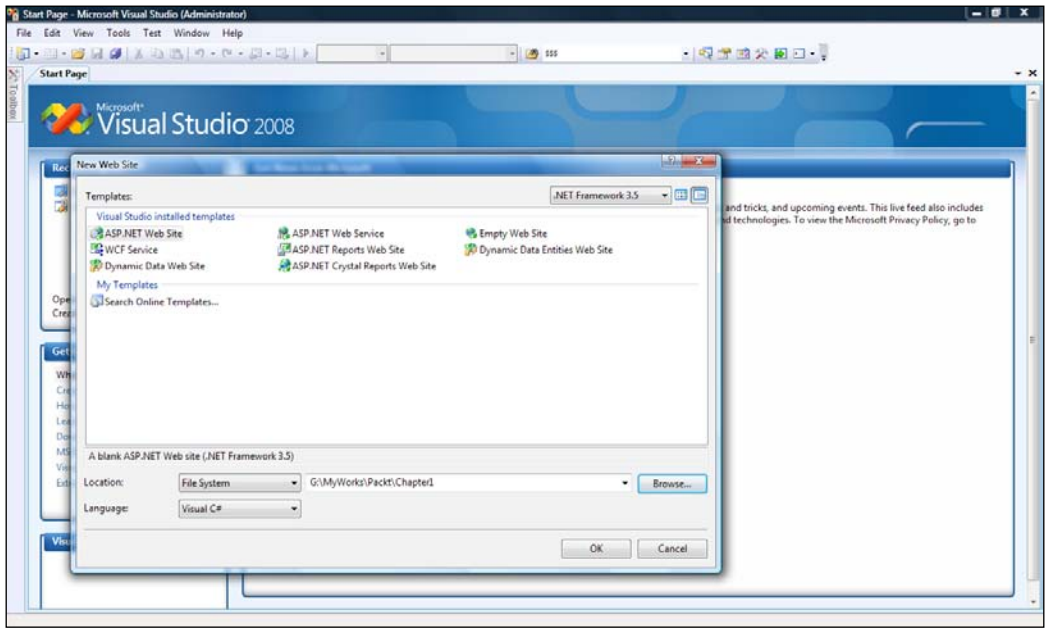


- Since jQuery is distributed with Visual Studio 2010, the library can be directly included and used on ASPX pages as displayed in the screen below:

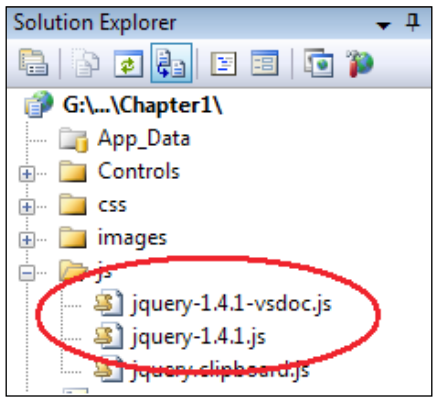


Using Visual Studio 2008

- Create a new ASP.NET website `Chapter1` in Visual Studio 2008 as shown below:



- Download the jQuery library from `http://code.jquery.com/jquery-1.4.1.js` and save to a local folder.
- Download the Visual Studio 2008 intellisense hint file from `http://code.jquery.com/jquery-1.4.1-vsdoc.js`.
- Ensure that the naming prefix of the intellisense hint file matches that of the jQuery library as shown below:



To enable jQuery intellisense in Visual Studio 2008, ensure that you have upgraded VS2008 to Service Pack 1 and applied the hotfix KB958502 `http://archive.msdn.microsoft.com/KB958502`.

Usage of jQuery Library

Create a test form in the project and drag-and-drop the jQuery source file on the form. Visual Studio will automatically add the following code to the ASPX file:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
```

The page is now enabled for using jQuery. We can now include the required jQuery markup in the script block as follows:

```
<script type="text/javascript">
$(document).ready(function() {
…….
…….
});
</ script>
```

All the jQuery markup on the page should be included in the `$(document).ready()` function. This function is executed as soon as the DOM is loaded and before the page contents are rendered.

Now, with this information, let's move on to the recipes where we will solve specific web development problems using jQuery. We have focused on the jQuery scripting required for solving the given problem. The ASP.NET code-behind logic is left to the reader to implement as required.

The recipes in this book have been tested with IE 8, Mozilla Firefox 3.6, and Chrome 9.0. The complete source code for all chapters is included with the book.

When using the provided source code, update the namespace in the Page directive of the ASPX files `<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Recipe1.aspx.cs" Inherits="Recipe1" %>` to your own local namespaces.

Creating default text for TextBoxes

In this recipe, let's see how to create a search box in ASP.NET with some default information text. The text is displayed only when the search box is out of focus.

Getting ready

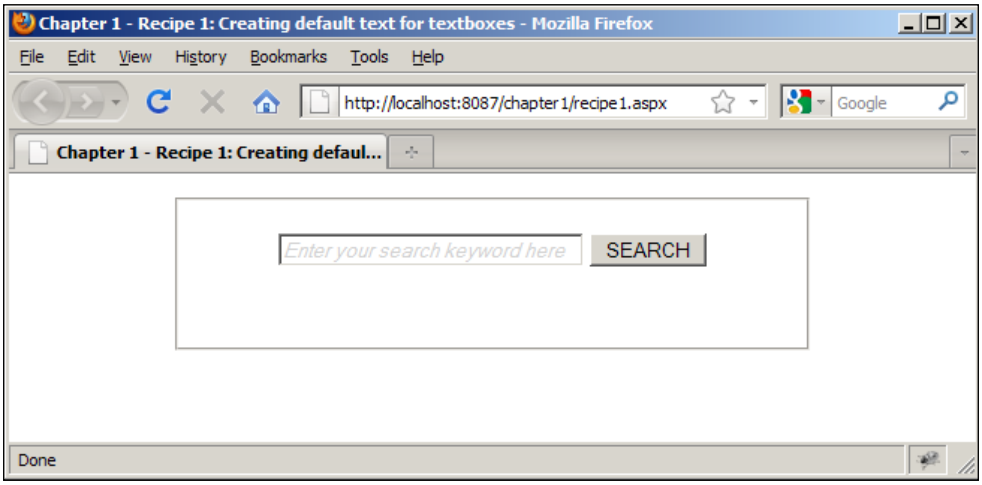
1. Add a new web form `Recipe1.aspx` to the current project.
2. Add a TextBox field with a **Search** button to the form as below:

```
<form id="form1" runat="server">
    <p></p>
    <div align="center">
       <fieldset style="width:400px;height:80px;">
        <p><asp:TextBox ID="TextBox1" width="200px"
CssClass="defaultText" ToolTip="Enter your search keyword here"
runat="server"></asp:TextBox>
        <asp:Button ID="btnSubmit" Text="SEARCH" runat="server"
/></p>
       </fieldset>
    </div>
    </form>
```

3. The CSS class `defaultText` attached to the TextBox is defined as below:

```
.defaultText
    {
      font-style:italic;
      color:#CCCCCC;
    }
```

4. Thus on loading, the page displays the search form as shown in the following screen:



We will now use jQuery to display the default text when the search box is out of focus.

How to do it...

1. In the `document.ready()` function, retrieve the TextBox control using its `ClientID` and save in a local variable:

```
var searchBox = $('#<%=TextBox1.ClientID%>') ;
```

2. On the `focus` event, check if it contains the default text:

```
searchBox.focus( function() {
                if (searchBox.val() == this.title) {
```

The ToolTip property of the ASP.NET TextBox control is rendered as `title` at runtime. Thus, the ToolTip text `Enter your search keyword here` is retrieved using `this.title`.

3. If yes, then remove the `defaultText` css style:

```
                    searchBox.removeClass("defaultText");
```

4. Also clear the search field:

```
                    searchBox.val("");
                }
            });
```

5. On the `blur` event, check if the TextBox is empty:

```
searchBox.blur( function() {
                if (searchBox.val() == "") {
```

6. If yes, then attach the `"defaultText"` css style:

```
                    searchBox.addClass("defaultText");
```

7. Add the default information text to the search field:

```
                    searchBox.val(this.title);
                }
            });
```

8. Call the `blur` event on page load so that the TextBox is initially out of focus:

```
searchBox.blur();
```

Thus, the complete jQuery solution is as follows:

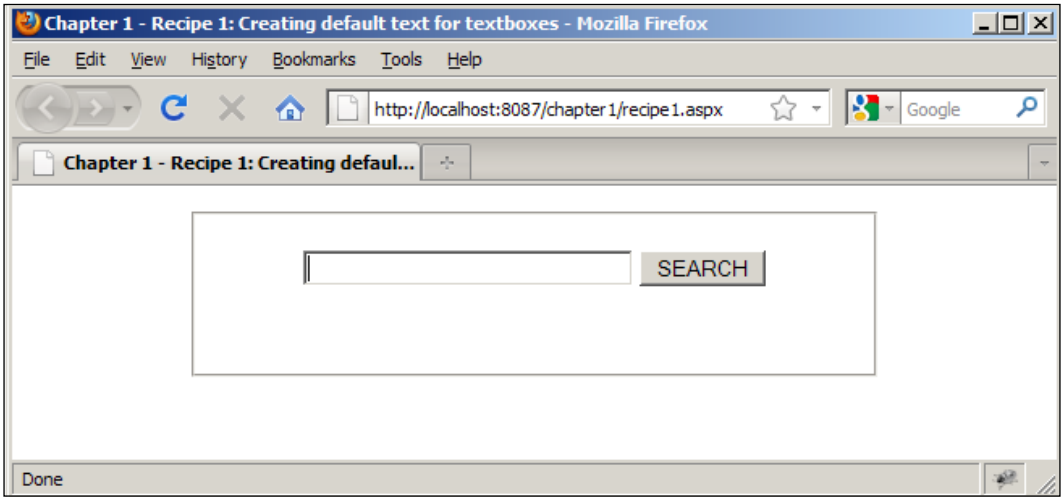
```
<script type="text/javascript">
        $(document).ready(function() {
            var searchBox = $('#<%=TextBox1.ClientID%>');
            searchBox.focus(
            function() {
```

```
            if (searchBox.val() == this.title) {
                searchBox.removeClass("defaultText");
                searchBox.val("");
            }
        });

        searchBox.blur(
        function() {
            if (searchBox.val() == "") {
                searchBox.addClass("defaultText");
                searchBox.val(this.title);
            }
        });
        searchBox.blur();
    });
</script>
```

ClientID is a unique server control identifier generated by ASP.NET to avoid duplicate IDs when controls are repeated; say, for example, in a Repeater control. It is generated by concatenating the ID of the control with the UniqueID of its parent control. Underscore character (_) is used to separate each part of the generated ID.

How it works...

Run the web form. Initially, the page is displayed with the default text. On focusing on the TextBox, the text disappears and the web user can key in the required search keywords. Note that the text style of the default text and the user input are different.



See also

Disallowing cut/copy/paste operations on a TextBox.

Auto focus on the first TextBox and tab on the Enter key

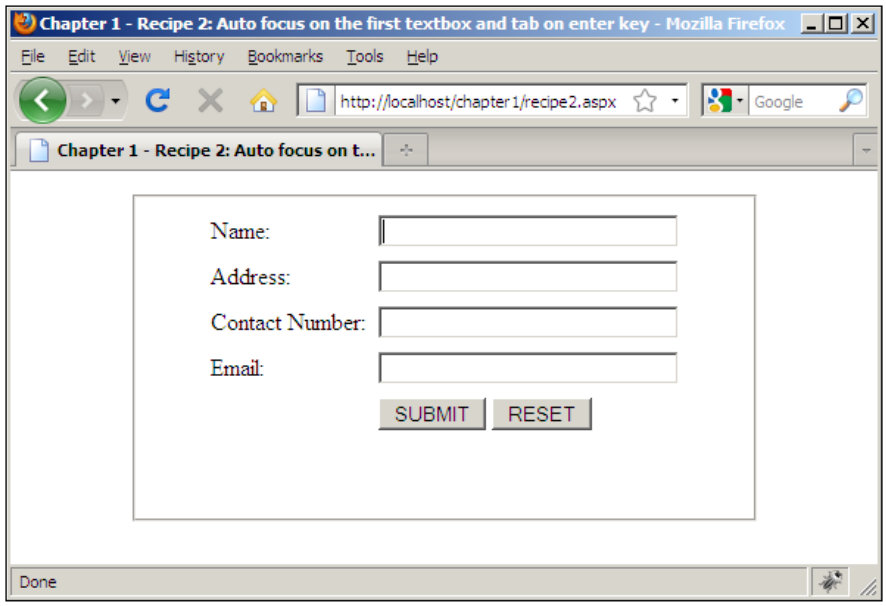
When keying in data in a lengthy form, at times web users prefer to tab to the next box by pressing the `Enter` key. In this recipe, we will see how we can achieve this tabbing effect.

Getting ready

1. Create a new web form `Recipe2.aspx` in the current project.
2. Add a few TextBox controls to create a simple data entry form to capture users' profile information as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:200px;">
    <table cellpadding="3" cellspacing="3" border="0">
    <tr><td>
    <asp:Label ID="lblName" Text="Name: " runat="server"></
asp:Label></td>
    <td><asp:TextBox ID="txtName" Width="200px" runat="server"></
asp:TextBox></td></tr>
    <tr><td><asp:Label ID="lblAddress" Text="Address: "
runat="server"></asp:Label></td>
    <td><asp:TextBox ID="txtAddress" Width="200px"
runat="server"></asp:TextBox></td></tr>
    <tr><td><asp:Label ID="lblContact" Text="Contact Number: "
runat="server"></asp:Label></td>
    <td><asp:TextBox ID="txtContact" Width="200px"
runat="server"></asp:TextBox></td></tr>
    <tr><td><asp:Label ID="lblEmail" Text="Email: "
runat="server"></asp:Label></td>
    <td><asp:TextBox ID="txtEmail" Width="200px" runat="server"></
asp:TextBox></td></tr>
    <tr><td></td><td><asp:Button ID="btnSubmit" Text="SUBMIT"
runat="server" />
    <asp:Button ID="btnReset" Text="RESET" runat="server" />
    </td></tr>
    </table>
    </fieldset>
    </div>
    </form>
```

3. The page displays the form as shown in the following screen:



How to do it...

1. In the `document.ready()` function in the jQuery script block, use `$('input:text:first')` selector to retrieve a handle to the first ASP.NET TextBox. Call the `focus` event on this TextBox:

```
$('input:text:first').focus();
```

The ASP.NET engine renders a TextBox control as `<input type=text>` HTML element at runtime. Hence, the selector `$('input:text')` can be used to retrieve TextBox controls. The selector `$('input:text:first')` retrieves the first TextBox on the form.

2. Use `bind` to attach an event handler for `"keydown"` event on all the TextBoxes in the form:

```
$('input:text').bind("keydown", function(e) {
```

3. Check if the *Enter* key is pressed:

```
if (e.which == 13) {
```

4. If the *Enter* key is pressed, prevent default form submission:

```
e.preventDefault();
```

5. Increment the index of the current TextBox by 1 to retrieve the handle to the immediately next TextBox on the form:

```
var nextIndex = $('input:text').index(this) + 1;
```

6. Focus on the next TextBox:

```
$('input:text')[nextIndex].focus();
}
});
```

7. Add reset functionality to the `Reset` button by adding a reset function to the form object on its `click` event:

```
$('#btnReset').click(
                function() {
                    $('form')[0].reset();
                });
```

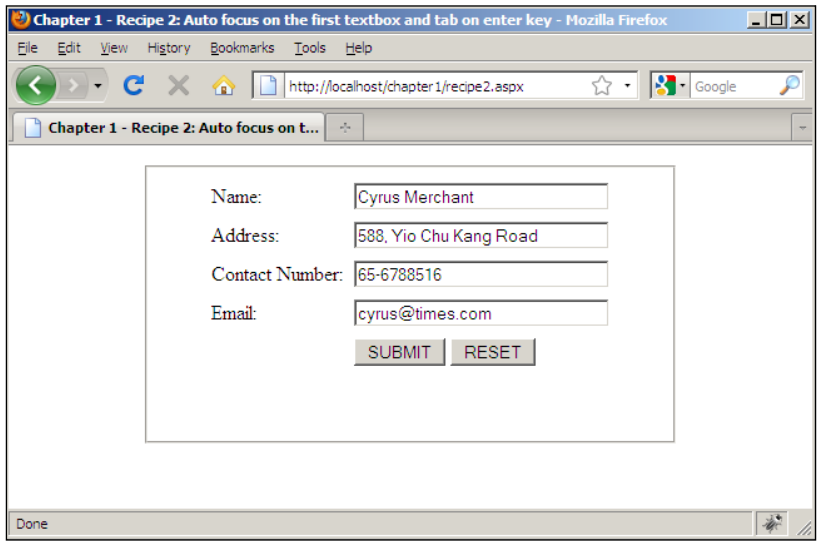
The selector `$('form')[0]` retrieves the first form object on the page.

Thus, the complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
            $('input:text:first').focus();
            $('input:text').bind("keydown", function(e) {
                if (e.which == 13) {   //Enter key
                    e.preventDefault(); //to skip default behavior of
the enter key
                    var nextIndex = $('input:text').index(this) + 1;
                    $('input:text')[nextIndex].focus();
                }
            });

            $('#btnReset').click(
                 function() {
                     $('form')[0].reset();
                 });

        });
    </script>
```

How it works...

Run the web form. On loading, the cursor auto focuses on the first TextBox. Subsequently, on pressing the *Enter* key, we can tab to the remaining TextBoxes.



See also

Selecting text in a TextBox and copying to the clipboard.

Disallowing cut/copy/paste operations on a TextBox

In this recipe, let's see how to disable cut/copy/paste operations on an ASP.NET Textbox. When requesting sensitive information from the end user such as passwords, credit card details, and transaction pin numbers used in Internet banking, etc. it is preferred that the web user keys in the characters manually rather than pasting from the clipboard.

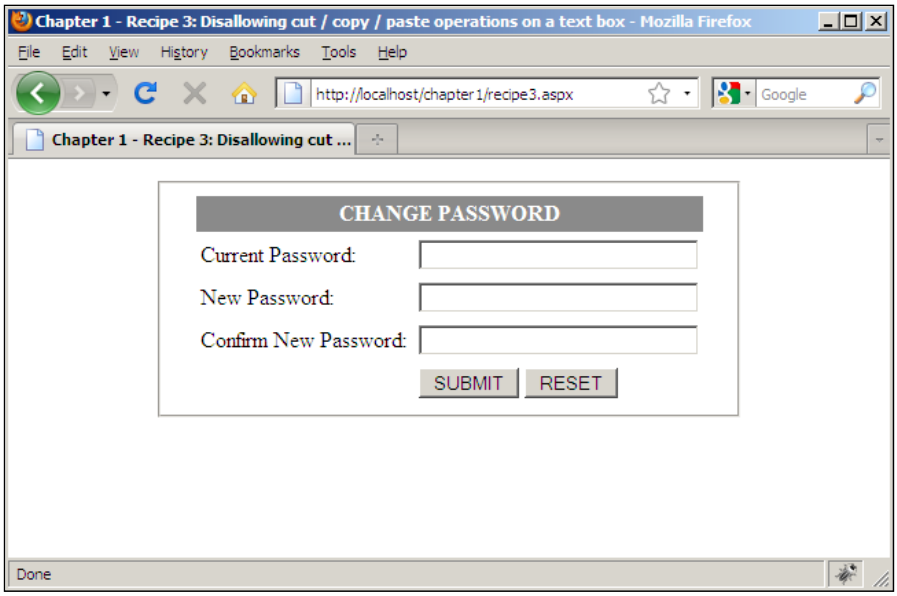
Getting ready

1. Add a new web form `Recipe3.aspx` to the current project.
2. Create a simple password change form:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:180px;">
```

```
    <table cellpadding="3" cellspacing="3" border="0">
    <tr><td colspan="2" class="header">CHANGE PASSWORD</td></tr>
    <tr><td> <asp:Label id="lblCurrentPwd" Text="Current Password:
" runat="server"/></td>
    <td> <asp:TextBox ID="txtCurrentPwd" Width="200px"
runat="server" TextMode="Password"></asp:TextBox></td></tr>
    <tr><td><asp:Label id="lblNewPwd" Text="New Password:"
runat="server"/></td>
    <td><asp:TextBox ID="txtNewPwd" Width="200px" runat="server"
TextMode="Password"></asp:TextBox></td></tr>
    <tr><td><label id="lblConfirmNewPwd" runat="server">Confirm
New Password:</label></td>
    <td><asp:TextBox ID="txtConfirmNewPwd" Width="200px"
runat="server" TextMode="Password"></asp:TextBox></td></tr>
    <tr><td></td><td><asp:Button ID="btnSubmit" runat="server"
Text="SUBMIT" />
    <asp:Button ID="btnReset" runat="server" Text="RESET" /></
td></tr>
    </table>
    </fieldset>
    </div>
    </form>
```

3. The page displays the form as shown in the screen below:



How to do it...

1. In the `document.ready()` function in the jQuery script block, use `bind` to attach the required event handler for `cut`, `copy`, and `paste` events for the `New Password` TextBox:

```
$('#<%=txtNewPwd.ClientID%>').bind('cut copy paste', function(e) {
```

2. Override the default cut/copy/paste behavior:

```
e.preventDefault();
```

3. Display an information message:

```
alert("Cut / Copy / Paste disabled in this textbox");
});
```

4. Repeat the above for the `Confirm New Password` TextBox:

```
$('#<%=txtConfirmNewPwd.ClientID%>').bind('cut copy paste',
function(e) {
                e.preventDefault();
                alert("Cut / Copy / Paste disabled in this textbox
too!");
            });
```

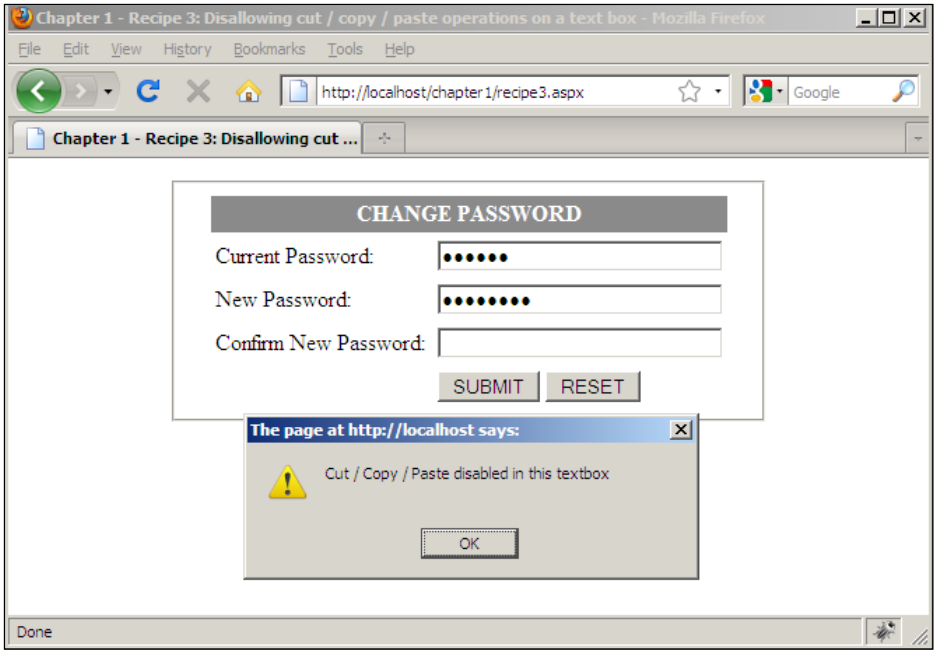
Some browsers by default do not allow `copy` and `cut` events in a `password` field. Hence, in such cases, the jQuery function will not be invoked when these events are raised.

Thus, the complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
            $('#<%=txtNewPwd.ClientID%>').bind('cut copy paste',
function(e) {
                e.preventDefault();
                alert("Cut / Copy / Paste disabled in this textbox");
            });
            $('#<%=txtConfirmNewPwd.ClientID%>').bind('cut copy
paste', function(e) {
                e.preventDefault();
                alert("Cut / Copy / Paste disabled in this textbox
too!");
            });
        });
    </script>
```

How it works...

Run the web page. Try cut/copy/paste operations on the `New Password` and `Confirm New Password` fields. The page will throw the information message as shown in the screen below:



See also

Creating default text for TextBoxes

Highlighting text in a TextBox and copying to the clipboard

In this recipe, we will see how to highlight text in a multiline TextBox control and copy to the clipboard.

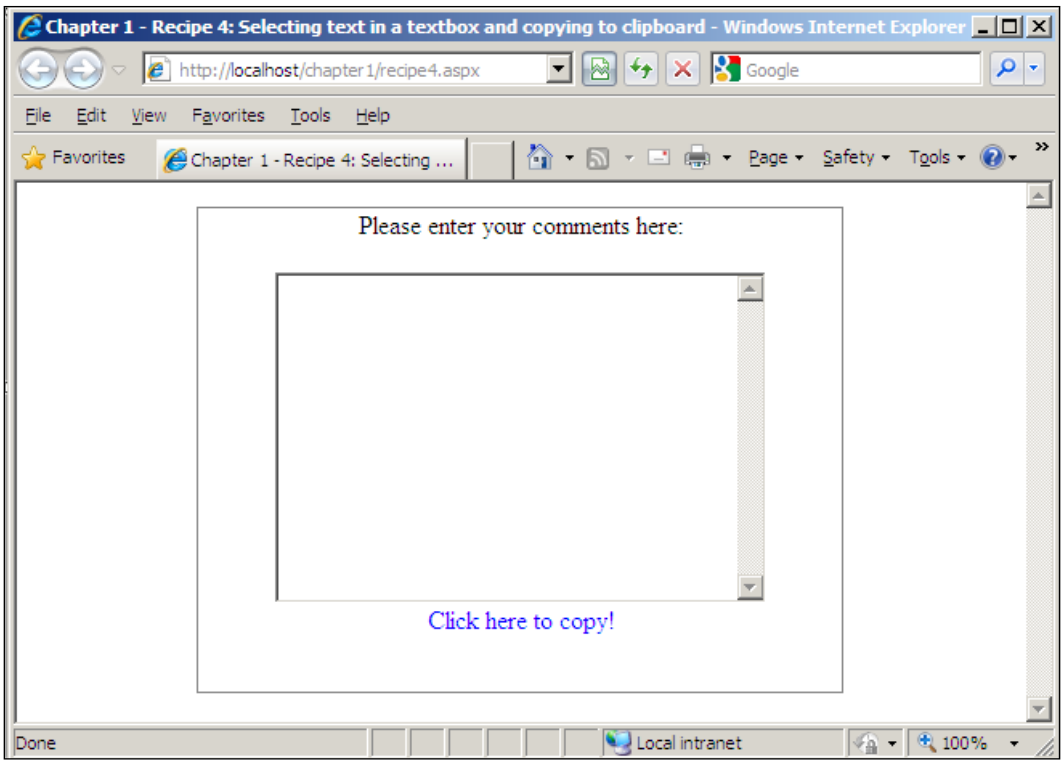
Due to the limitation of the jQuery clipboard plugin, this recipe is applicable to IE browsers only.

Getting ready

1. Create a new web form `Recipe4.aspx` to the current project.
2. Add a multiline TextBox with a hyperlink control to the form as shown below:

```
<form id="form1" runat="server">
    <div align="center">
       <fieldset style="width:400px;height:300px;">
       <p>Please enter your comments here:</p>
        <asp:TextBox ID="TextBox1" TextMode="MultiLine" Rows="5"
Width="300" Height="200" runat="server"></asp:TextBox>
       <br />
       <asp:HyperLink ID="lnkHighlight" Text="Click here to copy!"
runat="server"></asp:HyperLink>
    </fieldset>
    </div>
    </form>
```

3. The page displays the form as shown in the following screen:



4. Add css style for highlighted text using `::selection` CSS3 selector:

```
::selection
    {
      background:#0000FF;
    }
```

5. Add the following css style for the hyperlink:

```
a
    {
      cursor:hand;
      color:#0000FF;
    }
```

6. Download the jQuery clipboard plugin from `http://plugins.jquery.com/project/clipboard`.
7. Include the clipboard plugin on the page along with the jQuery library:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
<script src="js/jquery.clipboard.js" type="text/javascript"></
script>
```

How to do it...

1. In the `document.ready()` function, use the `clipboardReady()` function of the plugin:

```
$.clipboardReady(function() {
```

2. Trigger the `copy to clipboard` action on the click event of the hyperlink:

```
$("a").click(function() {
```

3. Highlight the text in the TextBox Control:

```
$('#<%=TextBox1.ClientID%>').select();
```

4. Copy the contents of the TextBox into the clipboard using the clipboard plugin:

```
$.clipboard($('#<%=TextBox1.ClientID%>').val());
});
        });
```

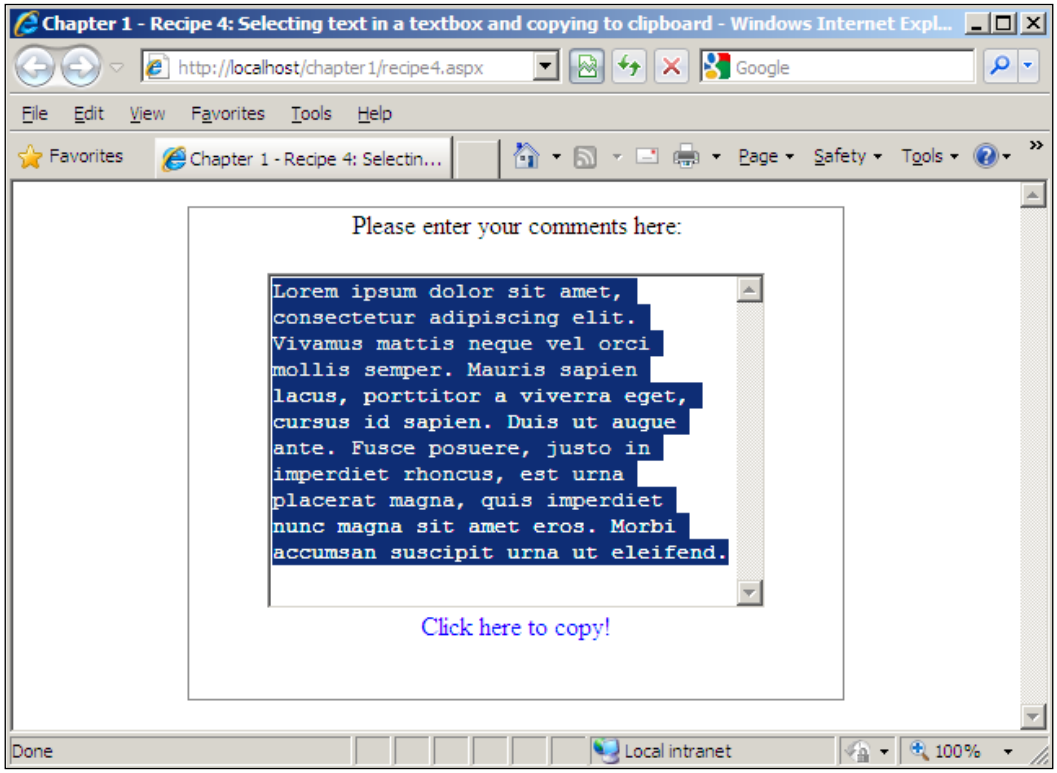
The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
        $.clipboardReady(function() {
            $("a").click(function() {
                $('#<%=TextBox1.ClientID%>').select();
```

```
                $.clipboard($('#<%=TextBox1.ClientID%>').val());
                });
        });

        });
    </script>
```

How it works...

Run the web page. Enter some text in the multiline TextBox field and click the hyperlink. The text will be highlighted as shown below:



Launch any text editor and press *Ctrl+V* to paste the text copied from the TextBox.

There's more...

To implement this functionality in non-IE browsers, an alternative to the clipboard plugin such as **Zero Clip Board** can be used. Zero Clip Board actually uses a flash file and a JavaScript interface. More information on Zero Clip Board can be found at `http://code.google.com/p/zeroclipboard/`.

See also

Auto focus on the first TextBox and tab on the *Enter* key.

Displaying selected items of a CheckBoxList

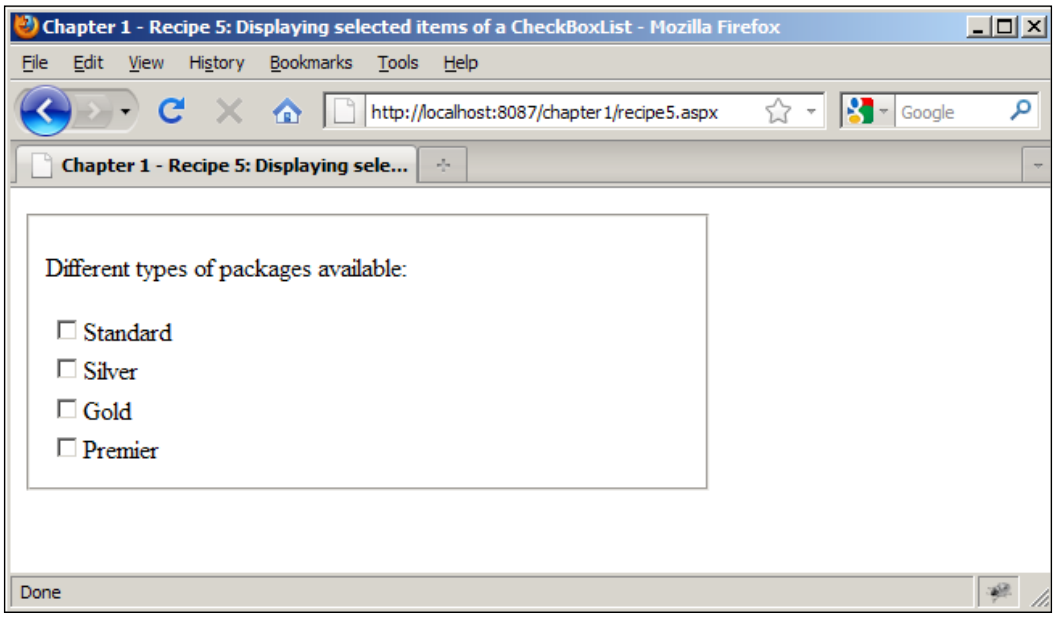
In this recipe, we will retrieve the selected items of an ASP.NET CheckBoxList.

Getting ready

1. Create a new form `Recipe5.aspx` in the current project. Add a CheckBoxList control to the form. Also add a div area to display the selected items of the CheckBoxList at runtime:

```
<form id="form1" runat="server">
     <div align="left">
        <fieldset style="width:400px;height:150px;">
       <p>Different types of packages available:</p>
        <asp:CheckBoxList ID="CheckBoxList1" runat="server">
        <asp:ListItem Text="Standard" Value="1"></asp:ListItem>
        <asp:ListItem Text="Silver" Value="2"></asp:ListItem>
        <asp:ListItem Text="Gold" Value="3"></asp:ListItem>
        <asp:ListItem Text="Premier" Value="4"></asp:ListItem>
        </asp:CheckBoxList>
        </fieldset>
        <br />
        <div id="message"></div>
    </div>
    </form>
```

2. The page will display the CheckBoxList as shown below:



How to do it...

1. In the `document.ready()` function, define the click function of the CheckBoxList:

```
$('#<%=CheckBoxList1.ClientID%>').click(function() {
```

2. Create a local string variable and initialize to an empty string:

```
var str = "";
```

3. Apply a jQuery selector to retrieve the checked boxes and apply the `each` function to each element returned:

```
$('#<%=CheckBoxList1.ClientID%> input[type=checkbox]:checked').
each(function() {
```

Note that at runtime, the CheckBoxList control is rendered as a container table `<table id="CheckBoxList1">` and each of the ListItems are rendered as `<input type="checkbox">`.

4. Retrieve the text of the selected CheckBox and append to the local string:

```
str = str + " " + $(this).next().text();
```

Each of the checkbox elements is rendered as a `<input type="checkbox">` and `<label>` pair. The label holds the text of the element. Hence, when retrieving the description of the element, we have used `$(this).next().text()`, where `$(this)` refers to the current checkbox element and `next()` refers to the consecutive `<label>` element.

5. After the execution of the `each()` function, display the local string in the div area `'message'`:

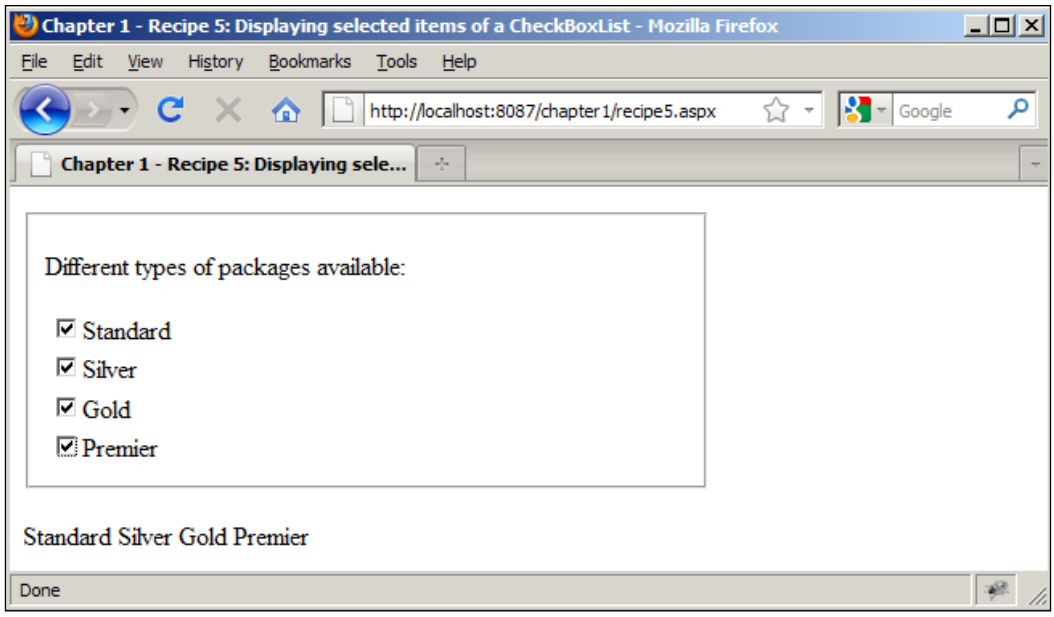
```
$('#message').text(str);
});
```

The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
            $('#<%=CheckBoxList1.ClientID%>').click(
                function() {
                    var str = "";
                    $('#<%=CheckBoxList1.ClientID%>
input[type=checkbox]:checked').each(function() {
                        str = str + " " + $(this).next().text();
                    });
                    $('#message').text(str);
                });
        });
    </script>
```

How it works...

Run the page. Select the required checkboxes. The page will display the text of the selected checkboxes as follows:



See also

Selecting/deselecting all items in ASP.NET CheckBoxList

Selecting/deselecting all items in CheckBoxList

In this recipe, we will see how to select/deselect all items in an ASP.NET CheckBoxList using a **Select All** CheckBox Control.

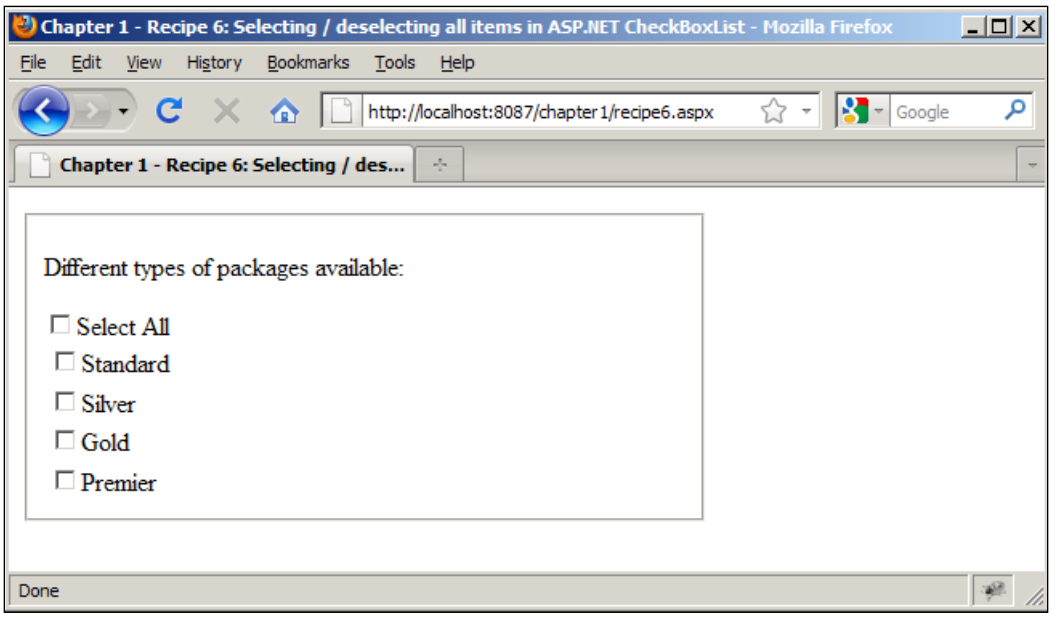
Getting ready

1. Create a new web form `Recipe6.aspx` in the current project.
2. Add a CheckBoxList to the form as well as a **Select All** CheckBox control on the form:

```
<form id="form1" runat="server">
    <div align="left">
       <fieldset style="width:400px;height:170px;">
       <p>Different types of packages available:</p>
    <asp:CheckBox ID="chkSelectAll" runat="server" Text="Select
All" />
    <asp:CheckBoxList ID="chkList" runat="server">
        <asp:ListItem Text="Standard" Value="1"></asp:ListItem>
```

```
            <asp:ListItem Text="Silver" Value="2"></asp:ListItem>
            <asp:ListItem Text="Gold" Value="3"></asp:ListItem>
            <asp:ListItem Text="Premier" Value="4"></asp:ListItem>
        </asp:CheckBoxList>
        </fieldset>
        </div>
        </form>
```

3. The page will display the form as shown in the screen below:



How to do it...

1. In the `document.ready()` function in the jQuery script block, add the required handler for the click event of the **Select All** CheckBox:

   ```
   $('#chkSelectAll').click(function() {
   ```

2. Use the jQuery selector to retrieve all checkboxes from the CheckBoxList and toggle the `'checked'` attribute:

   ```
   $("#<%=chkList.ClientID%> input[type='checkbox']").attr('checked',
   $('#chkSelectAll').is(':checked'));
   }
   ```

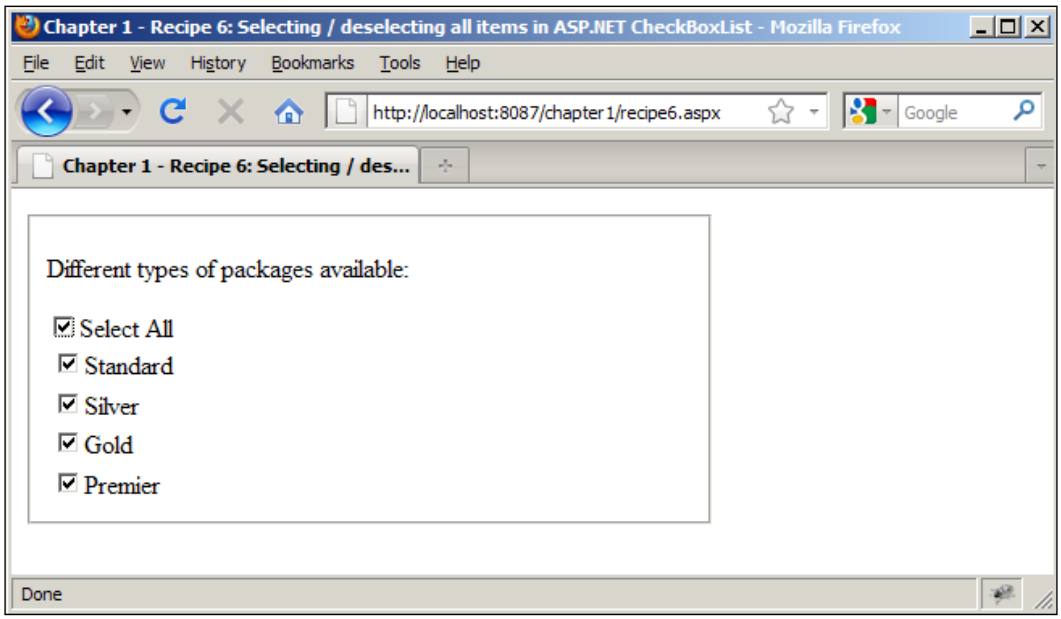
The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
```

```
            $('#chkSelectAll').click(function() {
            $("#<%=chkList.ClientID%> input[type='checkbox']").
            attr('checked', $('#chkSelectAll').is(':checked'));
           }
    );
       });
</script>
```

How it works...

Run the web page. Now select/deselect the **Select All** CheckBox. You will notice that the elements in the CheckBoxList are checked/unchecked based on the CheckBox control.



See also

Displaying selected items of a CheckBoxList

Getting selected text and value from DropDownList

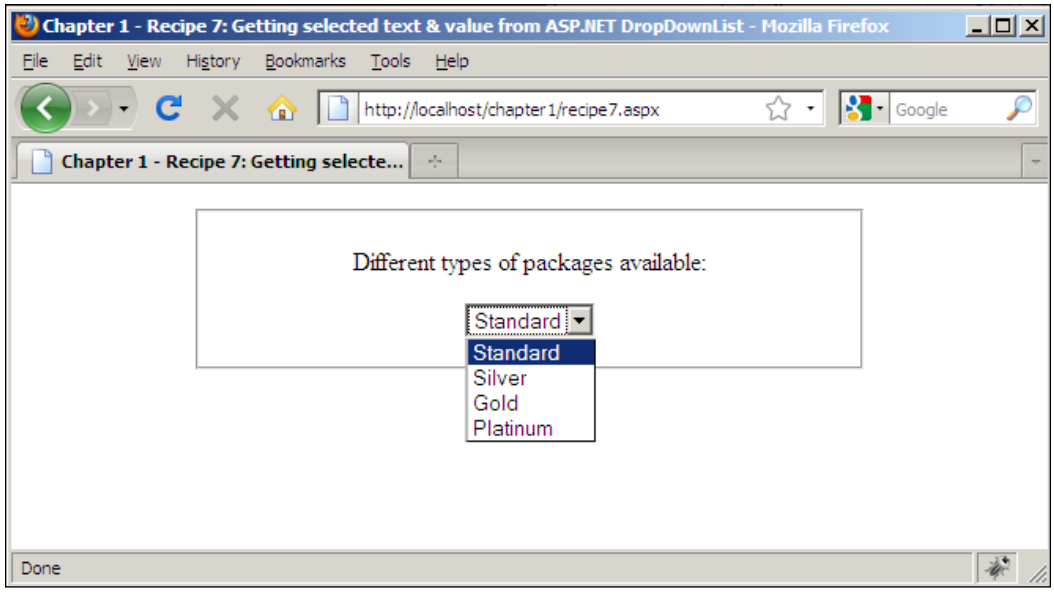
In this recipe, we will retrieve the Text/Value pair of an ASP.NET DropDownList using jQuery.

Getting ready

1. Create a new web form `Recipe7.aspx` in the current project.
2. Add a DropDownList to the form. Include a div area to the form to display the selected text from the DropDownList:

```
<form id="form1" runat="server">
<div align="center">
   <fieldset style="width:400px;height:80px;">
   <p>Different types of packages available:</p>
<asp:DropDownList ID="DropDownList1" runat="server">
<asp:ListItem Text="--Please Select--" Value=""></
asp:ListItem>
<asp:ListItem Text="Standard" Value="1"></asp:ListItem>
<asp:ListItem Text="Silver" Value="2"></asp:ListItem>
<asp:ListItem Text="Gold" Value="3"></asp:ListItem>
<asp:ListItem Text="Platinum" Value="4"></asp:ListItem>
</asp:DropDownList>
</fieldset>
</div>
<br />
<div align="center" id="message"></div>
</form>
```

3. The page displays the DropDownList as shown below:



How to do it...

1. In the `document.ready()` function in the jQuery script block, use `bind()` to attach a handler to the `keyup` and `change` events:

   ```
   $('select[id$=<%=DropDownList1.ClientID%>]').bind("keyup change",
   function() {
   ```

2. Check if the selected value of the DropDownList is empty:

   ```
   if ($(this).val() != "")
   ```

3. Retrieve the Text/Value pair and display in the div area `'message'`:

   ```
   $('#message').text("Text: " + $(this).find(":selected").text()
                       + ' Value: ' + $(this).val());
   ```

4. If, however, the DropDownList is empty, clear the div area:

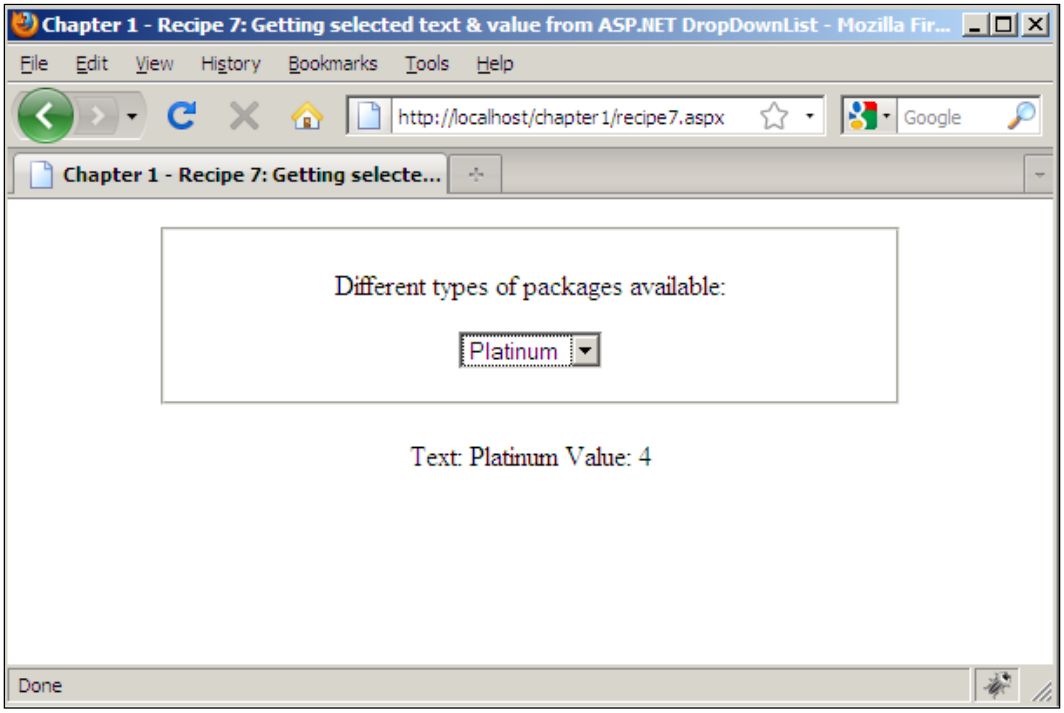
   ```
   else

   $('#message').text("");
   });
   ```

The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
        $('select[id$=<%=DropDownList1.ClientID%>]').bind("keyup
        change", function() {
            if ($(this).val() != "")
                $('#message').text("Text: " + $(this).
                find(":selected").text()
                    + ' Value: ' + $(this).val());
            else
                $('#message').text("");
            });
        });
    </script>
```

How it works...

Run the web form. Select any item from the DropDownList. The Text/Value pair will display below the control as follows:



See also

Appending items at runtime to an ASP.NET DropDownList.

Appending items at runtime to a DropDownList

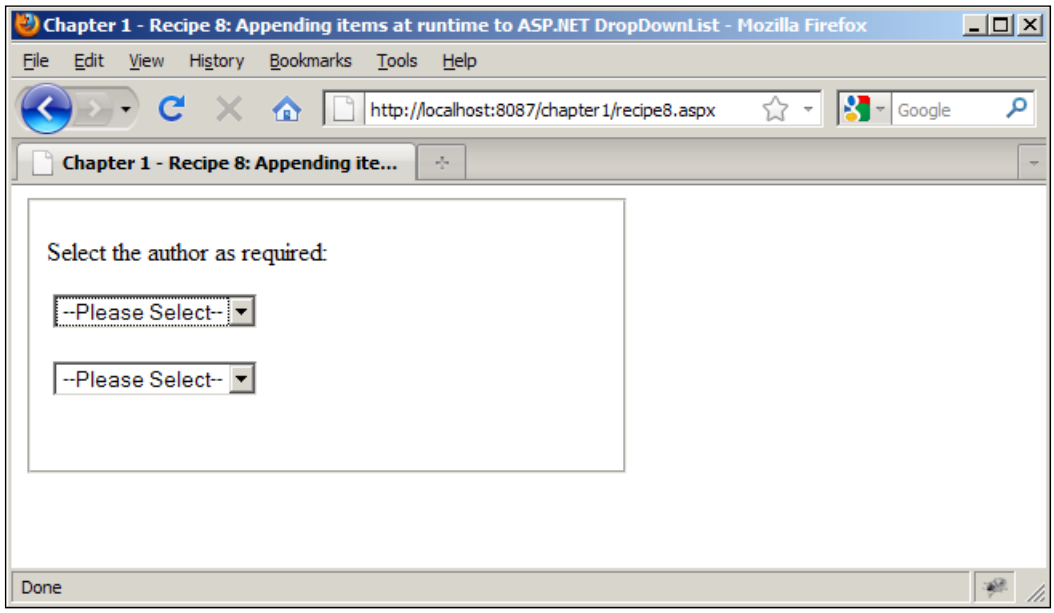
In web applications, there is often a need to populate a DropDownList dynamically based on the events triggered on other controls on the page. In this recipe, we will see how to create a cascading dropdown where the contents of the second DropDownList are populated based on the selected elements of the first DropDownList.

Getting ready

1. Create a new web form `Recipe8.aspx` in the current project.
2. Drag-and-drop two DropDownList from the ToolBox on the form. Populate the first DropDownList as shown below:

```
    <form id="form1" runat="server">
    <div align="left">
       <fieldset style="width:350px;height:150px;">
       <div id="contentArea">
       <p>Select the author as required:</p>
       <table cellpadding="0" cellspacing="0" border="0">
       <tr><td> </td>
       <td>
        <asp:DropDownList ID="DropDownList1" runat="server">
        <asp:ListItem Value="" Text="--Please Select--"></
asp:ListItem>
        <asp:ListItem Value="MA" Text="Mitch Albom"></
asp:ListItem>
        <asp:ListItem Value="PC" Text="Paulo Coelho"></
asp:ListItem>
        <asp:ListItem Value="JA" Text="Jane Austen"></
asp:ListItem>
        </asp:DropDownList>
        </td></tr>
        <tr><td colspan="2"> </td></tr>
        <tr><td> </td>
        <td>
        <asp:DropDownList ID="DropDownList2" runat="server">
        <asp:ListItem Value="" Text="--Please Select--"></
asp:ListItem>
        </asp:DropDownList>
        </td></tr>
        </table>
        </div>
    </fieldset>
    </div>
    </form>
```

3. The page displays the controls as shown in the screen below:



How to do it...

1. In the `document.ready()` function in the jQuery script block, attach the handler for the `"change"` event in the first DropDownList using `bind()`:

```
$("select[id$=DropDownList1]").bind("change", function() {
```

- Clear the second DropDownList:

```
$("select[id$=DropDownList2] option").remove();
```

- Append a default item in the second DropDownList:

```
$("select[id$=DropDownList2]").append("<option value=''>--Please
Select--</option>");
```

- Check the selected value of the first DropDownList and populate the required data in the second DropDownList:

```
if ($(this).val() == "MA") {

$("select[id$=DropDownList2]").append("<option value='MA1'>Have a
Little Faith</option>");
                    $("select[id$=DropDownList2]").append("<option
value='MA2'>Tuesdays with Morrie</option>");
                    $("select[id$=DropDownList2]").append("<option
value='MA3'>The Five People You Meet in Heaven</option>");
```

```
                                    }
                    if ($(this).val() == "PC") {

    $("select[id$=DropDownList2]").append("<option value='PC1'>The
    Alchemist</option>");
                        $("select[id$=DropDownList2]").append("<option
    value='PC2'>By the River Piedra I Sat Down and Wept</option>");
                        $("select[id$=DropDownList2]").append("<option
    value='PC3'>The Pilgrimage</option>");
                    }
                    if ($(this).val() == "JA") {

    $("select[id$=DropDownList2]").append("<option value='JA1'>Pride
    and Prejudice</option>");
                        $("select[id$=DropDownList2]").append("<option
    value='JA2'>Emma</option>");
                        $("select[id$=DropDownList2]").append("<option
    value='JA3'>Mansfield Park</option>");
                    }
    });
```

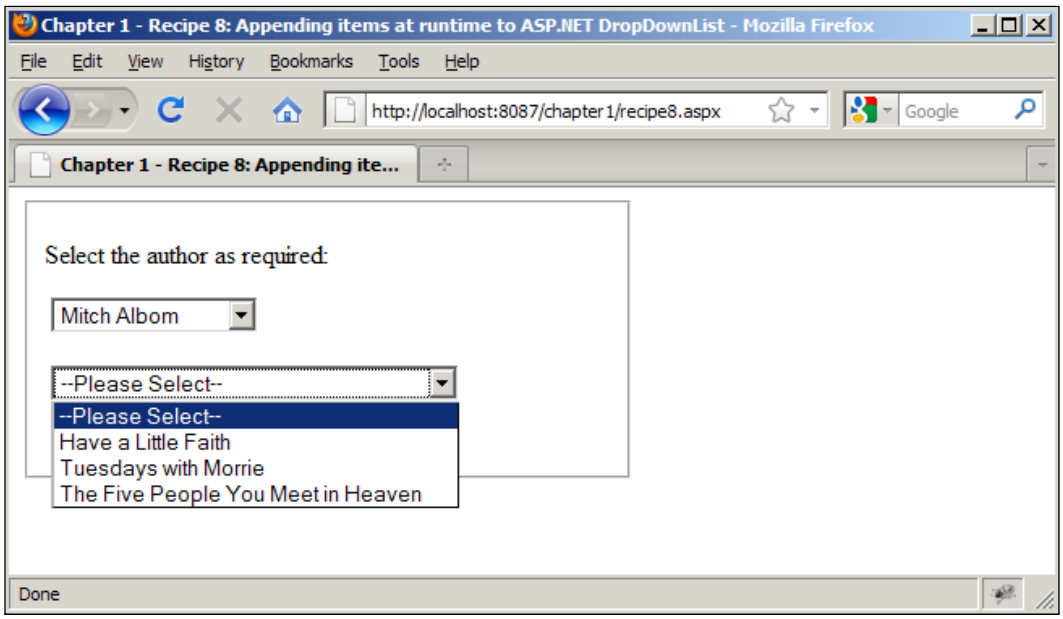
The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
        $("select[id$=DropDownList1]").bind("change", function() {
        $("select[id$=DropDownList2] option").remove();
        $("select[id$=DropDownList2]").append("<option value=''>--
Please Select--</option>");
                if ($(this).val() == "MA") {

                    $("select[id$=DropDownList2]").append("<option
value='MA1'>Have a Little Faith</option>");
                    $("select[id$=DropDownList2]").append("<option
value='MA2'>Tuesdays with Morrie</option>");
                    $("select[id$=DropDownList2]").append("<option
value='MA3'>The Five People You Meet in Heaven</option>");
                                    }
                if ($(this).val() == "PC") {

                    $("select[id$=DropDownList2]").append("<option
value='PC1'>The Alchemist</option>");
                    $("select[id$=DropDownList2]").append("<option
value='PC2'>By the River Piedra I Sat Down and Wept</option>");
                    $("select[id$=DropDownList2]").append("<option
value='PC3'>The Pilgrimage</option>");
```

```
                }
                if ($(this).val() == "JA") {

                    $("select[id$=DropDownList2]").append("<option
value='JA1'>Pride and Prejudice</option>");
                    $("select[id$=DropDownList2]").append("<option
value='JA2'>Emma</option>");
                    $("select[id$=DropDownList2]").append("<option
value='JA3'>Mansfield Park</option>");
                }
            });
        });
    </script>
```

How it works...

Run the web page. Select any item from the first DropDownList. On `"change"` event, the jQuery script checks for the value of the selected element and accordingly populates the second DropDownList as shown below:



There's more...

An alternative way of appending items to the DropDownList is as follows:

```
$("select[id$=DropDownList2]").append($("<option></option>").
val("MA3").html("The Five People You Meet in Heaven"));
```

See also

Getting selected text and value from ASP.NET DropDownList

Creating 'Back to Top' ASP.NET hyperlink

In this recipe, we will create 'Back to Top' ASP.NET links to scroll up to the top of the page. These links are useful when the user needs to scroll the page in case the content is more than one screenful.

Getting ready

1. Create a new web form `Recipe9.aspx` in the current project. Add a few div areas with some text content in the form. Also add hyperlinks below each div area as below:

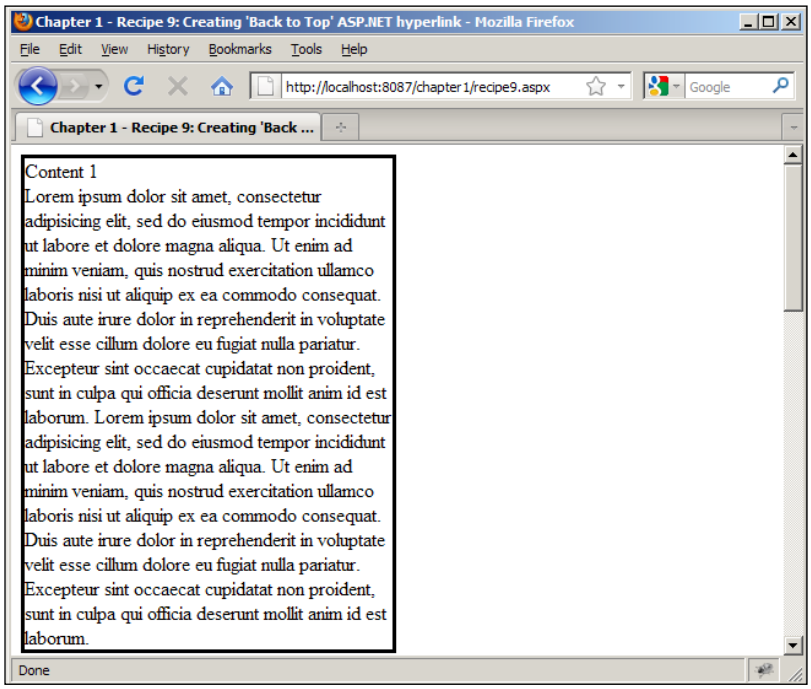
```
<form id="form1" runat="server">
    <div>
        <asp:HyperLink ID="Top" runat="server"></asp:HyperLink>
        <div style="width:300px;border:solid;">
        Content 1
        <br />
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        </div>
        <asp:HyperLink ID="HyperLink1" runat="server"
CssClass="backtotoplink"></asp:HyperLink>
        <div style="width:300px;border:solid;">
        Content 2
```

```
        <br />
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        </div>
        <asp:HyperLink ID="HyperLink2" runat="server"
CssClass="backtotoplink"></asp:HyperLink>
        <div style="width:300px;border:solid;">
        Content 3
        <br />
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
        </div>
        <asp:HyperLink ID="HyperLink3" runat="server"
CssClass="backtotoplink"></asp:HyperLink>
    </div>
    </form>
```

2. Define the css style for the hyperlinks as below:

```
.backtotoplink
    {
     color:#00FF00;
     cursor:hand;
    }
```

3. The page displays the content as shown below:



How to do it...

1. In the `document.ready()` function in the jQuery script block, add the `innerHTML` attribute to the hyperlink:

```
$(".backtotoplink").attr("innerHTML", "Back to Top");
```

2. Add the `title` attribute:

```
$(".backtotoplink").attr("title", "Back to Top");
```

3. Add the `href` attribute:

```
$(".backtotoplink").attr("href", "#Top");
```

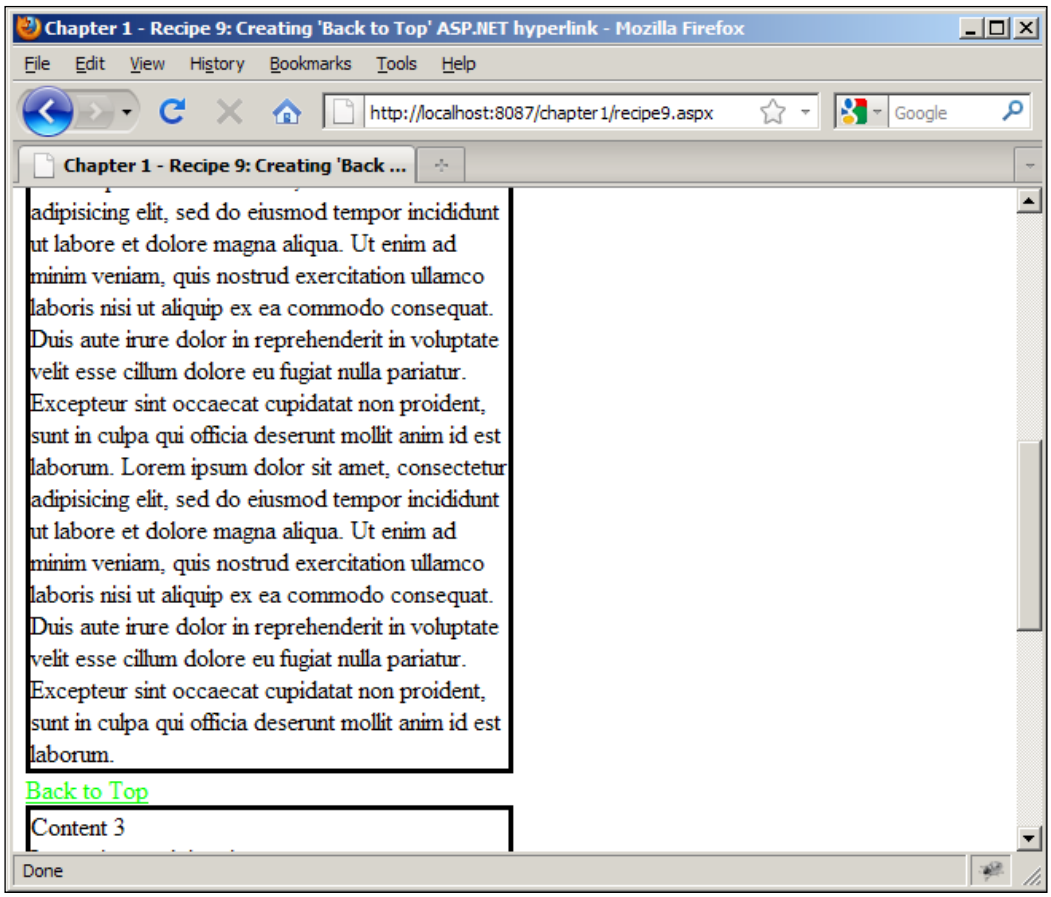
ASP.NET Hyperlinks are rendered as <a> tags on the web page. By manipulating the `href` attribute of anchor tags via jQuery, we can set the ASP.NET hyperlink to point to the top of the page.

The complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
            $(".backtotoplink").attr("innerHTML", "Back to Top");
            $(".backtotoplink").attr("title", "Back to Top");
            $(".backtotoplink").attr("href", "#Top");
        });
    </script>
```

How it works...

Run the web page. Scroll the page to view the contents. To go back to the top of the page, click on any of the 'Back to Top' links.



See also

Updating URL of ASP.NET hyperlink at runtime

Updating URL of ASP.NET hyperlink at runtime

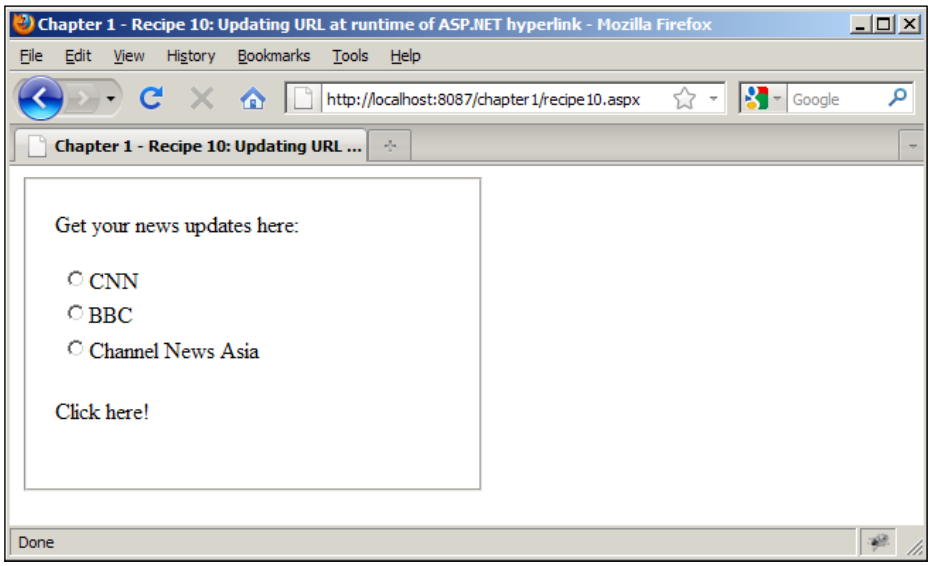
In this recipe, we will update the target URL of an ASP.NET hyperlink control dynamically based on events triggered by other controls.

Getting ready

1. Create a new web form `Recipe10.aspx` in the current project. Add an ASP.NET hyperlink control on the page. Also add a RadioButtonList to the form as follows:

```
    <form id="form1" runat="server">
    <div align="left">
       <fieldset style="width:300px;height:200px;">
       <table cellpadding="0" cellspacing="0" border="0">
       <tr><td width="10px"></td>
       <td>
       <p>Get your news updates here:</p>
        <asp:RadioButtonList ID="RadioButtonList1" runat="server">
        <asp:ListItem Text="CNN" Value="http://www.cnn.com"></
asp:ListItem>
        <asp:ListItem Text="BBC" Value="http://www.bbc.co.uk"></
asp:ListItem>
        <asp:ListItem Text="Channel News Asia" Value="http://www.
channelnewsasia.com"></asp:ListItem>
        </asp:RadioButtonList>
        <br />
        <asp:HyperLink ID="HyperLink1" runat="server">Click
here!</asp:HyperLink>
       </td></tr>
       </table>
    </fieldset>
    </div>
    </form>
```

2. The page will display the content as shown in the screen below:



How to do it...

- In the `document.ready()` function, use the `bind()` function to add the handler for the `"change"` event of the RadioButtonList:

```
$("input[type=radio]").bind("change", function() {
```

> The ASP.NET RadioButtonList control is rendered as `<input type=radio>` HTML element at runtime. Hence, the jQuery selector `$("input[type=radio]")` can be used to retrieve the RadioButtonList.

- Set the `"href"` value of the Hyperlink control to the value of the selected radio button:

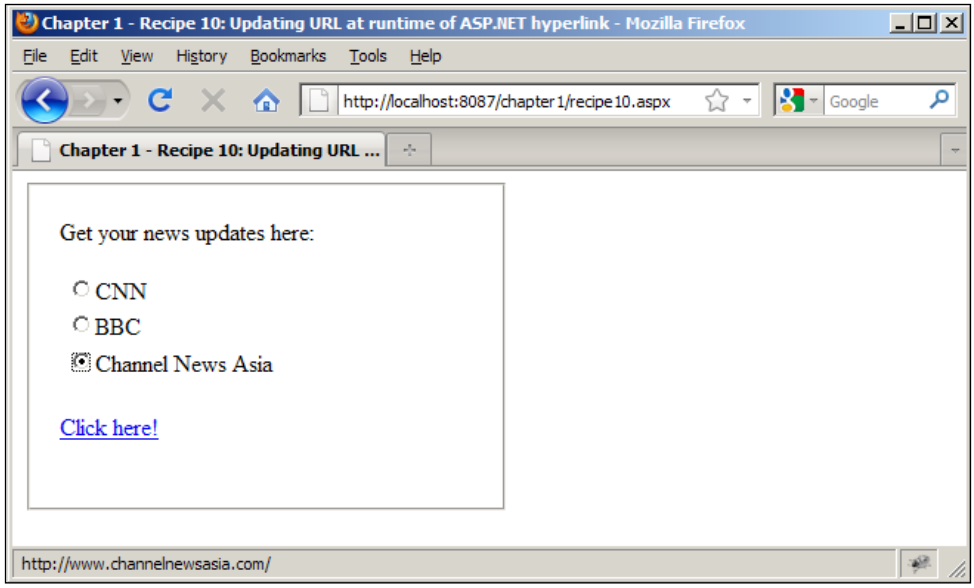
```
$("a").attr("href",$(this).val());
});
```

The complete jQuery solution is as shown below:

```
<script type="text/javascript">
        $(document).ready(function() {
            $("input[type=radio]").bind("change", function() {
                $("a").attr("href",$(this).val());
            });
        });
    </script>
```

How it works...

Run the web page. Select any required option of the RadioButtonList. The URL of the **Click here!** hyperlink will update as per your selection.



There's more...

In addition to the `bind()` function, jQuery also provides `live()` function to tie events to controls. So, the same recipe, can be implemented using `live()` as follows:

```
$("input[type=radio]").live("change", function() {
                $("a").attr("href",$(this).val());
            });
```

The `live()` function attaches event handlers to elements that match the selector now and in the future. The `bind()` function, on the other hand, only attaches event handlers to controls that are added when the DOM is initially loaded.

Thus, `bind()` is suited for pages with static controls while `live()` is suited for pages with dynamic controls.

See also

Creating 'Back to Top' ASP.NET hyperlink.

2
Validation of ASP.NET Controls

In this chapter, we will cover:

- Validation of a sample user login form
- Validation of basic field types in a user profile form
- Character limit validation in Multiline ASP.NET TextBox
- Validation of date range in ASP.NET form
- Validation of ASP.NET CheckBoxList
- Validation of ASP.NET RadioButtonList
- Validation of ASP.NET ListBox Control
- Validation of ASP.NET DropDownList Control

Introduction

In applications requiring user interaction, it is important to ensure that the data entered by the end user meets the specifications of the system. Validation is a set of rules applied to the input data to ensure that the input is as expected. From the point of view of security, this step also helps to block out content that might otherwise corrupt the system. Thus, in web applications, it is a good practice to run through structured validation tests on the input. Validation can be classified into two broad categories: client side validation and server side validation. Client side validation takes place at the browser level before the page is posted to the server. Server side validation, on the other hand, takes place at the web server. Though it is a secure mode of validation, it comes at the cost of performance.

In real world web applications, generally a combination of client side and server side validation is applied. To reduce server load, generally initial validation tests are run on the client before the data is passed to the server for further checks.

ASP.NET comes with a collection of validation controls to work with web forms. The Validation suite consists of the following server controls:

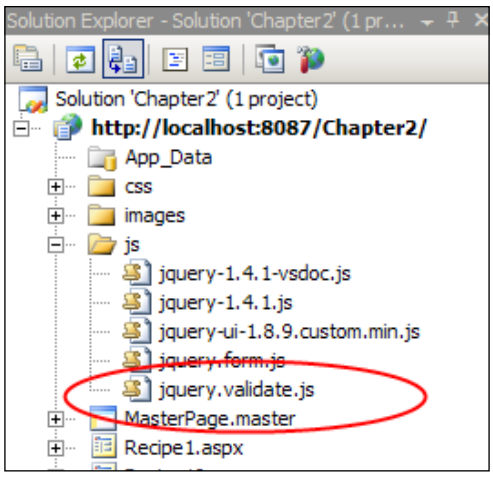
- `RequiredFieldValidator`
- `RangeValidator`
- `RegularExpressionValidator`
- `CompareValidator`
- `CustomValidator`

The above controls can be used for both client side and server side validation.

In addition to ASP.NET validation controls, jQuery can also be interfaced with ASP.NET web forms to perform the required client side validation checks. In this chapter, we will demonstrate the use of jQuery for validation of ASP.NET controls. In addition to applying basic jQuery scripting, we will also use the jQuery validation plugin with web forms.

Getting started

1. To begin with, let's create a new ASP.NET website in Visual Studio and name it `Chapter2`.
2. Include the jQuery library in a sub-folder `js` in the project.
3. Download the jQuery validation plugin from `http://plugins.jquery.com/project/validate` and save the file to the `js` folder.



Usage of validation plugin

jQuery comes with numerous plugins to add on additional functionality to the base library. For handling the nitty-gritty of validation, the jQuery validation plugin is very handy and provides a rich collection of properties and methods for use. This plugin was developed by Jörn Zaefferer, a member of the jQuery team and the lead developer of jQuery UI. It was developed in 2006 and has since then been constantly updated and improvised.

To enable the validation plugin on a web form, all you need to do is drag-and-drop the plugin to the required form. Visual Studio will automatically add the following code to the ASPX page:

```
<scriptsrc="js/jquery.validate.js" type="text/javascript" ></script>
```

To validate a web form using the plugin, we apply the following method on the form element:

```
validate([options])
```

Let's take a quick look at some of the validate `options` that we will use in the recipes described in this chapter:

- `rules`: This consists of key/value pairs. The key is the name of the element while the value is the validation rule to be applied to the element.
- `messages`: This consists of key/value pairs. The key is the name of the element while the value is the custom message to be displayed if validation fails for that element.
- `errorLabelContainer`: This is the actual container area within which the error messages are displayed.
- `errorContainer`: This is the main content area containing the `errorLabelContainer`.
- `wrapper`: This wraps the error messages within the required element and is used in conjunction with `errorLabelContainer`.
- `onsubmit`: Validates the form on submit. Can be set to `false` to validate on other events.
- `highlight`: This option specifies how to highlight the invalid fields.
- `unhighlight`: This option reverts the changes done by the `highlight` option.

We will see more details on the usage of the validate options in the recipes described in this chapter. Now, let's move on to the recipes and see how jQuery can be interfaced in various ways for validation of ASP.NET controls.

For the recipes based on the validate plugin, include this plugin along with the jQuery library on the web form as shown:

```
<scriptsrc="js/jquery-1.4.1.js" type="text/
javascript"></script>
<scriptsrc="js/jquery.validate.js" type="text/
javascript"></script>
```

Validation of a sample user login form

Let's start with validation of a basic user login form using the validation plugin. The idea is to familiarize ourselves with the basics of using this plugin with ASP.NET controls.

Getting ready

1. Add a new web form `Recipe1.aspx` to the current project.
2. Include the jQuery library and the validation plugin on the form.
3. Create a simple login form using ASP.NET controls as shown:

```
<table border="0" cellpadding="3" cellspacing="3">
<tr><td colspan="2" class="header">LOGIN USER</td></tr>
<tr>
<td align="right">
<asp:Label ID="lblUserName" runat="server" Text="UserName: "></
asp:Label>
</td>
<td align="left">
<asp:TextBox ID="txtUserName" runat="server"></asp:TextBox>
</td>
</tr>
<tr>
<td align="right">
<asp:Label ID="lblPassword" runat="server" Text="Password: "></
asp:Label>
</td>
<td align="left">
<asp:TextBox ID="txtPassword" runat="server"
TextMode="Password"></asp:TextBox>
</td>
</tr>
<tr>
<td align="center" colspan="2">
<asp:Button ID="btnSubmit" runat="server" Text="Submit"/> 
<asp:Button ID="btnReset" runat="server" Text="Reset"/>
</td>
</tr>
</table>
```

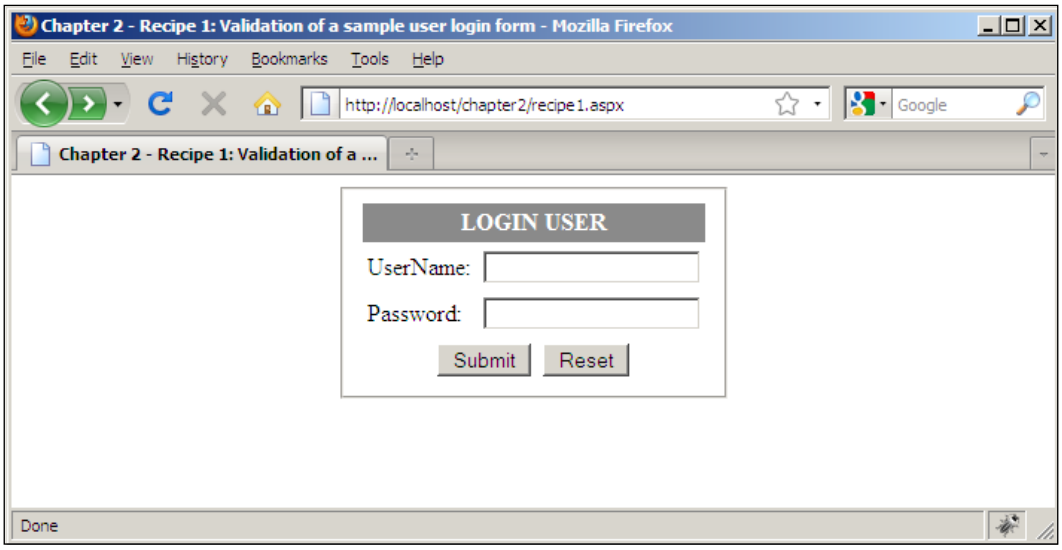
4. Add a div area below the login controls where we will display the error messages:

```
<div align="center" class="alertmsg">
</div>
```

5. Add the following CSS class for the above div area:

```
.alertmsg
    {
  color:#FF0000;
    }
```

6. The page will appear as shown in the next screen:



We will perform the following validations on the login form:

- The `UserName` and `Password` fields are mandatory.
- The minimum length of the `Password` field is eight.
- We will also define custom error messages and display the error labels in the error div area.

How to do it...

1. In the `document.ready()` function of the jQuery script block, call the validate function on the form element:

```
var validator = $("#form1").validate({
```

2. Add the `rules` option to the validate function:

```
rules: {
```

3. Define the `UserName` as a mandatory field:

```
txtUserName: "required",
```

4. Define the `Password` as a mandatory field and define its minimum length as eight:

```
txtPassword: { required: true, minlength: 8 }
                }
                ,
```

5. Add the `messages` option to the validate function:

```
messages: {
```

6. Define a custom error message for `UserName` validation:

```
txtUserName: "Please enter your UserName",
```

7. Define custom error messages for `Password` validation:

```
txtPassword: { required: "Please enter your Password",
                        minlength: "Password should be at least 8
                        characters long"
                    }
                },
```

8. Wrap the error labels in a ListItem element:

```
wrapper: 'li',
```

9. Define the `errorLabelContainer` where the error labels will be displayed when the validation fails:

```
errorLabelContainer: $("#form1 div.alertmsg")
            });
```

10. On the `click` event of the `Reset` button, clear the form fields as shown:

```
$("#btnReset").click(function(e) {
                e.preventDefault();
                $("#txtUserName").val("");
                $("#txtPassword").val("");
            });
```

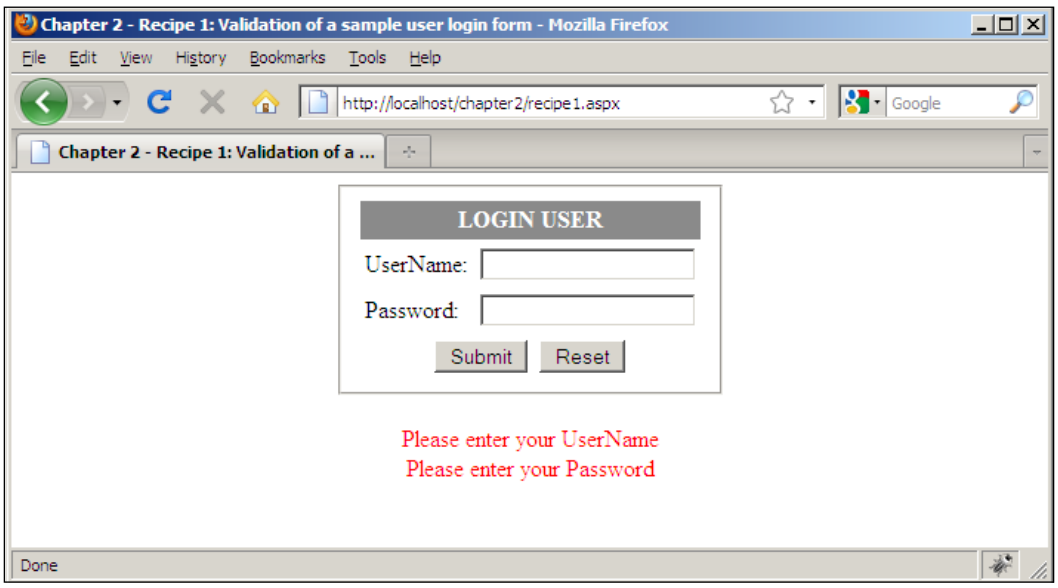
11. Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            var validator = $("#form1").validate({
                rules: {
                    txtUserName: "required",
                    txtPassword: { required: true, minlength: 8 }
                }
                ,
                messages: {
                    txtUserName: "Please enter your UserName",
```

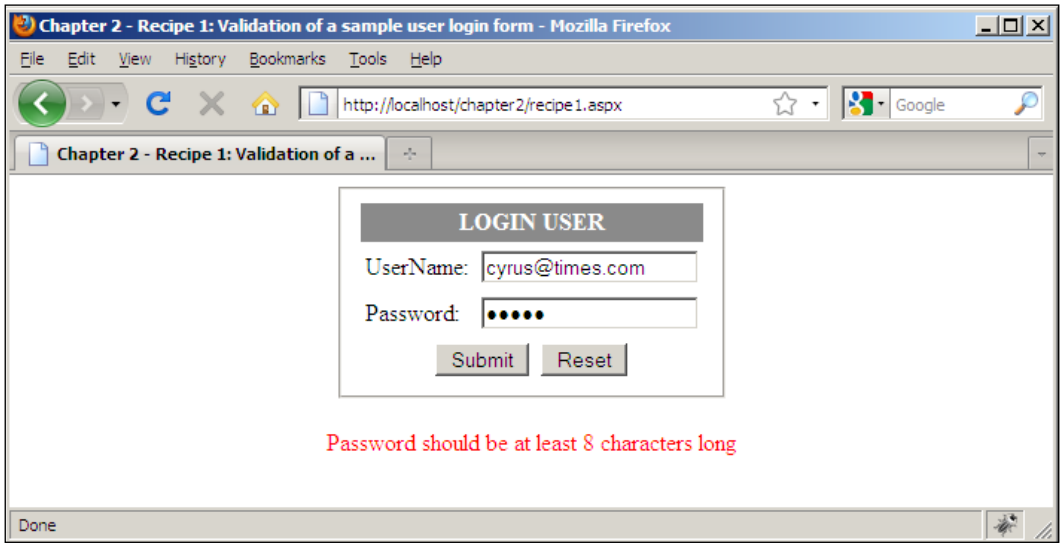
```
                    txtPassword: { required: "Please enter your
Password",
                        minlength: "Password should be at least 8
characters long"
                    }
                },
                wrapper: 'li',
                errorLabelContainer: $("#form1 div.alertmsg")
            });
            $("#btnReset").click(function(e) {
                e.preventDefault();
                $("#txtUserName").val("");
                $("#txtPassword").val("");
            });
        });
</script>
```

How it works...

Run the web form. Click on the **Submit** button without entering any data in the TextBoxes. The page will display the custom error messages as displayed in the following screen:



Now enter some sample data in the TextBoxes such that the Password length is less than eight characters. Click on the **Submit** button. The page will display the following error message:



See also

Validation of basic field types in a user profile form

Validation of basic field types in a user profile form

In this recipe, we will create a simple user profile form for registration. We will then validate the fields on the client side using the jQuery validation plugin.

Getting ready

1. Create a new web form `Recipe2.aspx` in the current project.
2. Create a simple user profile form containing fields such as Name, Email, Password, Date of Birth, Address details, and Website URL, etc. as follows:

```
<fieldset style="width:350px;height:400px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><tdcolspan="2" class="header">REGISTER NEW USER</td></tr>
<tr><tdalign="right">Name<spanclass="mandatory">*</span></td>
<tdalign="left">
<asp:TextBoxID="txtName" runat="server"></asp:TextBox>
```

```
</td></tr>
<tr><tdalign="right">Email<spanclass="mandatory">*</span></td>
<tdalign="left">
<asp:TextBoxID="txtEmail" runat="server"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Password<spanclass="mandatory">*</span></td>
<tdalign="left">
<asp:TextBoxID="txtPassword" runat="server" TextMode="Password"></
asp:TextBox>
</td></tr>
<tr><tdalign="right">Confirm Password<spanclass="mandatory">*</
span></td>
<tdalign="left">
<asp:TextBoxID="txtConfirmPwd" runat="server"
TextMode="Password"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Date of Birth<spanclass="mandatory">*</
span></td>
<tdalign="left">
<asp:TextBoxID="txtDOB" runat="server"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Address 1<spanclass="mandatory">*</span></td>
<tdalign="left">
<asp:TextBoxID="txtAddress1" runat="server"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Address 2</td>
<tdalign="left">
<asp:TextBoxID="txtAddress2" runat="server"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Postal Code<spanclass="mandatory">*</span></
td>
<tdalign="left">
<asp:TextBoxID="txtPostal" runat="server"></asp:TextBox>
</td></tr>
<tr><tdalign="right">Website</td>
<tdalign="left">
<asp:TextBoxID="txtWebsite" runat="server"></asp:TextBox>
</td></tr>
<tr><tdcolspan="2">
<asp:CheckBoxID="chkTandC" runat="server" />
            I accept the Terms and Conditions</td></tr>
<tr><tdcolspan="2" align="center">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
 <asp:Button ID="btnReset" runat="server" Text="Reset" /></
td></tr>
</table>
</fieldset>
```

3. Define the `errorContainer` area on the form as shown:

```
<div align="center" class="errorContainer">
<fieldset style="width:550px;">
</fieldset>
</div>
```

4. Attach the following CSS class to the area:

```
.errorContainer
    {
  display:none;
    }
```

5. Within this `errorContainer` area, we will define a specific `errorLabelContainer` for displaying the error labels, as shown:

```
<div align="center" class="errorContainer">
<fieldset style="width:550px;">
<palign="left" class="alertMsg">
    There was an error processing your request. Please correct the
following to proceed:
</p>
</fieldset>
</div>
```

6. Define the CSS class for the error messages as follows:

```
.alertMsg
    {
  margin-left:20px;
  color:#660000;
    }
```

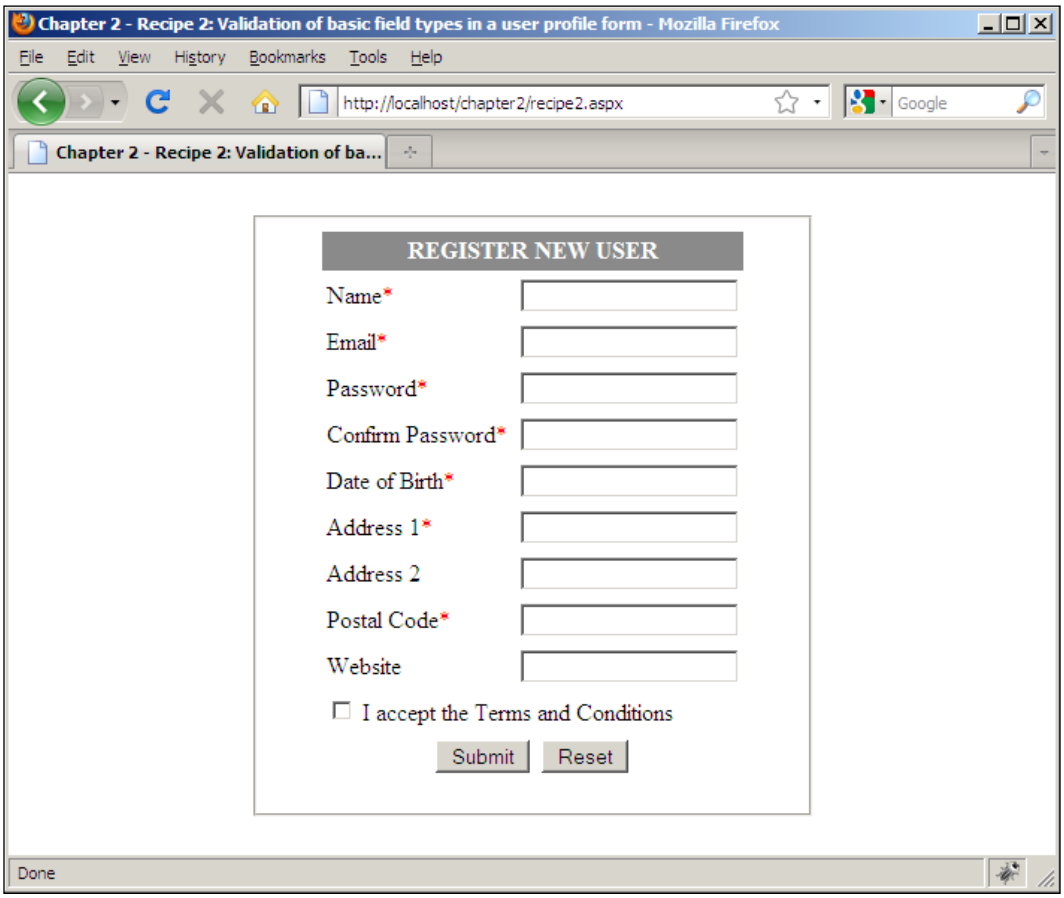
7. For highlighting the invalid fields, define the following CSS class:

```
.input-highlight
    {
  background-color:#CCCCCC;
    }
```

8. Download the jQuery form plugin from `http://plugins.jquery.com/project/form`.
9. Save the plugin locally in the `js` folder in the current project.
10. Include the following form plugin on the web form:

```
<scripttype="text/javascript" src="js/jquery.form.js"></script>
```

11. The page displays the form as shown in the next screen:



We will define the following validations on the form:

- The input fields **Name**, **Email**, **Password**, **Confirm Password**, **Date of Birth**, **Address 1**, **Postal Code**, **Terms and Conditions** are defined as mandatory.
- **Email** is an email address field.
- **Password** and **Confirm Password** fields should match. The minimum length of both fields is eight characters.
- **Date of Birth** is a date field.
- The maximum length of both **Address 1** and **Address 2** fields is 100 characters.
- **Postal Code** is a numeric field.
- Website is a URL field.
- The **Terms and Conditions** should be checked for successful form submission.

How to do it...

1. In the `document.ready()` function of the jQuery script block, set the default options for the `validate()` function:

   ```
   $.validator.setDefaults({
   ```

2. Use the `highlight` option to add the `input-highlight` class to invalid fields:

   ```
   highlight: function(input) {
                       $(input).addClass("input-highlight");
                   },
   ```

3. Use the `unhighlight` option to remove the `input-highlight` CSS class from the fields:

   ```
   unhighlight: function(input) {
                       $(input).removeClass("input-highlight");
                   }
               });
   ```

4. After setting the default options, now call the `validate()` function on the web form:

   ```
   var validator = $("#form1").validate({
   ```

5. Add rules for validation:

   ```
   rules: {
   ```

6. Define `Name` as a mandatory field:

   ```
   txtName: "required",
   ```

7. Define `Email` as mandatory. Add a check for email address validation:

   ```
   txtEmail: { required: true, email: true },
   ```

8. Define `Password` as mandatory. Specify its minimum length as eight characters:

   ```
   txtPassword: { required: true, minlength: 8 },
   ```

9. Define `Confirm Password` as mandatory. Specify its minimum length as eight characters and its text should be the same as the `Password` field:

   ```
   txtConfirmPwd: { required: true, minlength: 8, equalTo:
   "#txtPassword" },
   ```

10. Define `Date of Birth` as a mandatory date field:

    ```
    txtDOB: { required: true, date: true },
    ```

11. Define `Address 1` as mandatory and its maximum length as 100 characters:

    ```
    txtAddress1: { required: true, maxlength: 100 },
    ```

12. Define maximum length of `Address 2` as 100 characters:

```
txtAddress2: { maxlength: 100 },
```

13. Define `Postal Code` as a mandatory numeric field:

```
txtPostal: { required: true, digits: true },
```

14. Define `Website` as a URL field:

```
txtWebsite: { url: true },
```

15. The `Terms and Conditions` should be checked for successful form submission:

```
chkTandC: "required"
                },
```

16. For each of the above validations attached to the controls, define the corresponding custom error labels:

```
messages: {
txtName: "Please enter your Name",
    txtEmail: { required: "Please enter your Email",
                email: "Please enter a valid email address"
    },
    txtPassword: { required: "Please enter your Password",
                   minlength: "Password should be at least 8
                   characters long"
},
    txtConfirmPwd: { required: "Please reenter your Password to
                     confirm",
                     minlength: "The Confirm Password should be at
                     least 8 characters long",
                     equalTo: "The entered password and confirm
                     password should match"
    },
     txtDOB: { required: "Please enter your Date of Birth",
               date: "Please enter a valid date"
    },
     txtAddress1: { required: "Please enter your Mailing Address",
                    maxlength: "Address can be upto maximum 100
                    characters"
    },
     txtAddress2: { maxlength: "Address can be upto maximum 100
                    characters"
    },
     txtPostal: { required: "Please enter the Postal Code",
                  digits: "Please enter a valid postal code"
    },
     txtWebsite: { url: "Please enter a valid URL"
```

```
},
 chkTandC: { required: "Please accept the Terms & Conditions
 to proceed" }
},
```

17. Specify the `wrapper` element for the error labels:

```
wrapper: 'li',
```

18. Specify the `errorContainer` area using the jQuery CSS selector:

```
errorContainer: $("div.errorContainer"),
```

The jQuery selector `$("div .errorContainer")` returns all elements within div elements on the form that have the CSS `errorContainer` class attached to them.

19. Within the above `errorContainer` area, specify the required `errorLabelContainer` using the jQuery CSS selector:

```
errorLabelContainer: $("#form1 p.alertMsg")
});
```

The jQuery selector `$("#form1 p.alertMsg")` returns all elements within <p> tags in the form with ID `form1` that have the CSS class `alertMsg` attached to them.

20. On the `click` event of the `Reset` button, use the `resetForm()` function of the jQuery form plugin to reset the form data:

```
$("#btnReset").click(function(e) {
                e.preventDefault();
                $("#form1").resetForm();
            });
```

At one glance, the complete jQuery solution for the form validation is as follows:

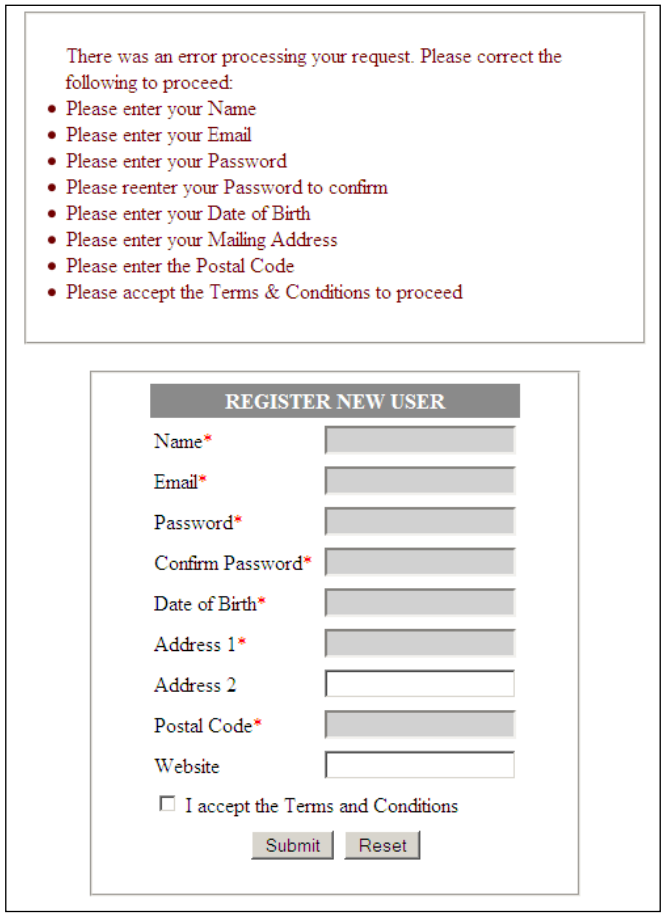
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $.validator.setDefaults({
                highlight: function(input) {
                    $(input).addClass("input-highlight");
                },
                unhighlight: function(input) {
                    $(input).removeClass("input-highlight");
                }
            });
            var validator = $("#form1").validate({
                rules: {
```

```
        txtName: "required",
        txtEmail: { required: true, email: true },
        txtPassword: { required: true, minlength: 8 },
        txtConfirmPwd: { required: true, minlength: 8,
        equalTo: "#txtPassword" },
        txtDOB: { required: true, date: true },
        txtAddress1: { required: true, maxlength: 100 },
        txtAddress2: { maxlength: 100 },
        txtPostal: { required: true, digits: true },
        txtWebsite: { url: true },
        chkTandC: "required"

    },
    messages: {
        txtName: "Please enter your Name",
        txtEmail: { required: "Please enter your Email",
            email: "Please enter a valid email address"
        },
        txtPassword: { required: "Please enter your
        Password",
            minlength: "Password should be at least 8
            characters long"
        },
        txtConfirmPwd: { required: "Please reenter your
        Password to confirm",
            minlength: "The Confirm Password should be at
            least 8 characters long",
            equalTo: "The entered password and confirm
            password should match"
        },
        txtDOB: { required: "Please enter your Date of
        Birth",
            date: "Please enter a valid date"
        },
        txtAddress1: { required: "Please enter your
        Mailing Address",
            maxlength: "Address can be upto maximum 100
            characters"
        },
        txtAddress2: { maxlength: "Address can be upto
        maximum 100 characters"
        },
        txtPostal: { required: "Please enter the Postal
        Code",
            digits: "Please enter a valid postal code"
        },
        txtWebsite: { url: "Please enter a valid URL"
        },
        chkTandC: { required: "Please accept the Terms &
        Conditions to proceed" }
```

```
                },
                wrapper: 'li',
                errorContainer: $("div.errorContainer"),
                errorLabelContainer: $("#form1 p.alertMsg")
            });
            $("#btnReset").click(function(e) {
                e.preventDefault();
                $("#form1").resetForm();
            });
        });
</script>
```

How it works...

Run the web form. Enter different test data in the form and click **Submit**. The page will display the corresponding error messages as demonstrated next:



See also

Validation of a sample user login form

Character limit validation in Multiline ASP.NET TextBox

In Multiline ASP.NET TextBoxes, there is often a need to restrict the number of input characters. In this recipe, we will provide a character counter for a Multiline ASP.NET TextBox using jQuery. We will also validate the character range in the control.

Getting ready

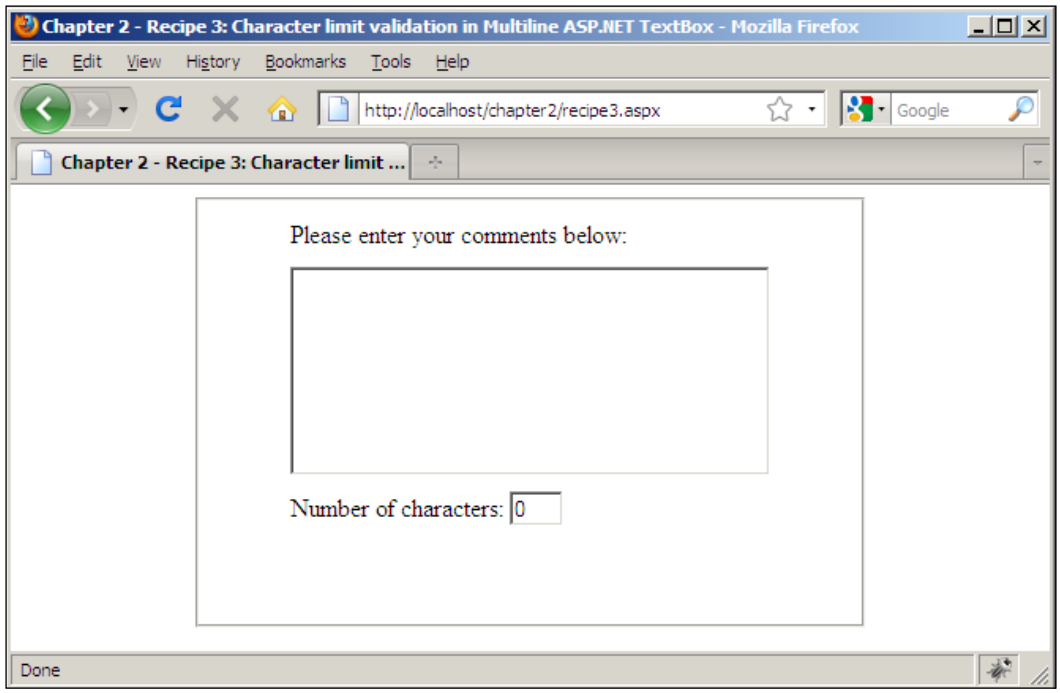
1. Add a new web form `Recipe3.aspx` to the current project.
2. Add a Multiline TextBox to the form. In addition, add a **ReadOnly** TextBox where we will display the number of characters currently in the Multiline TextBox:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:400px;height:250px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><td>Please enter your comments below:</td></tr>
<tr><td><asp:TextBoxID="TextBox1" runat="server"
TextMode="MultiLine" Rows="7" Width="300px"></asp:TextBox>
</td></tr>
<tr><td>Number of characters:
<asp:TextBoxID="TextBox2" runat="server" Width="30px"
ReadOnly="true"></asp:TextBox></td></tr>
<tr><td><divid="message" class="alertmsg"></div></td></tr>
</table>
</fieldset>
</div>
</form>
```

3. In the above form, we have also appended a div area `message` to display error messages.
4. Attach the following CSS class to the div area:

```
.alertmsg
    {
  color:#FF0000;
    }
```

5. The page will display the form as follows:



We will define the following validations on the form:

- Minimum number of characters in the Multiline TextBox is 5
- Maximum allowable characters in the Multiline TextBox is 200

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the minimum character count allowable in the Multiline TextBox:

   ```
   var minCount = 5;
   ```

2. Similarly, define the maximum character count:

   ```
   var maxCount = 200;
   ```

3. Disable the `cut/copy/paste` operations in the Multiline TextBox so that we can use `keypress` event to update the character count at runtime:

```
$("#TextBox1").bind("cut copy paste", function(e) {
                e.preventDefault();
            });
```

4. Define the `keypress` function of the Multiline TextBox:

```
$("#TextBox1").keypress(function() {
```

We use the `keypress` event instead of the `keyup` event so that when we hold down any key in the Multiline TextBox, the count is updated and displayed. If `keyup` is used, the count will be updated only when the key is released.

5. Get the current character count in the Multiline TextBox:

```
var strCount = $("#TextBox1").val().length;
```

6. Set the text of the **ReadOnly** TextBox to the current character count:

```
$("#TextBox2").val(strCount);
```

7. Check if the current character count falls in the range `minCount` to `maxCount` as shown:

```
if ((strCount < minCount) || (strCount > maxCount)) {
```

8. If the count is out of range then display an error message:

```
$("#message").text("Please key in characters in the range 5 -
200");
                }
```

9. If the count is within range, clear the error div area:

```
else {
                    $("#message").text("");
                }
            });
```

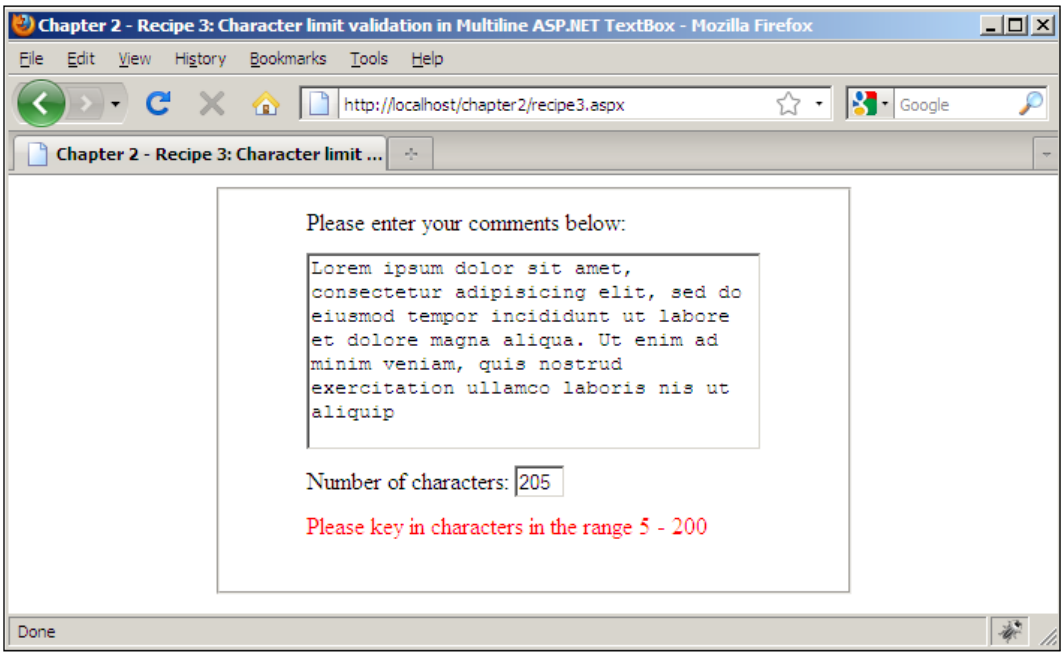
The complete jQuery solution is as follows:

```
<scripttype="text/javascript">
        $(document).ready(function() {
            var minCount = 5;
            var maxCount = 200;
            $("#TextBox1").bind("cut copy paste", function(e) {
                e.preventDefault();
            });
```

```
            $("#TextBox1").keypress(function() {
            var strCount = $("#TextBox1").val().length;
                $("#TextBox2").val(strCount);
                if ((strCount < minCount) || (strCount > maxCount)) {
                    $("#message").text("Please key in characters in
                    the range 5 - 200");
                }
                else {
                    $("#message").text("");
                }
            });
        });
</script>
```

How it works...

Run the web form. Enter any text in the Multiline TextBox. You will notice that the character count is updated dynamically. Also, when the character count is outside the allowable range, the system throws an error message as shown next:



See also

Validation of date range in ASP.NET Form

Validation of date range in ASP.NET Form

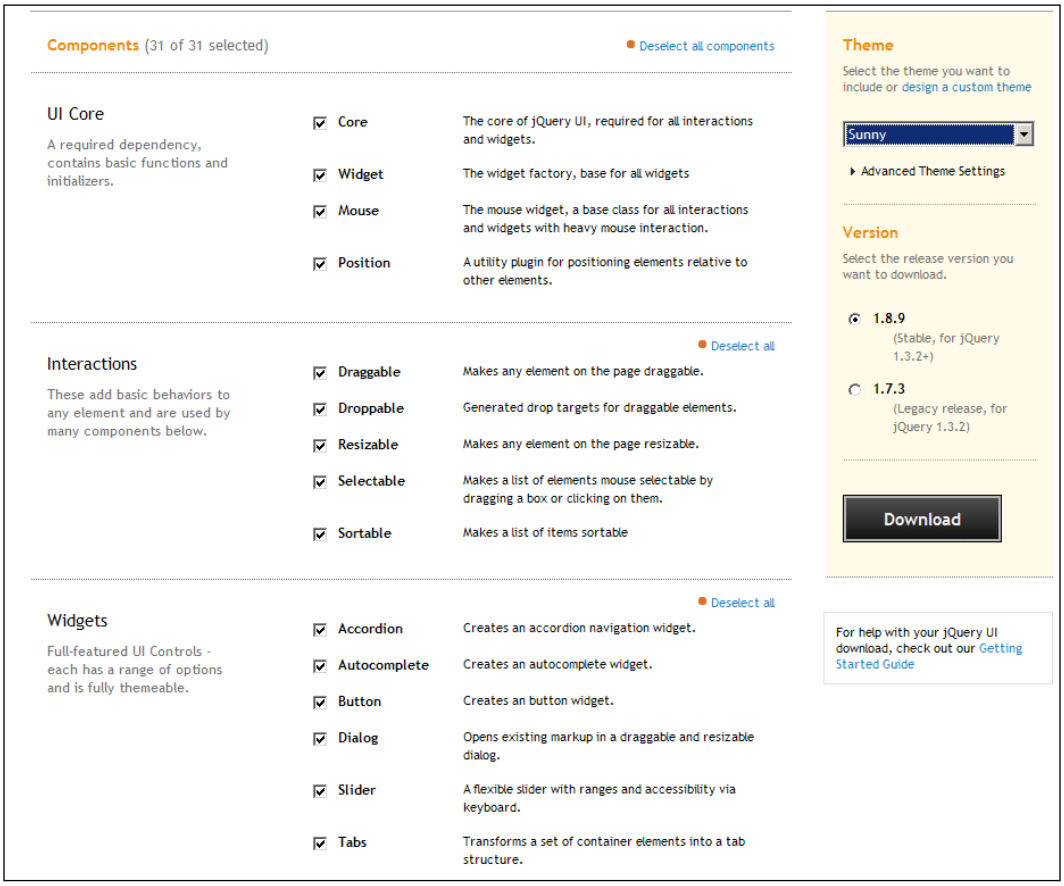
In this recipe, we will define the start date and end date fields in a form. We will then validate if the end date is later than the start date.

Getting ready

1. Add a new web form `Recipe4.aspx` in the current project.
2. Add TextBox controls to the form for the Start Date and End Date fields as follows:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:400px;height:150px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><tdcolspan="2">Please select the date range:</td></tr>
<tr><td>Start Date:</td>
<td><asp:TextBoxID="txtStartDate" runat="server"></asp:TextBox></
td></tr>
<tr><td>End Date:</td>
<td><asp:TextBoxID="txtEndDate" runat="server"></asp:TextBox></
td></tr>
<tr><tdcolspan="2">
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
<asp:Button ID="btnReset" runat="server" Text="Reset" /></td></tr>
</table>
</fieldset>
</div>
</form>
```

3. We will use the jQuery UI datepicker widget in this recipe. Go to the download page of jQuery UI at `http://jqueryui.com/download`, which allows customizable downloads based on the features required in the web application. Download the default build and save locally to the `js` folder. The page also allows downloading the required theme. We have selected the **Sunny** theme for this recipe:



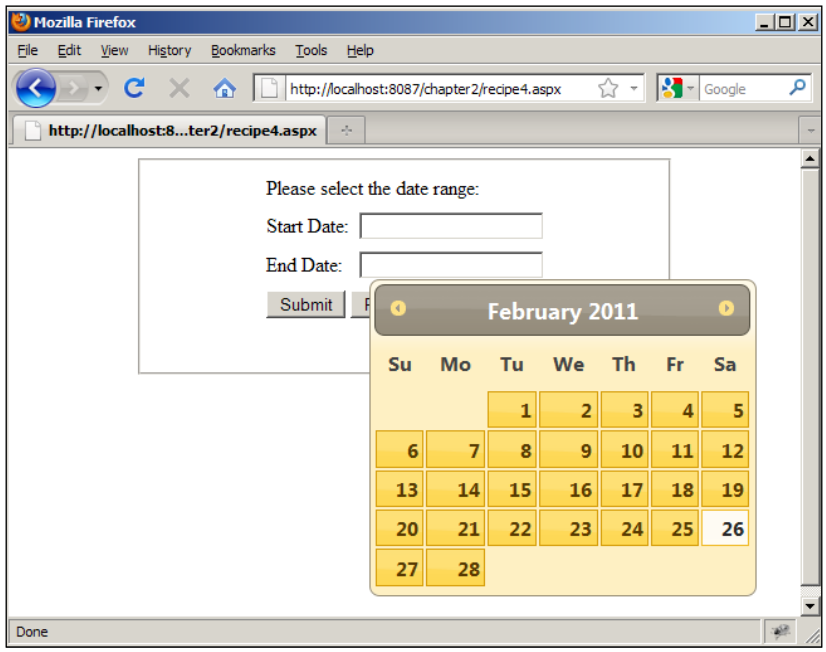
4. In addition to the jQuery library, include jQuery UI as well as the stylesheet on the page as follows:

```
<linkhref="css/sunny/jquery-ui-1.8.9.custom.css" rel="stylesheet"
type="text/css" />
<scriptsrc="js/jquery-1.4.1.js" type="text/javascript"></script>
<scriptsrc="js/jquery-ui-1.8.9.custom.min.js" type="text/
javascript"></script>
```

5. In the `document.ready()` function of the jQuery script block, attach the datepicker widget as follows:

```
$("#txtStartDate").datepicker();
$("#txtEndDate").datepicker();
```

6. Thus, the page displays the datepicker as shown in the next screen:



How to do it...

1. In the `document.ready()` function, define the callback function on the `click` event of the **Submit** button:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Read the Start Date from the input field:

```
var startdate = new Date();
startdate = Date.parse($("#txtStartDate").val());
```

4. Read the End Date from the input field:

```
var enddate = new Date();
enddate = Date.parse($("#txtEndDate").val());
```

5. Check if the End Date is later than the Start Date:

```
if (startdate > enddate) {
```

6. If yes, then display an error message to the user:

```
alert("StartDate is later than EndDate");
}
            });
```

7. Clear the form when the **Reset** button is clicked:

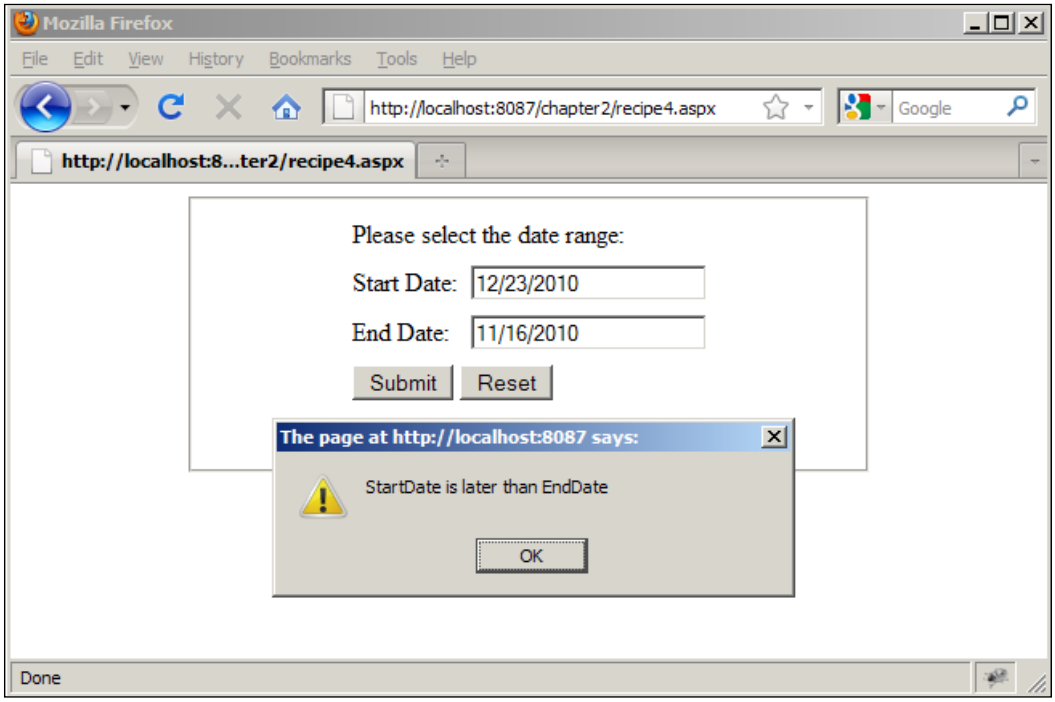
```
$("#btnReset").click(function(e) {
                e.preventDefault();
                $("#txtStartDate").val("");
                $("#txtEndDate").val("");
            });
```

Thus, the complete jQuery solution is as shown:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#txtStartDate").datepicker();
            $("#txtEndDate").datepicker();

            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                var startdate = new Date();
                startdate = Date.parse($("#txtStartDate").val());
                var enddate = new Date();
                enddate = Date.parse($("#txtEndDate").val());
                if (startdate > enddate) {
                    alert("StartDate is later than EndDate");
                }
            });
            $("#btnReset").click(function(e) {
                e.preventDefault();
                $("#txtStartDate").val("");
                $("#txtEndDate").val("");
            });
        });
</script>
```

How it works...

Run the web form. Enter any test dates in the input form. When the Start Date is later than the End Date, the system throws an error message as shown in the following screen:



See also

Validation of ASP.NET CheckBoxList

Validation of ASP.NET CheckBoxList

In this recipe, we will demonstrate the use of ASP.NET's CustomValidator control for validating a CheckBoxList. We will write the required client validation logic using jQuery.

Getting ready

1. Add a new web form `Recipe5.aspx` in the current project.

2. Add a CheckBoxList control on the form and populate some sample data:

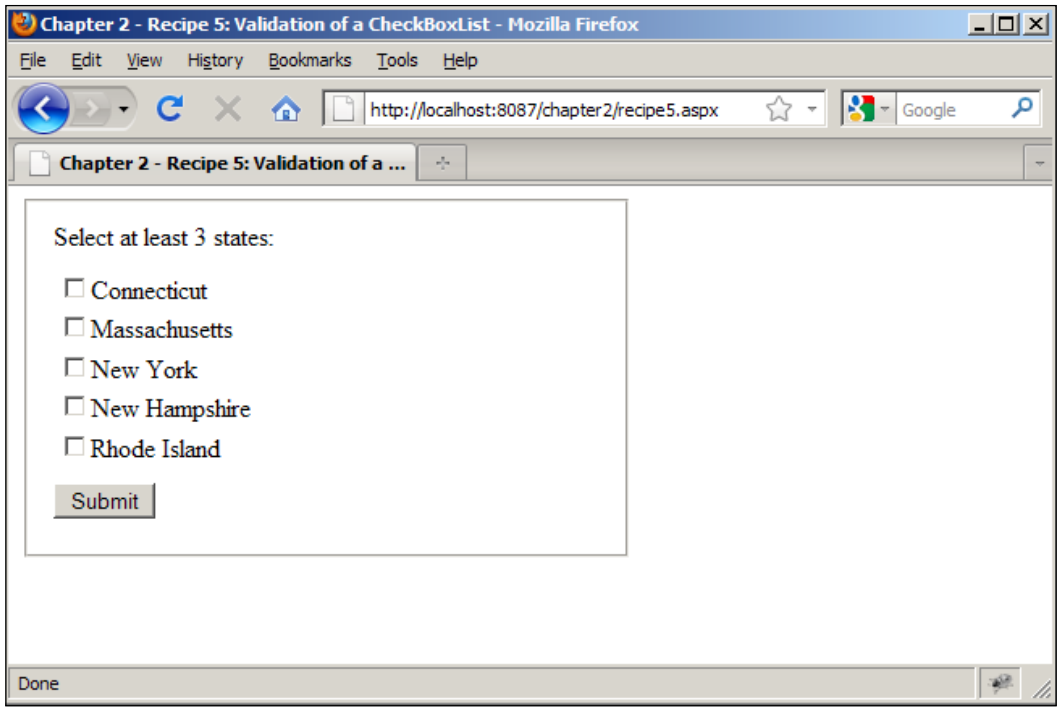
```
<fieldset style="width:350px;height:200px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><td>
    Select at least 3 states:</td></tr>
<tr><td>
<asp:CheckBoxListID="CheckBoxList1" runat="server">
<asp:ListItemText="Connecticut" Value="CT"></asp:ListItem>
<asp:ListItemText="Massachusetts" Value="MA"></asp:ListItem>
<asp:ListItemText="New York" Value="NY"></asp:ListItem>
<asp:ListItemText="New Hampshire" Value="NH"></asp:ListItem>
<asp:ListItemText="Rhode Island" Value="RI"></asp:ListItem>
</asp:CheckBoxList>
</td></tr>
<tr><td><asp:Button ID="btnSubmit" runat="server" Text="Submit"
/></td></tr>
</table>
</fieldset>
```

3. To validate that, at least three items are selected from the CheckBoxList. Add the following ASP.NET CustomValidator control to the form:

```
<asp:CustomValidatorID="CustomValidator1" runat="server"
Display="Dynamic" ErrorMessage="Please select at least 3
states" ClientValidationFunction="CheckBoxList1_Validation"></
asp:CustomValidator>
```

- `ErrorMessage` is set to a custom error message.
- A `ClientValidationFunction` is defined for client side validation. This function is written in jQuery and we will cover it later in this recipe.
- The `Display` is set to `Dynamic` so that no content is displayed when the validation is successful.

4. The page displays the form as shown in the following screen:



How to do it...

1. In the script block, define the `ClientValidationFunction` of the `CustomValidator` control:

   ```
   function CheckBoxList1_Validation(sender, args)
       {
   ```

2. Default the validation return value to `false`:

   ```
   args.IsValid = false;
   ```

3. Get the total count of checked items from the CheckBoxList as follows:

   ```
   var cnt = $("#CheckBoxList1 input:checked").length;
   ```

4. Return the validation result by checking if at least three items have been selected:

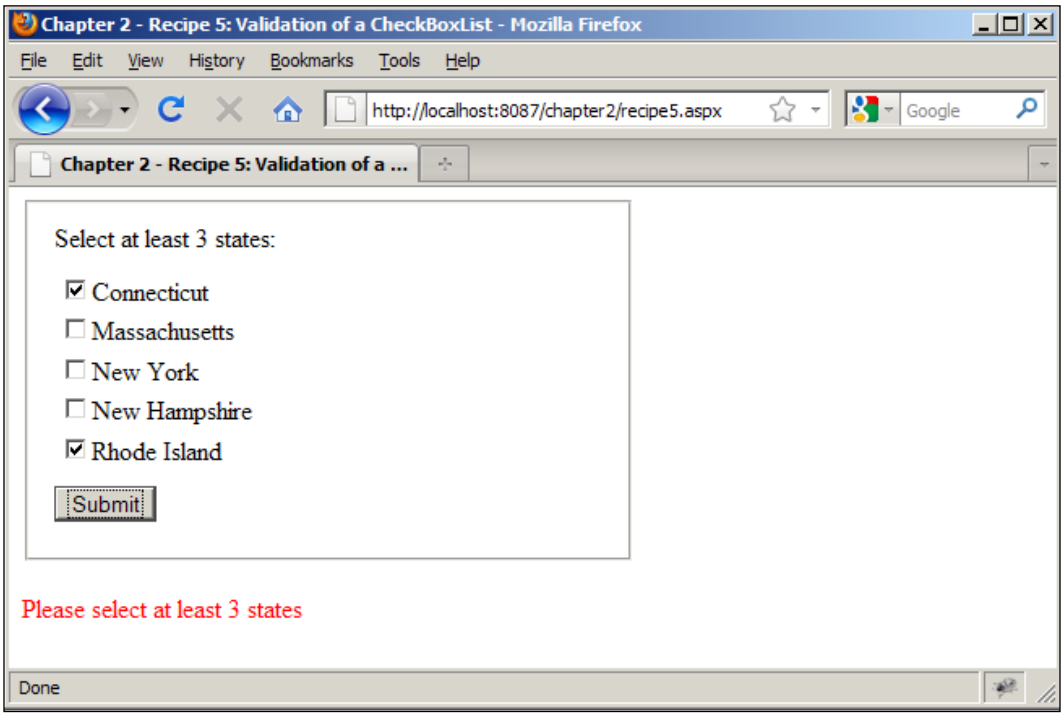
   ```
   args.IsValid = (cnt >= 3);
   }
   ```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
    function CheckBoxList1_Validation(sender, args)
    {
        args.IsValid = false;
        var cnt = $("#CheckBoxList1 input:checked").length;
        args.IsValid = (cnt >= 3);

    }
</script>
```

How it works...

Run the web form. Check the required number of items and click **Submit**. If less than three items are selected, the system displays an error message as shown in the following screen:



On selecting at least three checkboxes, the form is validated successfully and the error message is no longer displayed.

See also

Validation of ASP.NET RadioButtonList

Validation of ASP.NET RadioButtonList

ASP.NET's CustomValidator control can be used with jQuery to validate RadioButtonList controls on the client side. In this recipe, we will simulate ASP.NET's RequiredFieldValidator for validating a RadioButtonList using the same.

Getting ready

1. Create a new web form `Recipe6.aspx` in the current project.
2. Add a RadioButtonList to the page and populate with sample data as shown:

```
<fieldset style="width:350px;height:200px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><td>Please select a state:</td></tr>
<tr><td>
<asp:RadioButtonListID="RadioButtonList1" runat="server">
<asp:ListItemText="Connecticut" Value="CT"></asp:ListItem>
<asp:ListItemText="Massachusetts" Value="MA"></asp:ListItem>
<asp:ListItemText="New York" Value="NY"></asp:ListItem>
<asp:ListItemText="New Hampshire" Value="NH"></asp:ListItem>
<asp:ListItemText="Rhode Island" Value="RI"></asp:ListItem>
</asp:RadioButtonList>
</td></tr>
<tr><td>
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td></tr>
</table>
</fieldset>
```

3. To validate that the RadioButtonList is selected, add a `CustomValidator` control to the page as follows:

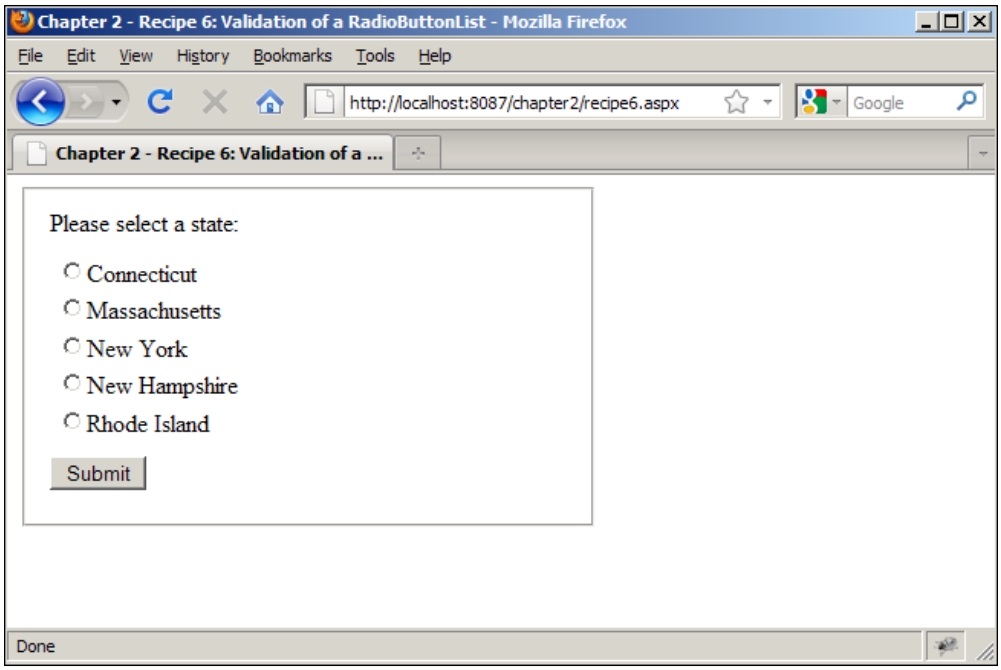
```
<asp:CustomValidatorID="CustomValidator1" runat="server"
ErrorMessage="Please select a state" Display="Dynamic" ClientValid
ationFunction="RadioButtonList1_Validate"></asp:CustomValidator>
```

Note that we have set the following properties of the CustomValidator:

- `ErrorMessage` is set to a custom error message.
- A `ClientValidationFunction` is defined for client side validation. This function is written in jQuery and we will cover it later in this recipe.

- The `Display` is set to `Dynamic` so that no content is displayed when the validation is successful.

4. The page displays the form as shown in the following screenshot:



How to do it...

1. In the script block, add the `ClientValidationFunction` of the CustomValidator control:

```
function RadioButtonList1_Validate(sender, args) {
```

2. Retrieve the number of selected options of the RadioButtonList control and return the validation result:

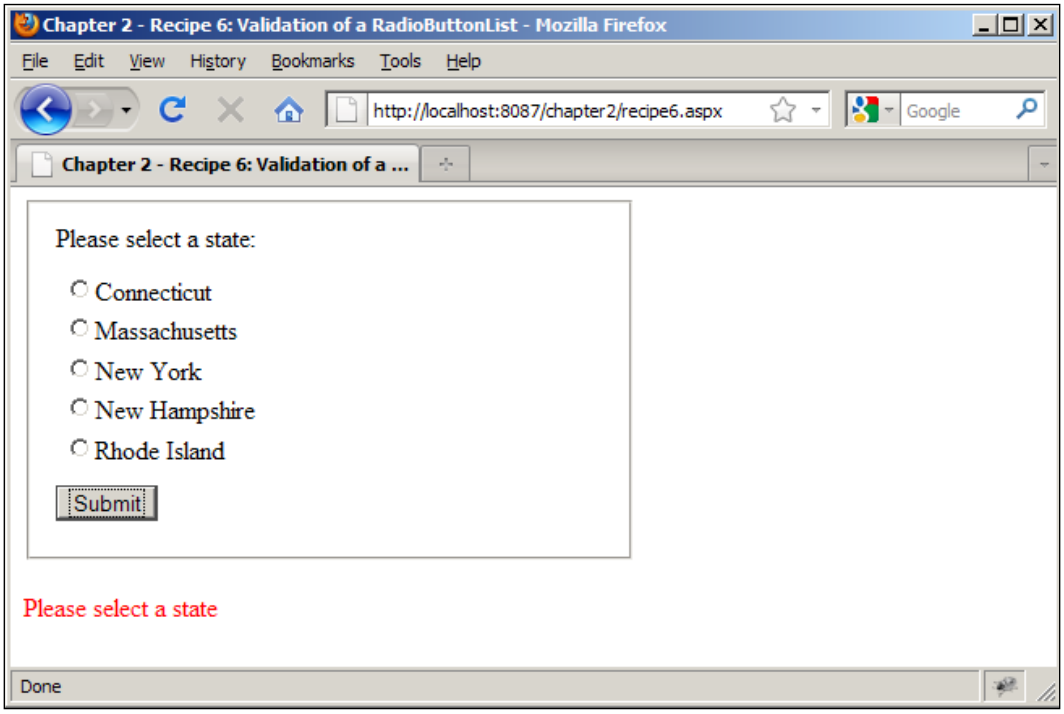
```
args.IsValid = ($("#RadioButtonList1 :radio:checked").length > 0);
}
```

Thus, the complete jQuery solution is as shown:

```
<script language="javascript" type="text/javascript">
function RadioButtonList1_Validate(sender, args) {
            args.IsValid = ($("#RadioButtonList1 :radio:checked").
length > 0);
        }
</script>
```

How it works...

Run the web form. Submit the form without selecting any option from the RadioButtonList. The page will throw the following validation error:



See also

Validation of ASP.NET CheckBoxList

Validation of ASP.NET ListBox Control

In this recipe, let's use the jQuery validation plugin to run validation checks on ASP.NET's ListBox control.

Getting ready

1. Add a new web form `Recipe7.aspx` to the current project.
2. Include the validation plugin on the page.

3. Add a ListBox control to the page and populate with sample data:

```
<fieldset style="width:350px;height:190px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><td>
      Select the required states:
</td></tr>
<tr><td>
<asp:ListBoxID="ListBox1" runat="server" Rows="5"
SelectionMode="Multiple">
<asp:ListItemText="Connecticut" Value="CT"></asp:ListItem>
<asp:ListItemText="Massachusetts" Value="MA"></asp:ListItem>
<asp:ListItemText="New York" Value="NY"></asp:ListItem>
<asp:ListItemText="New Hampshire" Value="NH"></asp:ListItem>
<asp:ListItemText="Rhode Island" Value="RI"></asp:ListItem>
</asp:ListBox>
</td></tr>
<tr><td>
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td></tr>
</table>
</fieldset>
```

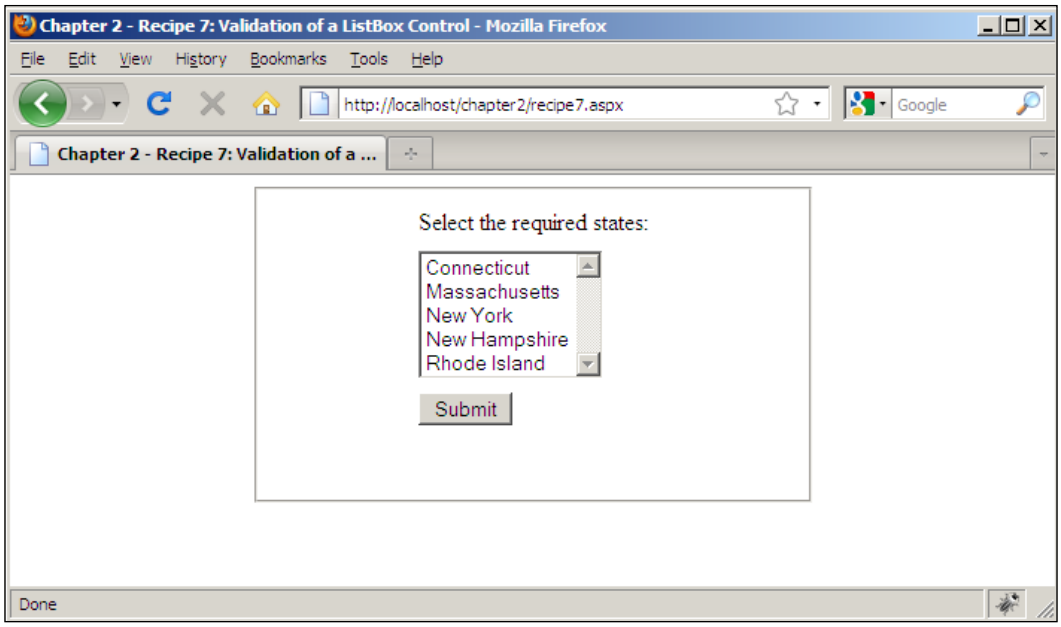
4. Add the following div area to display the error messages:

```
<divid="message" class="alertmsg"></div>
```

5. Define the CSS style for the above div area as follows:

```
.alertmsg
    {
  color:#FF0000;
    }
```

6. Thus, the page displays the form as shown in the following screen:



We will run the following validation checks on the ListBox:

- The ListBox is mandatory. i.e. at least one item is selected.
- The ListBox should allow a maximum of four items to be selected.

How to do it...

1. In the `document.ready()` function of the jQuery script block, call the `validate()` function on the web form:

   ```
   var validator = $("#form1").validate({
   ```

2. Define the `rules` option for the `validate()` function:

   ```
   rules: {
   ```

3. Define the ListBox control as mandatory (at least one item should be selected). Also define a range consisting of the minimum and maximum items allowed to be selected:

   ```
   ListBox1: { required: true, rangelength: [0,4] }
                   },
   ```

4. For each of the above validation rules, define custom error messages:

```
messages: {
    ListBox1: {required: "Please select a state",
        rangelength: "Please select maximum 4 states"
              }
    },
```

5. Define the `errorLabelContainer` where the error messages will be displayed if validation fails:

```
errorLabelContainer: $("#message")
});
```

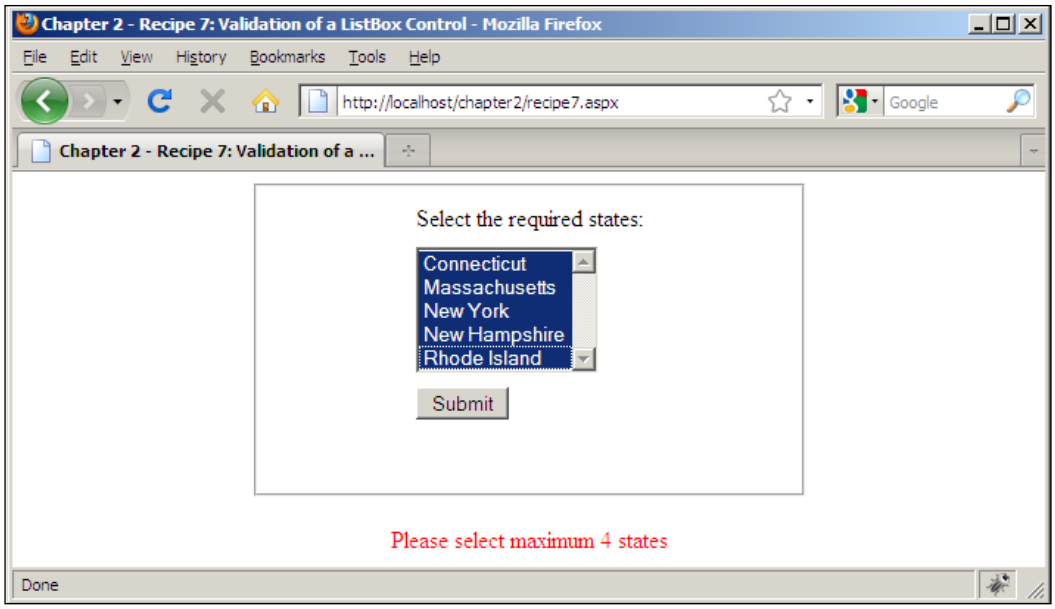
The complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
   var validator = $("#form1").validate({
      rules: {
      ListBox1: { required: true, rangelength: [0,4] }
                },
      messages: {
      ListBox1: {required: "Please select a state",
                 rangelength: "Please select maximum 4 states"
                }
            },
      errorLabelContainer: $("#message")
        });
    });
</script>
```

How it works...

Run the web page. On clicking the **Submit** button, the system forces a mandatory selection of an item from the list.

On selecting more than the maximum allowable items from the ListBox, the system will throw the error message as follows:



See also

Validation of ASP.NET DropDownList Control

Validation of ASP.NET DropDownList Control

In this recipe, let's use the jQuery validation plugin to validate an ASP.NET DropDownList control.

Getting ready

1. Add a new web form `Recipe8.aspx` to the current project.
2. Include the validation plugin on the page.
3. Add a DropDownList control to the form and attach a CSS class `required` to the control. Populate with sample data as shown:

```
<fieldset style="width:350px;height:100px;">
<table border="0" cellpadding="3" cellspacing="3">
<tr><td>
    Please select a state:
</td></tr>
<tr><td>
```

```
<asp:DropDownListID="DropDownList1" runat="server" ToolTip="Please
select a state" CssClass="required">
<asp:ListItemText="" Value="">--Please Select--</asp:ListItem>
<asp:ListItemText="Connecticut" Value="CT"></asp:ListItem>
<asp:ListItemText="Massachusetts" Value="MA"></asp:ListItem>
<asp:ListItemText="New York" Value="NY"></asp:ListItem>
<asp:ListItemText="New Hampshire" Value="NH"></asp:ListItem>
<asp:ListItemText="Rhode Island" Value="RI"></asp:ListItem>
</asp:DropDownList>
</td></tr>
<tr><td>
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
</td></tr>
</table>
</fieldset>
```

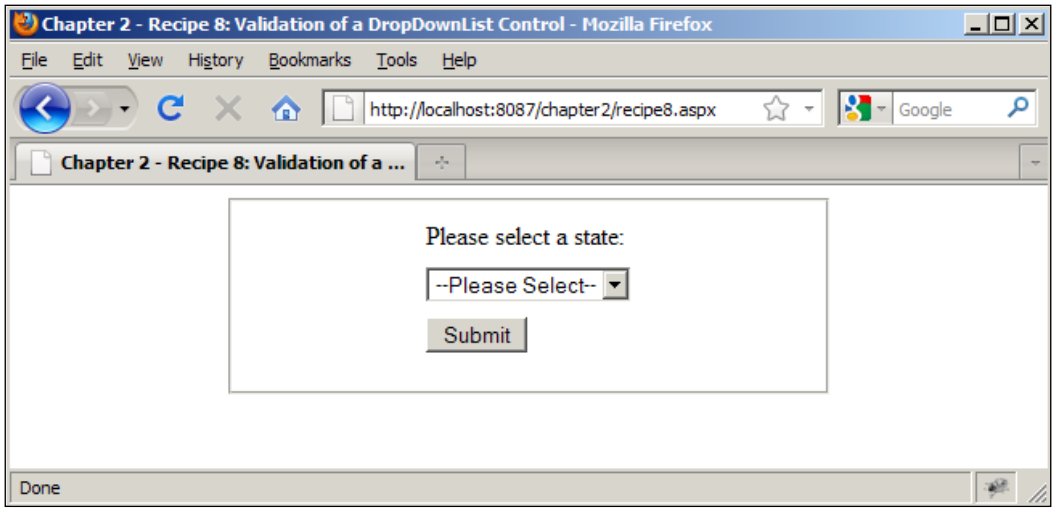
4. Add a div area for displaying the required error messages:

```
<divid="message" class="alertmsg"></div>
```

5. Attach the following CSS class to the above div area:

```
.alertmsg
    {
  color:#FF0000;
    }
```

6. The page will display the form as shown in the next screen:



How to do it...

1. In the `document.ready()` function of the jQuery script block, call the `validate()` function on the web form:

   ```
   var validator = $("#form1").validate({
   ```

2. Define the `errorLabelContainer` to display the error messages when validation fails:

   ```
   errorLabelContainer: $("#message")
   ```

At runtime, ASP.NET renders the DropDownList control as follows:

```
<selectname="DropDownList1" id="DropDownList1"
title="Please select a state" class="required">…..</
select>
```

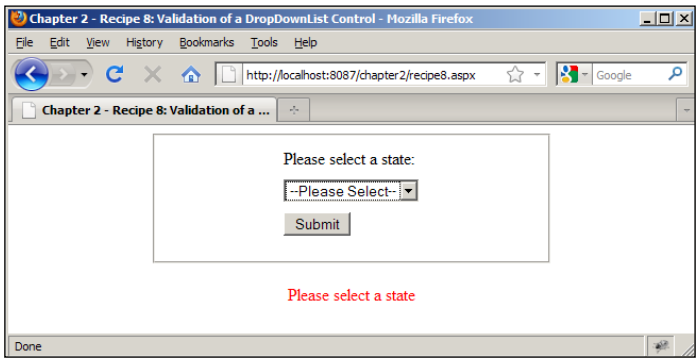
The CSS class `required` is recognized by the jQuery validation plugin as a mandatory field. When validation fails, the `title` text of the element is displayed as the error message. In the absence of the `title` text, the plugin adds the default validation error message `This field is required`.

The complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            var validator = $("#form1").validate({
               errorLabelContainer: $("#message")
        });
    });
</script>
```

How it works...

Run the web form. Submit the form without selecting any item from the DropDownList. The page displays the error message as shown in the next screen:



There's more...

An alternative way of applying the `required` validation rule to the DropDownList is by setting the `rules` option of the `validate()` function as follows:

```
var validator = $("#form1").validate({
        rules: { DropDownList1: "required" },
messages: {
 DropDownList1: {required: "Please select a state"
                  }
},
               errorLabelContainer: $("#message")
        });
```

Note that when we define the rules as above, we don't need to specify the CSS class `required` in the DropDownList.

See also

Validation of ASP.NET ListBox Control

3
Working with GridView Control

In this chapter, we will cover:

- Highlighting rows/cells of a GridView on hover
- Removing GridView rows/cells on click
- Removing a GridView column on clicking the header
- Dragging and dropping GridView rows
- Changing cursor style for selective rows of a GridView
- Formatting a GridView and applying animation effect
- Retrieving the content of a GridView cell on click

Introduction

The GridView control is a powerful ASP.NET databound control introduced in .NET framework version 2.0. It has succeeded the DataGrid control in the earlier .NET framework 1.x versions.

As a new and improved DataGrid, it enables developers to connect to a datasource and present data in a grid format. Tasks such as updation, deletion, sorting, and paging can be accomplished with ease using this control.

Using the jQuery library, the GridView control can be manipulated on the client side without the need of a server-side page refresh. In this chapter, we will cover recipes that explore how jQuery can be used to enhance tasks with this ASP.NET control.

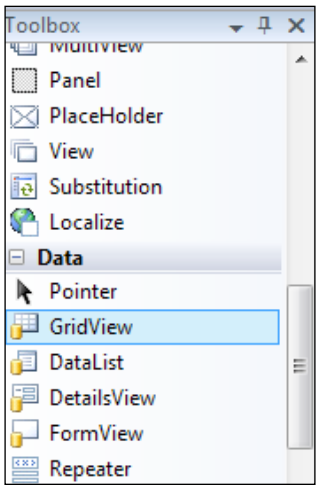
Getting started

1. To begin with, let's create a new ASP.NET website in Visual Studio and name it `Chapter3`.
2. Include the jQuery library in a script folder `js` in the project.

Creating and populating the GridView Control

Let's see how to go about creating a GridView control on an ASP.NET web form. We will base all the recipes in this chapter on this sample GridView:

1. Open the **ToolBox** window in Visual Studio.
2. Scroll down to the Data Controls section.
3. Select the **GridView** control as shown below. Drag-and-drop the control on the web form.



4. ASP.NET will automatically add the following code on the web form:

```
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

5. Add the System.Data namespace on the code-behind page:

   C#

```
using System.Data;
```

   VB

```
Imports System.Data
```

6. On the code-behind page, create a DataTable. Add sample columns to the DataTable and populate it with a few rows of data:

C#

```
public DataTable BuildDataTable()
{
  DataTable dt = new DataTable();
  dt.Columns.Add("BookID", typeof(string));
  dt.Columns.Add("Title",typeof(string));
  dt.Columns.Add("Author", typeof(string));
  dt.Columns.Add("Genre", typeof(string));
  dt.Columns.Add("RefNumber", typeof(string));

  dt.Rows.Add("34567","Alchemist","Paulo Coelho","Fiction",
"LNC1238.11");
  dt.Rows.Add("34568", "Jonathan Livingston Seagull", "Richard
Bach", "Fiction", "KL4532.67");
  dt.Rows.Add("34569", "Heart of Darkness", "Joseph Conrad",
"Classic", "CA5643.23");
  dt.Rows.Add("34570", "Oliver Twist", "Charles Dickens",
"Classic", "CA8933.55");
  return dt;
}
```

VB

```
Public Function BuildDataTable() As DataTable
  Dim dt As New DataTable()
  dt.Columns.Add("BookID", Type.GetType("System.String"))
  dt.Columns.Add("Title", Type.GetType("System.String"))
  dt.Columns.Add("Author", Type.GetType("System.String"))
  dt.Columns.Add("Genre", Type.GetType("System.String"))
  dt.Columns.Add("RefNumber", Type.GetType("System.String"))

  dt.Rows.Add("34567", "Alchemist", "Paulo Coelho", "Fiction",
"LNC1238.11")
  dt.Rows.Add("34568", "Jonathan Livingston Seagull", "Richard
Bach", "Fiction", "KL4532.67")
  dt.Rows.Add("34569", "Heart of Darkness", "Joseph Conrad",
"Classic", "CA5643.23")
  dt.Rows.Add("34570", "Oliver Twist", "Charles Dickens",
"Classic", "CA8933.55")
  Return dt
End Function
```

7. In the Page Load function, set the DataSource of the GridView to the DataTable. Bind the data to the GridView:

C#

```
protected void Page_Load(object sender, EventArgs e)
{
if (!Page.IsPostBack)
        {
GridView1.DataSource = BuildDataTable();
GridView1.DataBind();
}
}
```

VB

```
Sub Page_Load(ByVal Sender As System.Object, ByVal e As System.
EventArgs)
If Not Page.IsPostback Then
GridView1.DataSource = BuildDataTable()
GridView1.DataBind()
End If
End Sub
```

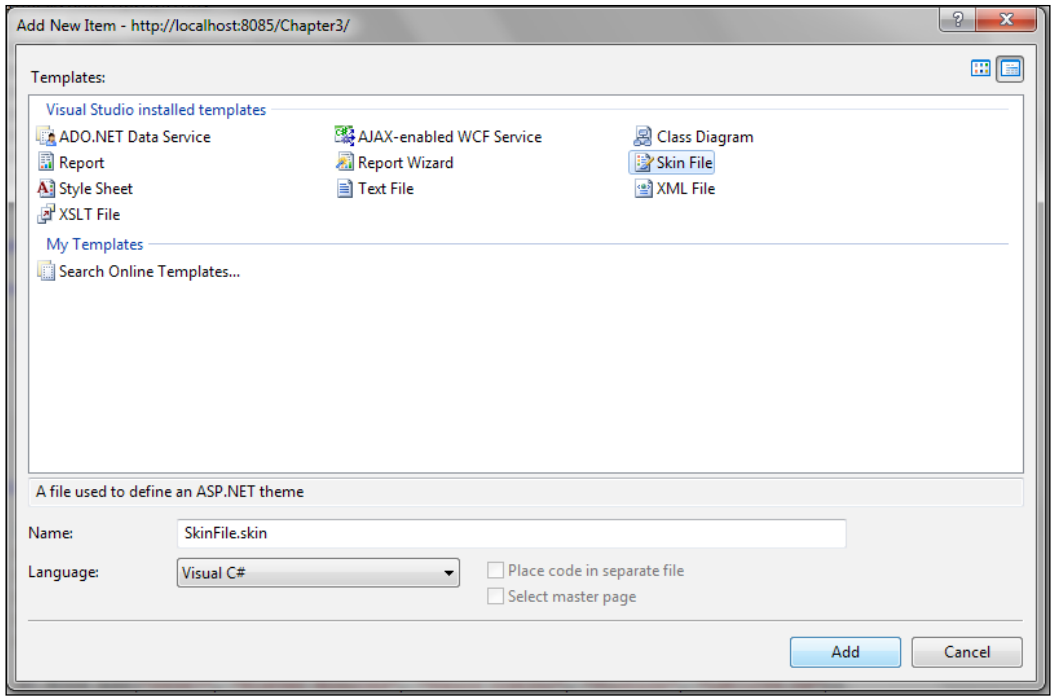
Applying skin file to GridView

For consistent styling of the GridView controls used in this chapter, we will make use of a standard skin file.

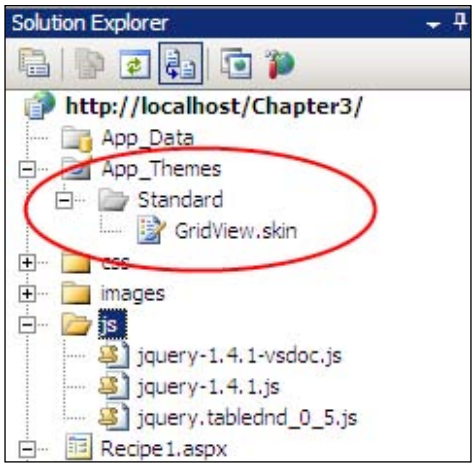
All theme files in an ASP.NET web application are saved in the `App_Themes` folder.

To apply the skin file to the GridView controls, follow these steps:

1. Right-click on the project and add a new folder **App_Themes**.
2. Now right-click on the **App_Themes** folder and click on **Add New Item**.
3. Select **Skin file** from the existing templates:



4. In ASP.NET, the skin files are named as `ControlName.skin`, so to style GridView controls we will name our file as `GridView.skin`.



5. Copy the following contents to the file and click **Save**:

```
<asp:GridView runat="server" SkinID="Professional" Font-
Name="Verdana" Font-Size="10pt" Cellpadding="4" HeaderStyle-
BackColor="#444444" HeaderStyle-ForeColor="White"/>
```

6. To apply a theme to a web page, we need to add the `StylesheetTheme` property to the page directive on the top of the ASPX page. Hence, update the web page as follows:

   C#

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.
aspx.cs" Inherits="_Default" StylesheetTheme="Standard" %>
```

   VB

```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="Default.
aspx.vb" Inherits="_Default" StylesheetTheme="Standard" %>
```

7. To apply the respective skin to the GridView control, set the `SkinID` of the control as follows:

```
<asp:GridView ID="GridView1" SkinID="Professional" runat="server">
</asp:GridView>
```

The GridView will now use the styles from the specified skin file. The web form will display the GridView data as shown in the following screenshot:

BookID	Title	Author	Genre	RefNumber
34567	Eleven Minutes	Paulo Coelho	Fiction	LNC1239.56
34568	One	Richard Bach	Fiction	KL4533.88
34569	Emma	Jane Austen	Classic	CA5644.45
34570	David Copperfield	Charles Dickens	Classic	CA5678.90

Now, with this information, let's move on to the recipes where we will see different manipulations of the ASP.NET GridView control using jQuery.

The recipes described in this chapter focus on the client side manipulations of the GridView control. The updation of the GridView data on the server side is beyond the scope of the chapter.

Highlighting rows/cells of a GridView on hover

In certain applications, for better readability there might be a need to highlight the rows/cells of a GridView on hover. We will see how to achieve this effect.

Getting ready

1. Create a new web form `Recipe1.aspx` in the current project.
2. Add a GridView control to the form and bind its data as explained earlier in this chapter.
3. The web form contents are as shown:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
        <tr>
            <td colspan="2">
                <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
                </asp:GridView>
            </td>
        </tr>
    </table>
    </fieldset>

</div>
    </form>
```

4. Add a css class `highlight` to the form so that we can change the background color of a row or cell:

```
.highlight
    {
       background-color:#9999FF;
    }
```

5. Change the cursor style for the GridView cells:

```
td { cursor:pointer;}
```

At runtime, ASP.NET renders a GridView control as a `<table>` element. Thus, each GridView row is rendered as a `<tr>` element while each GridView cell is rendered as a `<td>` element.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the hover event on each GridView row:

```
$("#<%=GridView1.ClientID%> tr").hover(
```

2. Define the handler for the `mouseenter` event:

```
function() {
                    $(this).addClass("highlight");
                },
```

3. Define the handler for the `mouseleave` event:

```
function() {
                    $(this).removeClass("highlight");
                });
```

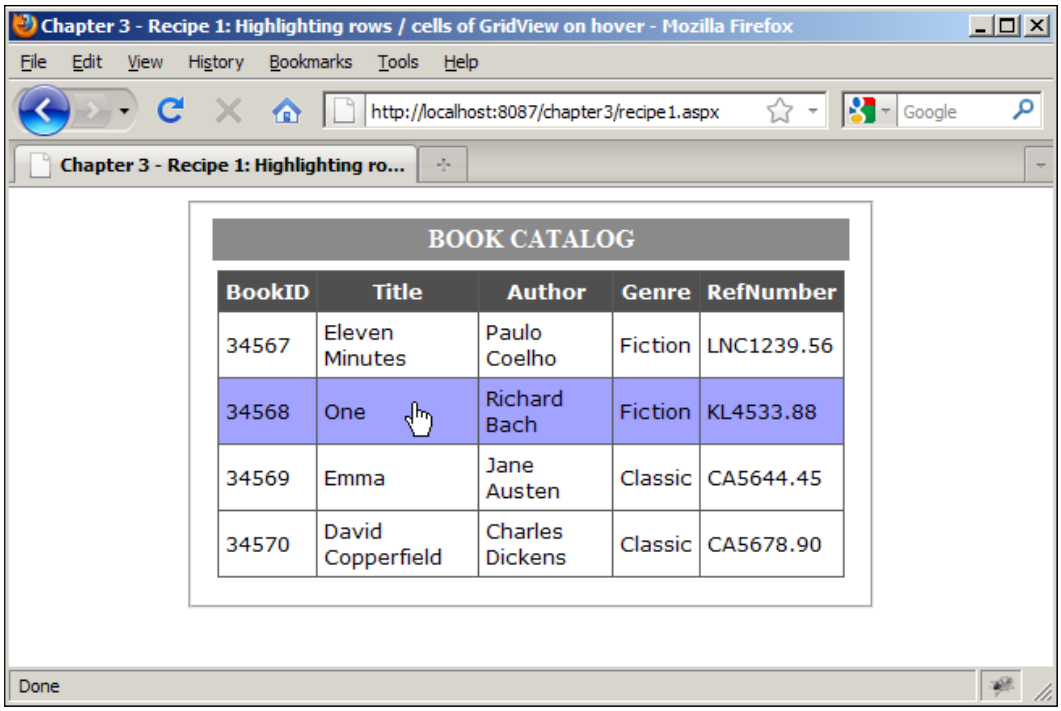
The `$(selector).hover(handlerIn, handlerOut)` is shorthand for `$(selector).mouseenter(handlerIn).mouseleave(handlerOut)`.

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
                $("#<%=GridView1.ClientID%> tr").hover(
                function() {
                    $(this).addClass("highlight");
                },
                function() {
                    $(this).removeClass("highlight");
                });
        });
    </script>
```

How it works...

Run the web form. Mouseover on any of the rows of the GridView. You will see that the respective row is highlighted as shown in the following screenshot:



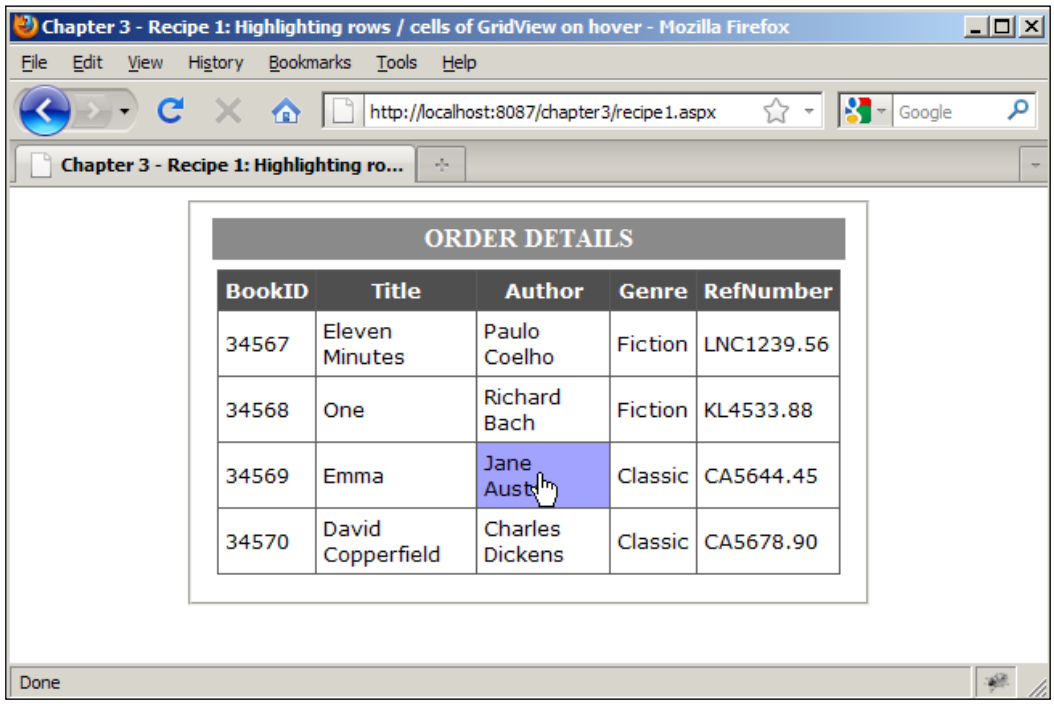
There's more...

The preceding jQuery solution can be modified slightly to achieve the same effect on a GridView cell instead of a GridView row.

By using the `$("#<%=GridView1.ClientID%> td")` selector, we can update the jQuery snippet to:

```
$("#<%=GridView1.ClientID%> td").hover(
                    function() {
                        $(this).addClass("highlight");
                    },
                    function() {
                        $(this).removeClass("highlight");
                    });
```

This will help us to achieve the following effect:



See also

Removing GridView rows/cells on click

Removing GridView rows/cells on click

In this recipe, let's see how to remove GridView rows or cells on a mouseclick. We will apply a simple animation effect to the target row or cell before removing it from the control.

Note that this recipe handles the client side update only. The server side GridView data refresh is not covered here.

Getting ready

1. Add a new web form `Recipe2.aspx` to the current project.
2. Add a GridView control to the form and bind its data as described earlier.

3. Thus, the web form contents are as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
       <tr><td colspan="2">
           <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
           </asp:GridView>
       </td></tr>
        </table>
        </fieldset>
        </div>
    </form>
```

4. Add a class `highlight` to update the background color of the target row or cell:

```
.highlight
    {
      background-color:#9999FF;
    }
```

5. Apply the following cursor style to the GridView cells:

```
td { cursor:pointer;}
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the click event on all rows of the GridView control except the header row as follows:

```
$("#<%=GridView1.ClientID%> tr").filter(":not(:has(table, th))").
click(function() {
```

When the `.filter()` selector is applied to the GridView rows, it constructs a new jQuery object that is a subset of the matching elements. Thus, `.filter(":not(:has(table, th))")` helps to filter out the header rows from the GridView control

2. Highlight the respective row:

```
$(this).addClass("highlight");
```

3. Apply a `fadeOut` animation effect with a timeout of 1000 ms. Remove the respective GridView row in the callback function of `fadeOut`:

```
$(this).fadeOut(1000, function() {
                    $(this).remove();
                });
            });
```

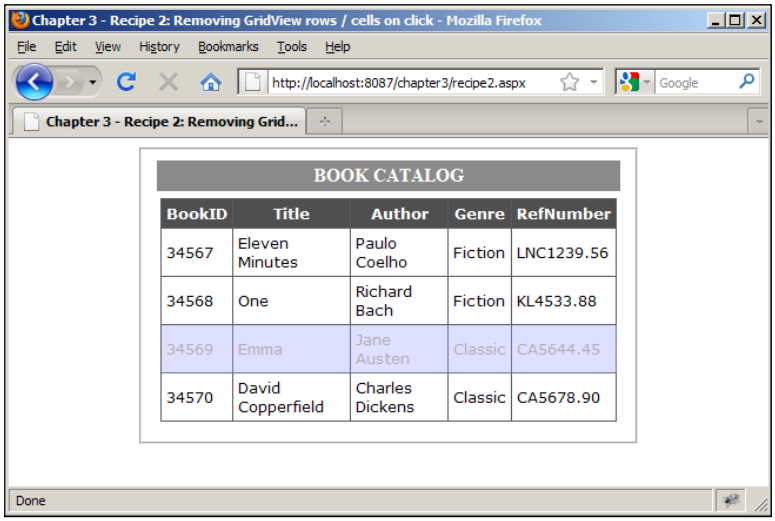
The jQuery selector `$(this)` retrieves the current element.

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#<%=GridView1.ClientID%> tr").filter(":not(:has(table,
th))").click(function() {
                $(this).addClass("highlight");
                $(this).fadeOut(1000, function() {
                    $(this).remove();
                });
            });
        });
    </script>
```

How it works...

Run the page. Click on any row of the GridView control. The row will slowly fade out and disappear from the control as shown in the following screenshot:



There's more...

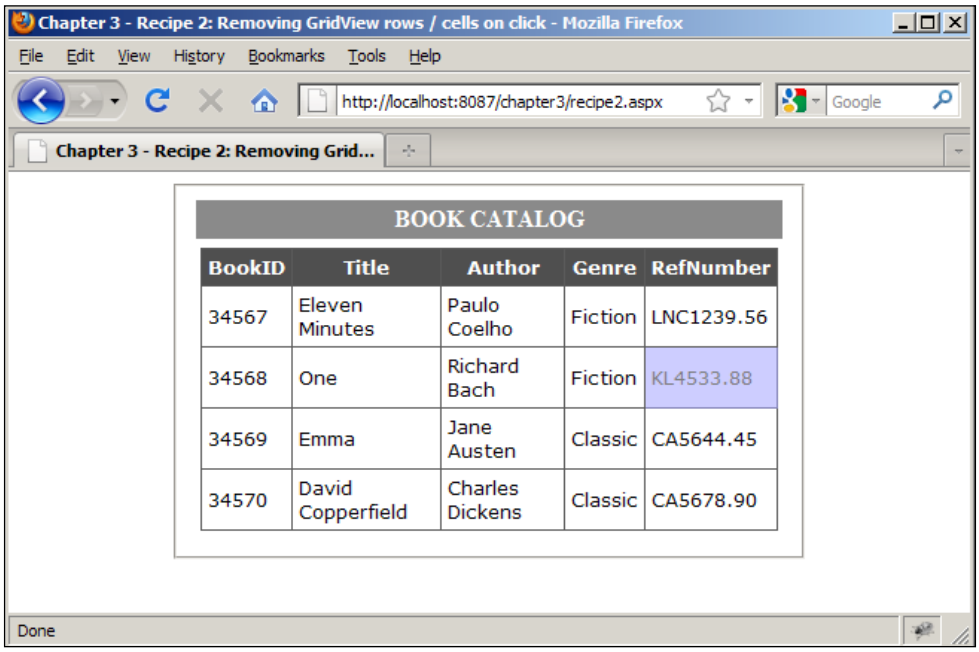
We can modify the preceding jQuery script slightly to apply the same effect on a GridView cell by using the jQuery selector `$("#<%=GridView1.ClientID%> td")`.

Thus, update the script as follows to remove a GridView cell:

```
$("#<%=GridView1.ClientID%> td").click(function() {
                $(this).addClass("highlight");
                $(this).fadeOut(1000, function() {
                    $(this).remove();
                });
            });
```

Note that in the preceding script, there is no need to explicitly filter the header since the header cells are rendered as `<th>` element.

Thus, on clicking any cell, it fades slowly and disappears as shown in the following screenshot:



See also

Removing a GridView column on clicking the header

Removing a GridView column on clicking the header

In this recipe, let's see how to delete a complete GridView column on clicking its header.

Getting ready

1. Add a new web form `Recipe3.aspx` to the current project.
2. Add a GridView control to the form and bind data to it as demonstrated earlier.
3. Thus, the web form contents are as follows:

```
<form id="form1" runat="server">
    <div align="center">
     <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
    <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
    <tr><td colspan="2">
        <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
        </asp:GridView>
    </td></tr>
    </table>
        </fieldset>
    </div>
    </form>
```

4. Add the following css style to the GridView header:

```
th { cursor:pointer;}
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the click event on the GridView header:

```
$("#<%=GridView1.ClientID %> th").click(function() {
```

2. Find the index of the respective column clicked by the web user:

```
var count = $(this).closest("th").prevAll("th").length;
```

The jQuery selector `.closest()` begins with the current element and travels up the DOM tree until it finds a matching element.

The jQuery selector `.prevAll()` searches all the predecessors of the current element in the DOM tree.

3. Define a callback function on each row of the GridView:

```
$(this).parents("#<%=GridView1.ClientID %>").find("tr").
each(function() {
```

4. Remove the GridView cell corresponding to the column index captured in the click event in the callback:

```
$(this).find("td:eq(" + count + ")").remove();
```

The jQuery selector `find("td:eq(" + count + ")")` retrieves the GridView cell with an index equal to `count`.

5. Remove the GridView header corresponding to the column index captured in the click event in the callback:

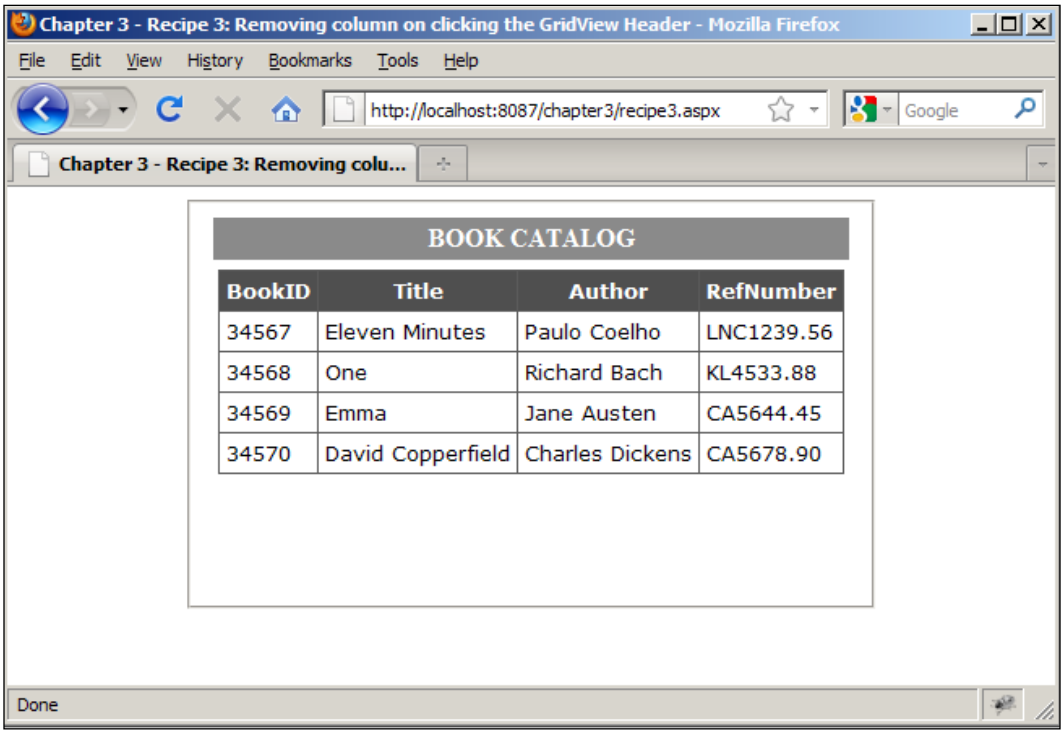
```
$(this).find("th:eq(" + count + ")").remove();
                });
            });
```

Thus, the complete jQuery solution is as follows:

```
<script type="text/javascript">
        $(document).ready(function() {
            $("#<%=GridView1.ClientID %> th").click(function() {
                var count = $(this).closest("th").prevAll("th").
length;
                $(this).parents("#<%=GridView1.ClientID %>").
find("tr").each(function() {
                    $(this).find("td:eq(" + count + ")").remove();
                    $(this).find("th:eq(" + count + ")").remove();
                });
            });
        });
    </script>
```

How it works...

Run the web page. Click on any GridView header. You will notice that the target column is deleted from the control as shown in the following screenshot:



See also

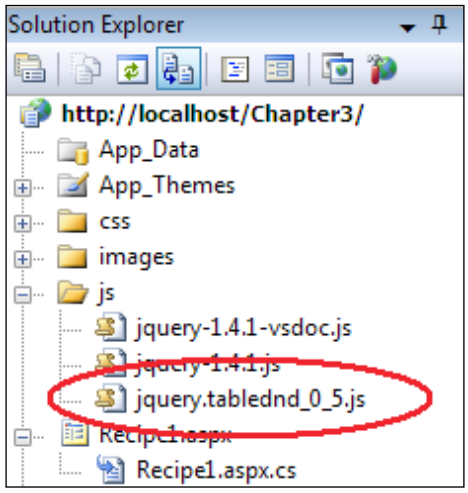
Formatting a GridView and applying animation effect

Dragging and dropping GridView rows

jQuery comes with a number of handy plugins to enhance the functionality of web controls. In user interactive applications, where there is a need to shuffle the rows of a GridView, we will see how we can apply a jQuery plugin to accomplish the task on the client without the need of a server-side refresh.

Getting ready

1. Add a new form `Recipe4.aspx` to the current project.
2. Download the jQuery **Drag and Drop** plugin available from `http://www.isocra.com/2008/02/table-drag-and-drop-jquery-plugin/`.
3. Save the plugin file to the script folder `js` in the project as shown in the following screenshot:



4. Include the jQuery library as well as the plugin on the page:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
    <script type="text/javascript" src="js/jquery.tablednd_0_5.
js"></script>
```

5. Add a GridView control to the form. Populate data to the control as described earlier in the chapter. Thus, the web form contents are as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
    <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
    <tr><td colspan="2">
        <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
        </asp:GridView>
    </td></tr>
    </table>
    </fieldset>
    </div>
    </form>
```

How to do it...

In the `document.ready()` function of the jQuery script block, call the `tableDnD()` function on the GridView control:

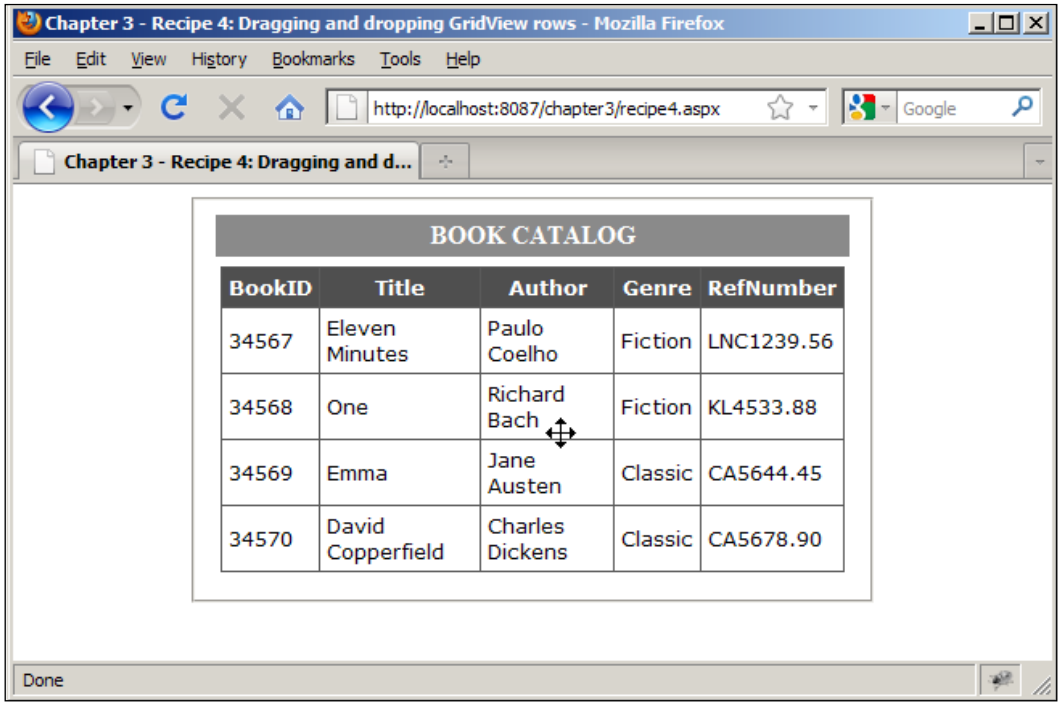
```
$("#<%=GridView1.ClientID %>").tableDnD();
```

Thus, the jQuery solution applied to the page is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        $("#<%=GridView1.ClientID %>").tableDnD();
        });
    </script>
```

How it works...

Run the web page. Mouseover on any row of the GridView. You will notice that the cursor style changes within the control. You can hold down the cursor and drag the rows to reshuffle the row order as required.



See also

Highlighting rows/cells of a GridView on hover

Changing cursor style for selective rows of a GridView

With the help of jQuery, the default cursor style can be changed based on the GridView content. In this recipe, we will assign different cursor styles to odd and even rows of a GridView.

Getting ready

1. Add a new web form `Recipe5.aspx` to the current project.
2. Add a GridView control to the form. Populate data to the control as described earlier in this chapter. Thus, the form contents will be as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
       <tr><td colspan="2">
           <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
           </asp:GridView>
       </td></tr>
        </table>
        </fieldset>
        </div>
    </form>
```

We will now use jQuery and css to add different cursor styles to odd and even rows of the GridView.

How to do it...

1. In the `document.ready()` function of the jQuery script block, retrieve the even rows using the `.filter(":even")` selector. Define the mouseover event for this set of rows:

```
$("#<%=GridView1.ClientID%> tr").filter(":even").bind("mouseover",
function() {
```

2. Add cursor style `pointer` to the even rows:

```
$(this).css("cursor","pointer");
            });
```

3. Retrieve the odd rows using the `.filter(":odd")` selector. Define the mouseover event for this set of rows:

```
$("#<%=GridView1.ClientID%> tr").filter(":odd").bind("mouseover",
function() {
```

4. Add cursor style `wait` to the odd rows:

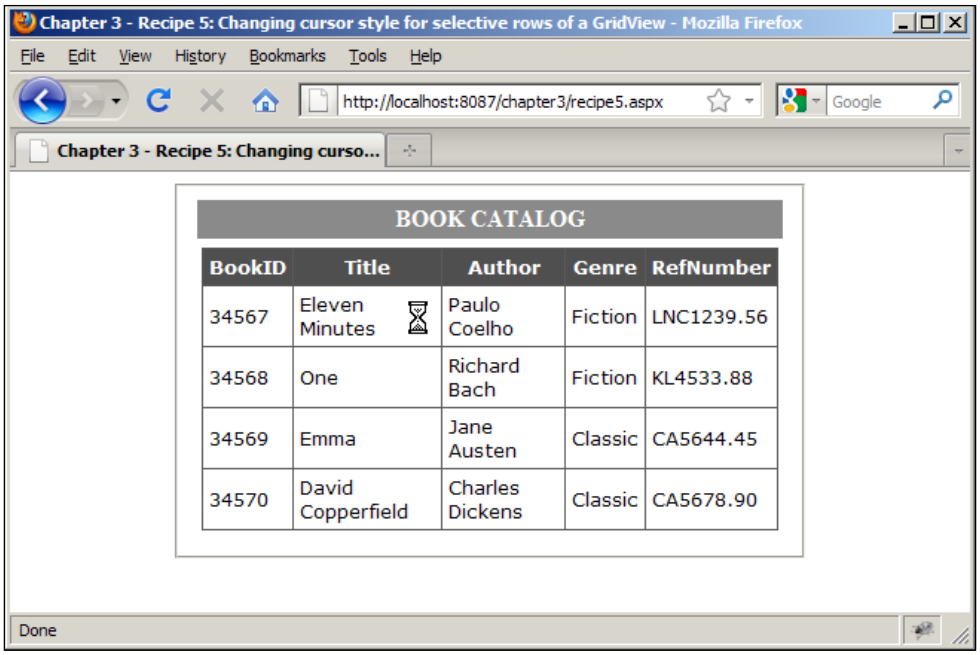
```
$(this).css("cursor", "wait");
            });
```

Thus, the complete jQuery solution is as follows:

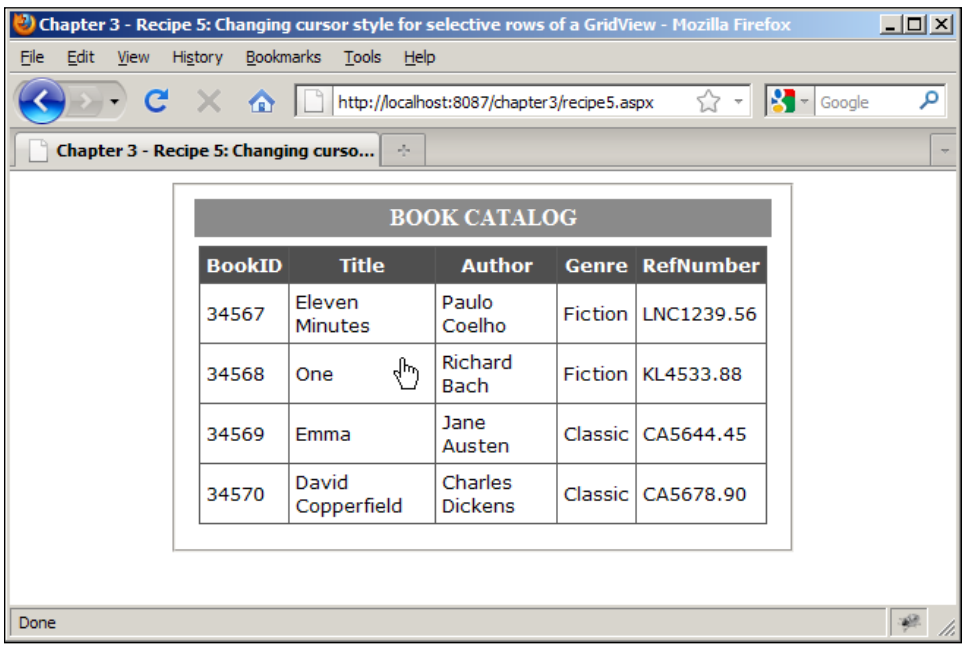
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#<%=GridView1.ClientID%> tr").filter(":even").
bind("mouseover", function() {
                $(this).css("cursor","pointer");
            });
            $("#<%=GridView1.ClientID%> tr").filter(":odd").
bind("mouseover", function() {
                $(this).css("cursor", "wait");
            });
        });
    </script>
```

How it works...

Run the web page. Now, mouseover on any odd numbered row. You will observe the cursor style changes as displayed in the screen in the following screenshot:



Now, mouseover on any even numbered row. The cursor style changes as shown in the following screenshot:



See also

Formatting a GridView and applying animation effect

Formatting a GridView and applying animation effect

In this recipe, we will apply a background color to the alternating rows of a GridView using jQuery. We will also apply a simple animation effect while loading the DOM.

Getting ready

1. Create a new web form `Recipe6.aspx` in the current project.
2. Add a GridView control to the form and populate with data as described earlier in this chapter. Thus, the form contents are as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
       <tr><td colspan="2">
           <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
           </asp:GridView>
       </td></tr>
        </table>
    </fieldset>
    </div>
    </form>
```

We will now apply a simple animation effect on the GridView. We will also use jQuery to set the css `background-color` property for applying background color to the alternating rows of the GridView.

How to do it...

1. In the `document.ready()` function of the jQuery script block, apply the `slideUp()` animation function to the GridView. Subsequently, chain the `slideDown()` function as follows:

```
$("#<%=GridView1.ClientID%>").slideUp(2500).slideDown(2500);
```

In the jQuery animation functions `.slideUp()` and `.slideDown()`, the duration of the animation is specified in milliseconds.

2. Select the odd rows using the jQuery `.filter(":odd")` selector and update the css `background-color` property as follows:

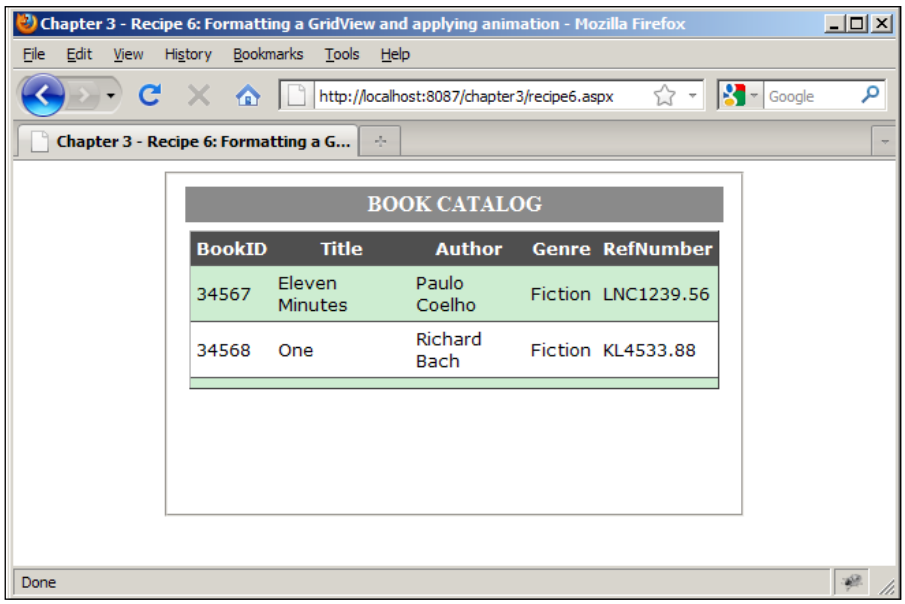
```
$("#<%=GridView1.ClientID%> tr").filter(":odd").css("background-color", "#c8ebcc");
```

Thus, the complete jQuery solution is as follows:

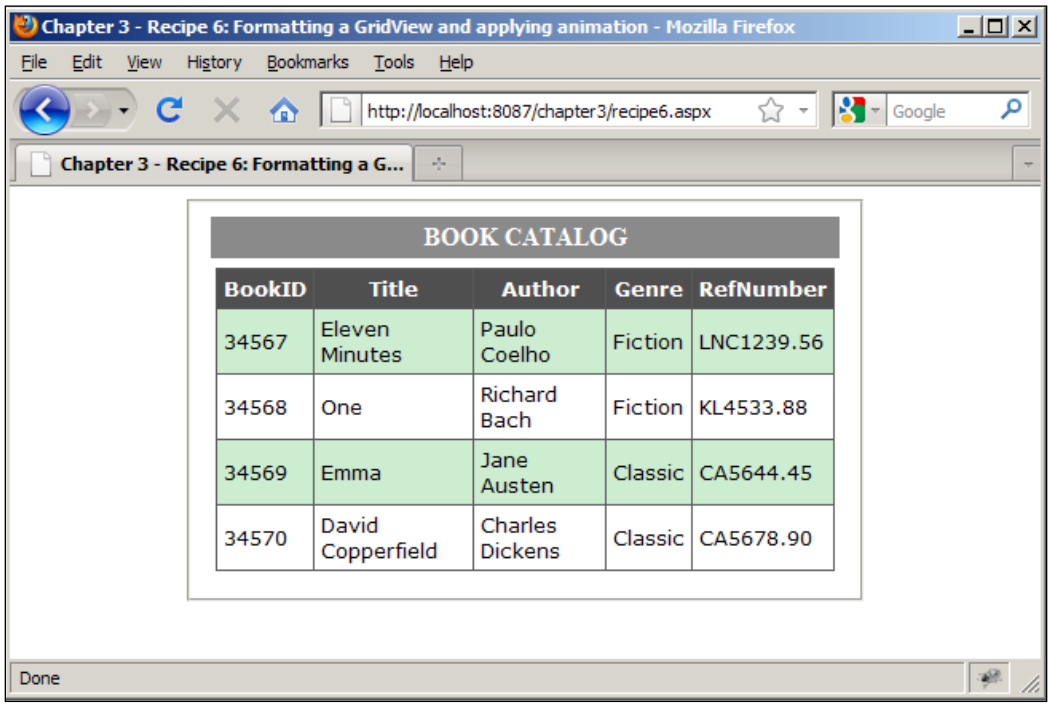
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#<%=GridView1.ClientID%>").slideUp(2500).
slideDown(2500);
            $("#<%=GridView1.ClientID%> tr").filter(":odd").
css("background-color", "#c8ebcc");
        });
    </script>
```

How it works...

Run the web page. On load, the GridView control will slide up in 2,500 milliseconds. Subsequently it will slide down in 2,500 milliseconds. The animation effect is captured in the following screenshot:



The required background color is applied to the alternating rows as shown in the following screenshot:



See also

Changing cursor style for selective rows of a GridView

Retrieving the content of a GridView cell on click

Often in web applications using GridView controls there is a need to retrieve the contents of the cell clicked by the web user. In this recipe, we will look at one such example.

Getting ready

1. Add a new web form `Recipe7.aspx` to the current project.
2. Add a GridView control to the form and populate with sample data as explained earlier in this chapter.

3. Add a div area `message` where we will display the contents of the clicked cell:

```
<div id="message"></div>
```

4. Thus, the form contents are as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:230px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">BOOK CATALOG</td></tr>
       <tr><td colspan="2">
           <asp:GridView ID="GridView1" SkinID="Professional"
runat="server">
           </asp:GridView>
       </td></tr>
        </table>
    </fieldset>
    <br />
    <div id="message"></div>
    </div>
    </form>
```

5. Add a css class to highlight the clicked cell:

```
.highlight
    {
      background-color:#9999FF;
    }
```

6. Add css style to change the default cursor style on all cells of the GridView:

```
td { cursor:pointer;}
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the click event on all rows of the GridView control except the header row:

```
$("#<%=GridView1.ClientID%> tr").filter(":not(:has(table, th))").
click(function(e) {
```

When the `.filter()` selector is applied to the GridView rows, it constructs a new jQuery object that is a subset of the matching elements. In this case, it helps to filter out the header rows from the selection.

2. Find the target cell that was clicked by the web user:

```
var $cell = $(e.target).closest("td");
```

`e.target` retrieves the DOM element that initiated the event. The jQuery selector `.closest()` travels up the DOM tree starting from the current element and retrieves the first matching element.

3. If any cells were highlighted previously, remove the highlighting from the respective cells:

```
$("#<%=GridView1.ClientID%> td").removeClass("highlight");
```

4. Highlight the target cell - the one clicked by the web user:

```
$cell.addClass("highlight");
```

5. Retrieve the contents of the target cell and display in the div area:

```
$("#message").text('You have selected: ' + $cell.text());
            });
```

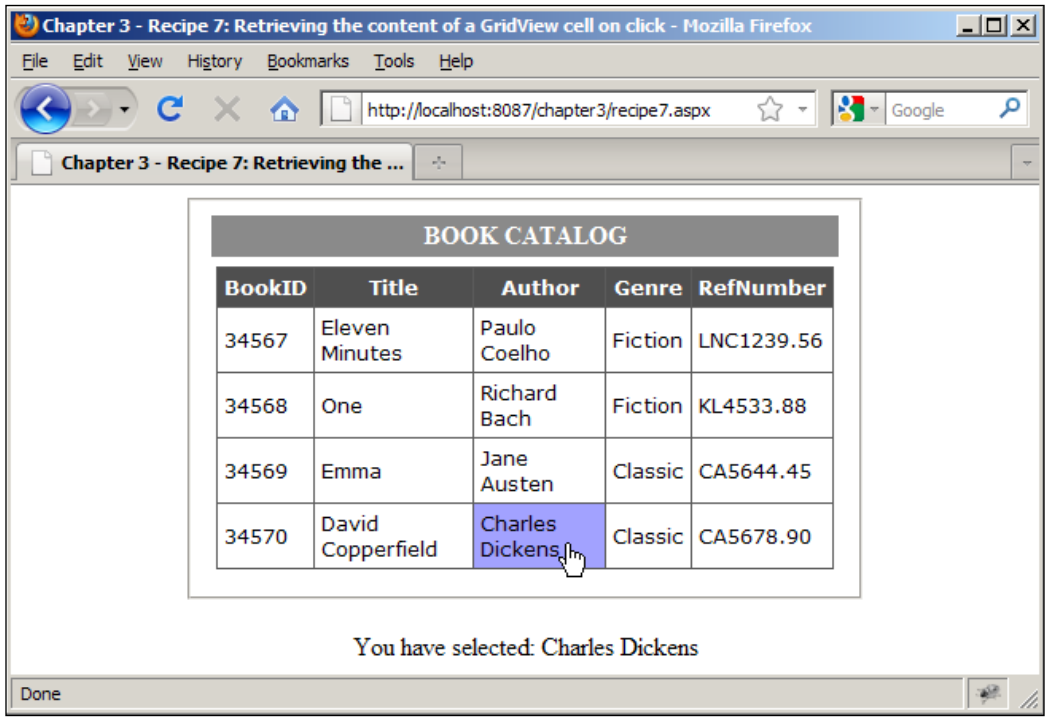
`$cell.text()` retrieves the contents of the target cell.

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#<%=GridView1.ClientID%> tr").filter(":not(:has(table,
th))").click(function(e) {
                var $cell = $(e.target).closest("td");
                $("#<%=GridView1.ClientID%> td").
removeClass("highlight");
                $cell.addClass("highlight");
                $("#message").text('You have selected: ' + $cell.
text());
            });
        });
    </script>
```

How it works...

Run the web page. Click on any cell of the GridView. You will notice that the cell is highlighted and the corresponding text is displayed below the GridView as shown in the following screenshot:



See also

Formatting a GridView and applying animation effect

4
Working with Image Control

In this chapter, we will cover:

- Adding/removing hyperlinks on images
- Displaying image captions
- Changing image opacity on mouseover
- Viewing enlarged images on mouseover on thumbnail
- Swapping images on a page
- Cropping images on a page
- Creating an image gallery viewer
- Zooming images on mouseover

Introduction

In ASP.NET, the Image control is an indespensible tool for creating rich graphic-intensive pages. As compared with HTML's image element, ASP.NET's Image control runs on the server and provides a rich collection of properties and methods for the use of developers.

A basic image control on an ASPX page is as follows:

```
<asp:Image ID="Image1" ImageUrl="~/images/Waterfall.JPG"
AlternateText="Working with Image Control" ImageAlign="Middle"
runat="server" />
```

Some of the useful properties of the control are:

- `ImageUrl`: This is used to set the path of the image to be displayed.
- `AlternateText`: This sets the alternate text to display on hover.
- `ImageAlign`: This sets the alignment of the image. i.e. Left, Right, Top, Bottom, etc.
- `CssClass`: Sets the css class of the control.
- `SkinID`: Helps to apply a Skin file from a theme.
- `DescriptionUrl`: The URL of the file that provides an additional description of the image to supplement the `AlternateText` property. The default value is an empty string.

jQuery, being a client side library, can be used with ASP.NET's Image control, thus enabling manipulation of the control on the client without the need of a server-side page refresh.

In addition to the core jQuery library, there are numerous plugins that can be interfaced with the Image control for cropping, building photo galleries, and a number of other interesting applications. In this chapter, we will touch on some useful recipes based on interfacing jQuery with the Image control.

Getting started

1. To begin with, let's create a new ASP.NET website in Visual Studio and name it `Chapter4`.
2. Save the jQuery library in a script folder `js` in the project.
3. Create a sub-folder `images` in the project and copy a couple of sample images.
4. To enable jQuery on any web form, drag-and-drop the jQuery library to add the following on the page:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
```

Now, let's move on to the recipes where we will see different ways of interfacing the ASP.NET Image control and jQuery.

Adding/removing hyperlinks on images

In this recipe, we will attach/remove hyperlinks on images at runtime depending on the events triggered on a web page.

Getting ready

1. Create a new web form `Recipe1.aspx` in the project.

2. Add an Image control to the page and link it to an image from the `images` folder:

```
<asp:Image ID="Image1" ImageUrl="~/images/Image1.JPG" Width="400"
Height="250" runat="server" />
```

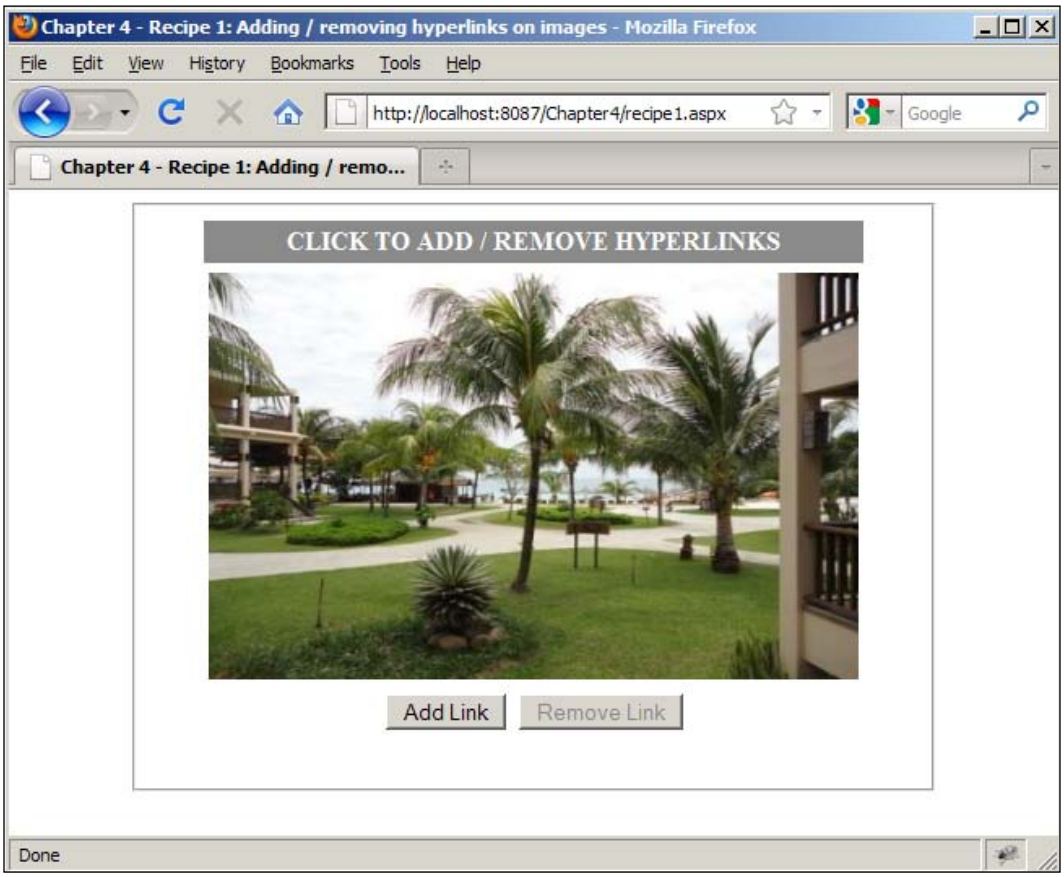
3. Add a Button control for adding a hyperlink to the Image control dynamically. Add another Button control for removing the hyperlink. Disable this button initially:

```
<asp:Button ID="btnAdd" runat="server" Text="Add Link" /> 
<asp:Button ID="btnRemove" runat="server" Text="Remove Link"
Enabled="false" />
```

Thus, the complete ASPX markup of the web form will be as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:470px;height:340px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">CLICK TO ADD / REMOVE
HYPERLINKS</td></tr>
        <tr>
            <td colspan="2">
                <asp:Image ID="Image1" ImageUrl="~/images/Image1.
JPG" Width="400" Height="250" runat="server" />
            </td>
        </tr>
        <tr><td colspan="2" align="center">
            <asp:Button ID="btnAdd" runat="server" Text="Add Link"
/> 
            <asp:Button ID="btnRemove" runat="server" Text="Remove
Link" Enabled="false" /></td></tr>
    </table>
    </fieldset>

</div>
    </form>
```

On load, the page will dispay the image as shown in the following screenshot:



How to do it...

1. In the `document.ready()` function of the jQuery script block, define the callback function for the `click` event of the **Add Link** button:

   ```
   $("#btnAdd").click(function(e) {
   ```

2. Wrap a hyperlink element around the image control:

   ```
   $("#Image1").wrap('<a href="http://www.packtpub.com" target="_
   blank"></a>');
   ```

The jQuery `.wrap()` method appends a HTML structure around the matched element.

3. Prevent default form submission on a button click:
   ```
   e.preventDefault();
   ```
4. Enable the **Remove Link** button:
   ```
   $("#btnRemove").removeAttr("disabled");
   ```
5. Disable the **Add Link** button:
   ```
   $("#btnAdd").attr("disabled", "disabled");
   });
   ```
6. Now, define the callback function on the **click** event of the **Remove Link** button:
   ```
   $("#btnRemove").click(function(e) {
   ```
7. Remove the parent element of the Image control i.e. the hyperlink element in this case:
   ```
   $("#Image1").unwrap();
   ```

The jQuery `.unwrap()` method removes the parent element of the matched element. It leaves the matched element in place.

8. Prevent default form submission on a button click:
   ```
   e.preventDefault();
   ```
9. Enable the **Add Link** button:
   ```
   $("#btnAdd").removeAttr("disabled");
   ```
10. Disable the **Remove Link** button:
    ```
    $("#btnRemove").attr("disabled", "disabled");
                });
    ```

Thus, the complete jQuery solution is as follows:

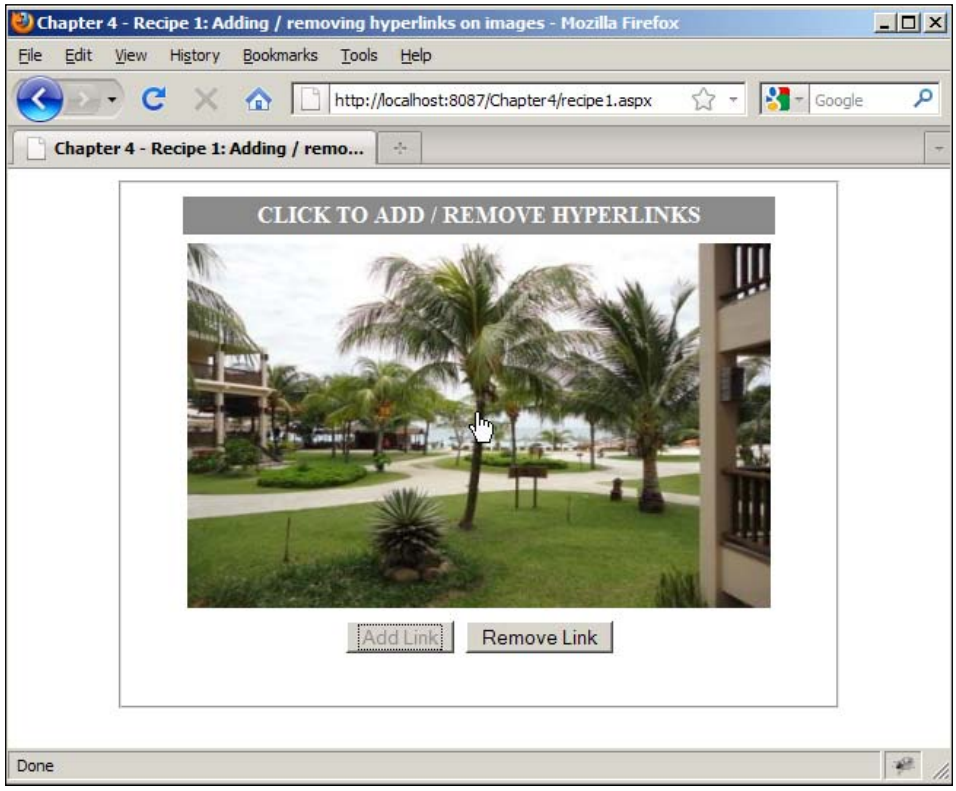
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            //ADDING HYPERLINK TO AN IMAGE
            $("#btnAdd").click(function(e) {
                $("#Image1").wrap('<a href="http://www.packtpub.com"
target="_blank"></a>');
                e.preventDefault();
                $("#btnRemove").removeAttr("disabled");
                $("#btnAdd").attr("disabled", "disabled");

            });
            //REMOVING HYPERLINK ON THE IMAGE
```

```
            $("#btnRemove").click(function(e) {
                $("#Image1").unwrap();
                e.preventDefault();
                $("#btnAdd").removeAttr("disabled");
                $("#btnRemove").attr("disabled", "disabled");
            });
        });
    </script>
```

How it works...

Run the web form. Mouseover on the image. Initially, there is no hyperlink on the image.

Now click the **Add Link** button. You will notice that the image is now hyperlinked as shown:



On clicking the **Remove Link** button, the hyperlink disappears.

See also

Displaying image captions.

Displaying image captions

In this recipe, we will display the image captions on mouseovers. We will make use of the `mouseenter` and `mouseleave` events to achieve the same.

Getting ready

1. Add a new web form `Recipe2.aspx` to the current project.
2. Define a css class for the images as follows:

```
.image
    {
      position:relative;
      top: 0;
      left: 0;
      width: 400px;
      height: 250px;
    }
```

3. Define a css class for the respective image captions as follows:

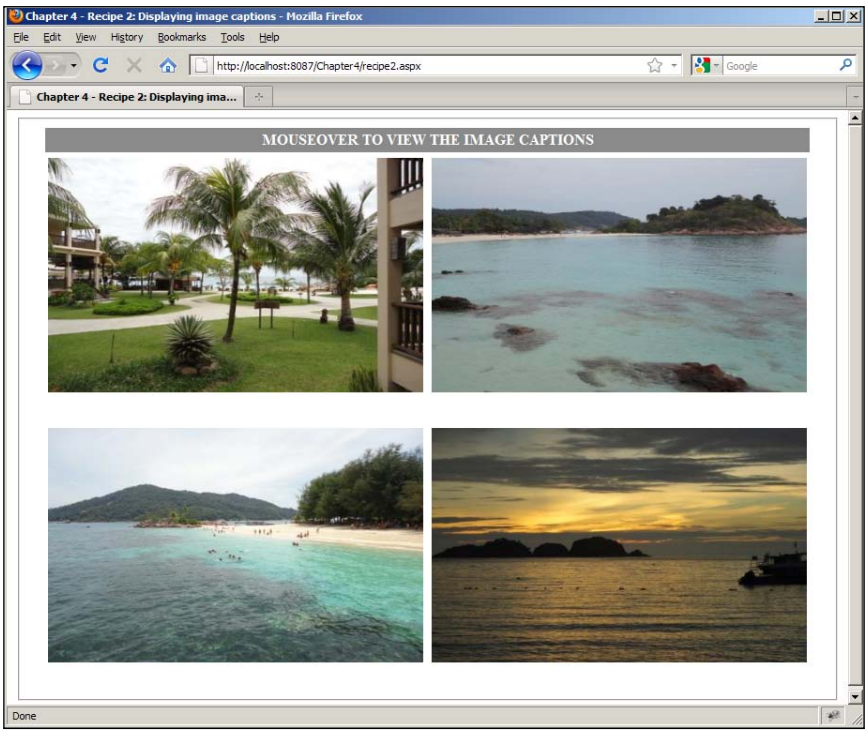
```
.caption
    {
      font-weight:bold;
      font-family:Arial;
      position:absolute;
    }
```

4. Add a couple of image controls on the form and set the css style to `image`. Define ToolTips for the images. We will use the ToolTip text as the image caption. Thus, the ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:850px;height:600px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">MOUSEOVER TO VIEW THE
IMAGE CAPTIONS</td></tr>
        <tr>
            <td>
                <asp:Image ID="Image1" CssClass="image"
ImageUrl="~/images/Image1.JPG" ToolTip="Resort" runat="server" />
            </td>
            <td>
```

```
                <asp:Image ID="Image2" CssClass="image"
ImageUrl="~/images/Image2.JPG" ToolTip="Beach" runat="server" />
            </td>
        </tr>
        <tr><td colspan="2"> </td></tr>
        <tr>
            <td>
                <asp:Image ID="Image3" CssClass="image"
ImageUrl="~/images/Image3.JPG" ToolTip="Marine Park"
runat="server" />
            </td>
            <td>
                <asp:Image ID="Image4" CssClass="image"
ImageUrl="~/images/Image4.JPG" ToolTip="Sunrise" runat="server" />
            </td>
        </tr>
    </table>
    </fieldset>

</div>
    </form>
```

5. Thus, on load, the page displays the image controls as shown in the following screenshot:



How to do it...

1. In the `document.ready()` function of the jQuery script block, define the callback function for the `mouseenter` event on any Image control on the form:

```
$("#form1 img").bind("mouseenter", function() {
```

The jQuery `.bind()` method helps to attach event handlers to the matched element for the specified event types.

2. Change the cursor style to `pointer` when the mouse is over the image:

```
$(this).css("cursor", "pointer");
```

3. Retrieve the ToolTip text using the `title` property. We will use this text as the caption:

```
var caption = $(this).attr("title");
```

At runtime, the ASP.NET Image control is rendered as `<img>` element and the `ToolTip` property is rendered as `title` text.

4. Add a div area containing the caption after the Image control element in the DOM tree. Add the css class `caption` to this div area:

```
$(this).after("<div class='caption'>" + caption + "</div>");

});
```

The jQuery `.after()` method inserts the specified content after the matched elements.

5. Define the callback function for the `mouseleave` event:

```
$("#form1 img").bind("mouseleave", function() {
```

6. Remove the div element that has the css class `caption` attached to it:

```
$('.caption').remove();
            });
```

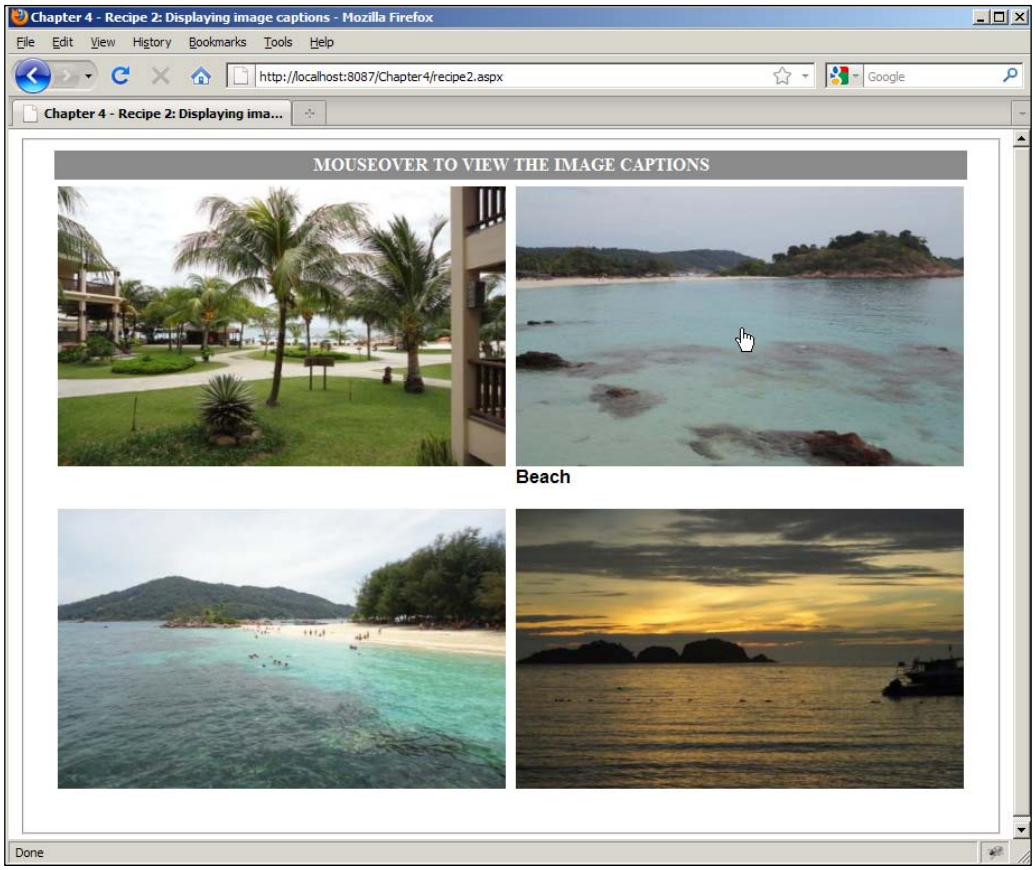
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#form1 img").bind("mouseenter", function() {
                $(this).css("cursor", "pointer");
```

```
                var caption = $(this).attr("title");
                $(this).after("<div class='caption'>" + caption + "</
div>");
            });
            $("#form1 img").bind("mouseleave", function() {
                $('.caption').remove();
            });
        });
    </script>
```

How it works...

Run the web form. Mouseover on any image on the form. You will notice that the respective image caption is displayed below the image, as captured in the following screenshot:



On mouseout, the image caption disappears.

See also

Adding/removing hyperlinks on images

Changing image opacity on mouseover

Opacity is the degree of transparency of an image. In this recipe, we will apply different opacity levels to an image.

Getting ready

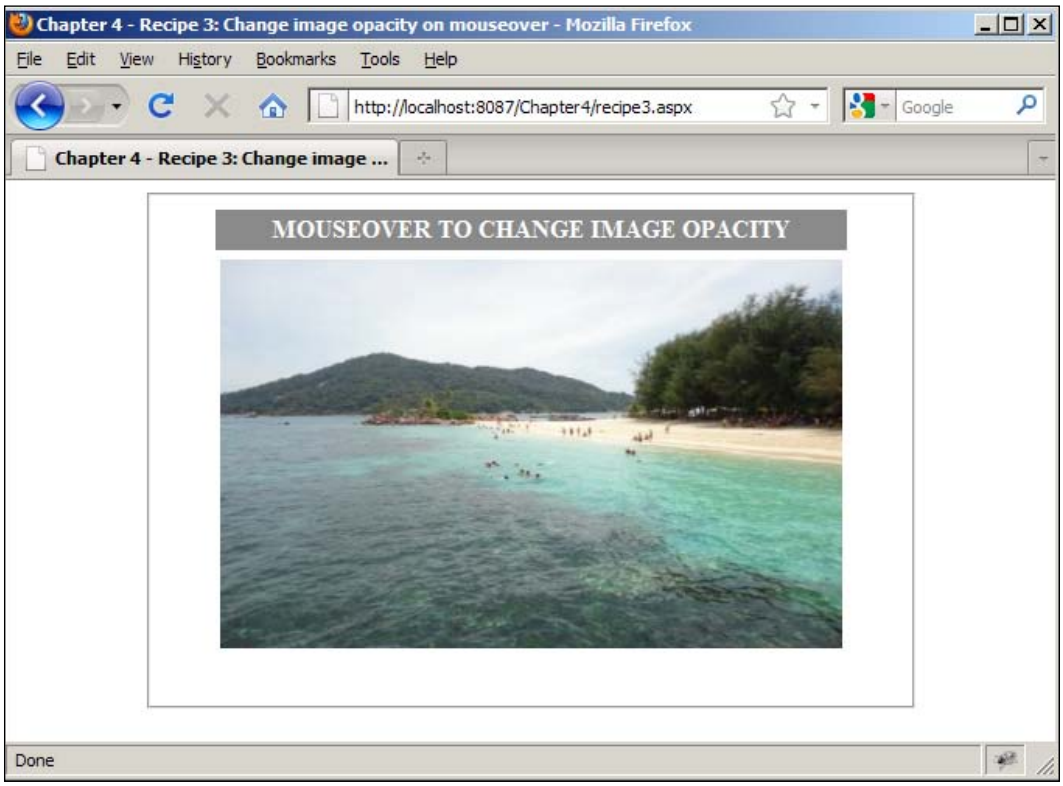
1. Add a new web form `Recipe3.aspx` to the current project.
2. Create a css style for the Image control as follows:

```
.image
    {
      position:relative;
      width: 400px;
      height: 250px;
    }
```

3. Now add an Image control to the form as below:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:470px;height:310px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">MOUSEOVER TO CHANGE
IMAGE OPACITY</td></tr>
        <tr>
            <td colspan="2">
                <asp:Image ID="Image1" ImageUrl="~/images/Image3.
jpg" CssClass="image" runat="server" />
            </td>
        </tr>
    </table>
    </fieldset>

</div>
    </form>
```

4. The page displays the image as follows:



We will now animate the opacity of the image on hover. We will define different opacity levels when the mouse enters and leaves the image.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the hover event on the Image control:

   ```
   $("#Image1").hover(
   ```

The `.hover()` method binds handlers for `mouseenter` and `mouseleave` events.

2. Define the handler when the mouse enters the image. Animate the opacity as follows:

```
function() {
                $(this).animate({ opacity: 0.6 }, 200);
            },
```

The jQuery `.animate()` method creates animation effects on the specified css property. It takes in the duration of the animation (in milliseconds) as a parameter. Thus, the above snippet animates the image opacity to 0.6 in 200 ms.

3. Define the handler when the mouse leaves the image. Animate the opacity as follows:

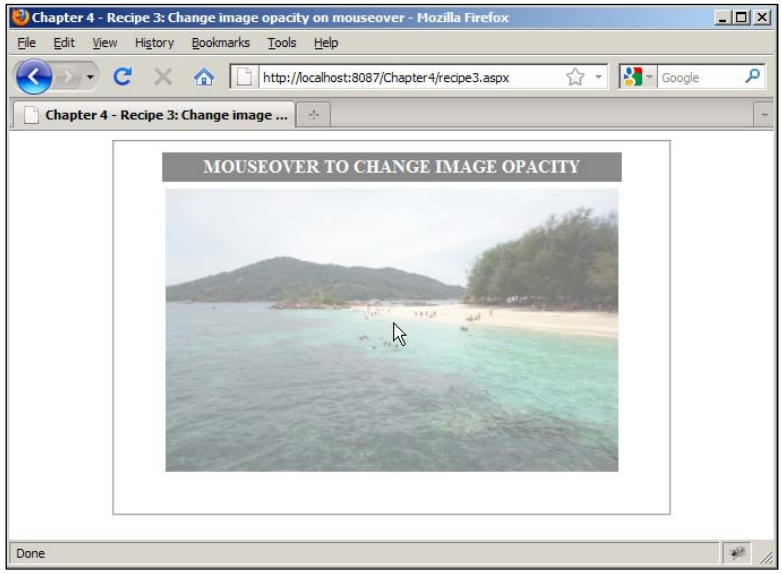
```
function() {
                $(this).animate({ opacity: 0.3 }, 200);
            });
```

Thus, the complete jQuery solution is as follows:

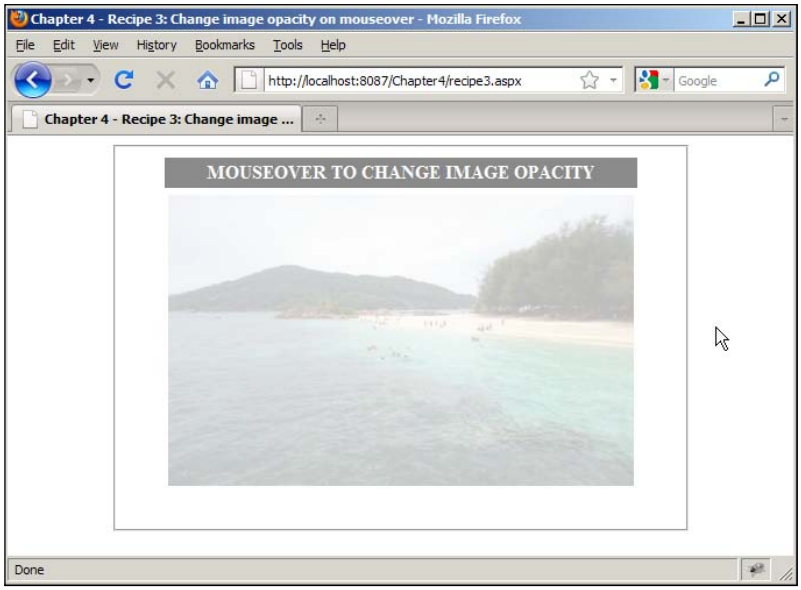
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        $("#Image1").hover(
            function() {
                $(this).animate({ opacity: 0.6 }, 200);
            },
            function() {
                $(this).animate({ opacity: 0.3 }, 200);
            });
        });
    </script>
```

How it works...

Run the page. Now mouseover on the image. You will see that the opacity level changes as follows:



Now mouseout on the image. The opacity updates again as shown:



See also

Zooming images on mouseover

Viewing enlarged images on mouseover on thumbnail

In this recipe, let's see how to view enlarged images on mouseover on the thumbnails.

Getting ready

1. Add a new web form `Recipe4.aspx` to the current project.
2. Add a couple of Image controls to the page. Define a css class `thumbnail` as shown next and set the CssClass of the Image controls to this class:

```
.thumbnail
    {
      position:relative;
      width:100px;
      height:68px;

    }
```

3. Create a div area below the Image controls where we will display the enlarged image:

```
<div id="imgContainer" ></div>
```

4. Create a css style for the enlarged image as follows:

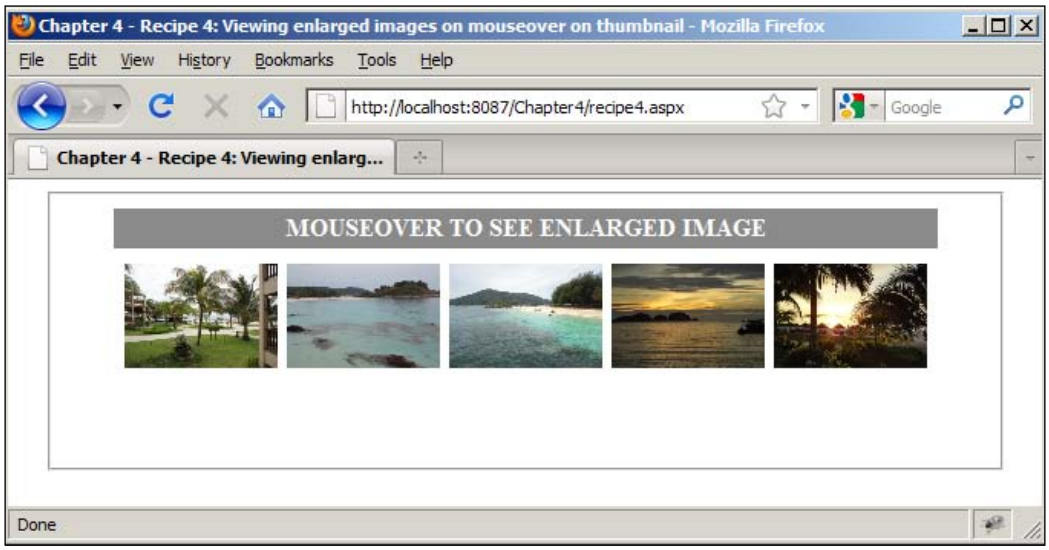
```
.image
    {
      position:relative;
      width:400px;
      height:250px;
    }
```

Thus, the complete ASPX markup for the form is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:600px;height:390px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">MOUSEOVER TO SEE ENLARGED
IMAGE</td></tr>
        <tr>
            <td colspan="2">
```

```
            <table cellspacing="2" cellpadding="2">
            <tr><td>
                <asp:Image ID="Image1" CssClass="thumbnail"
ImageUrl="~/images/Image1.jpg" runat="server" /> </td>
                <td><asp:Image ID="Image2" CssClass="thumbnail"
ImageUrl="~/images/Image2.jpg" runat="server" /> </td>
               <td> <asp:Image ID="Image3" CssClass="thumbnail"
ImageUrl="~/images/Image3.jpg" runat="server" /> </td>
                <td><asp:Image ID="Image4" CssClass="thumbnail"
ImageUrl="~/images/Image4.jpg" runat="server" /> </td>
                <td><asp:Image ID="Image5" CssClass="thumbnail"
ImageUrl="~/images/Image5.jpg" runat="server" />
              </td></tr>
               </table>
            </td>
        </tr>
        <tr><td align="center">
        <div id="imgContainer" ></div>
        </td></tr>
    </table>
    </fieldset>

</div>
    </form>
```

Thus, initially on page load, the web form displays the thumbnails as shown in the following screen:



We will now display the enlarged image in the div area when there is a mouseover on any thumbnail. On mouseout, the image disappears from the div area.

How to do it...

1. In the `document.ready()` function of the jQuery script block, use the css selector `$(".thumbnail")` to select all thumbnails. Define the `hover` event on this collection:

   ```
   $(".thumbnail").hover(
   ```

2. Define the handler for the `mouseenter` event. Use css to update the opacity of all thumbnails to 0.5:

   ```
   function() {
                   $(".thumbnail").css("opacity", "0.5");
   ```

3. Animate the opacity of the target thumbnail to 1 in 200 ms. This will produce an effect wherein only the target thumbnail is in focus while the rest appear faded:

   ```
   $(this).animate({ opacity: 1.0 }, 200);
   ```

4. Append an `<img>` element in the display area below the thumbnails. The css class of the enlarged image is `image` while the image source is the same as the target thumbnail:

   ```
   $("#imgContainer").append("<img class='image' src='" + $(this).
   attr("src") + "'/>");
   },
   ```

   The jQuery selector `$(this).attr("src")` retrieves the image source attribute of the target thumbnail.

5. Now, define a handler for the `mouseleave` event. Update the opacity of all thumbnails to 1:

   ```
   function() {
               $(".thumbnail").css("opacity", "1.0");
   ```

6. Select the enlarged image using its jQuery css selector `$(".image")`. Remove the enlarged image from the display area:

   ```
   $(".image").remove();
               }
   ```

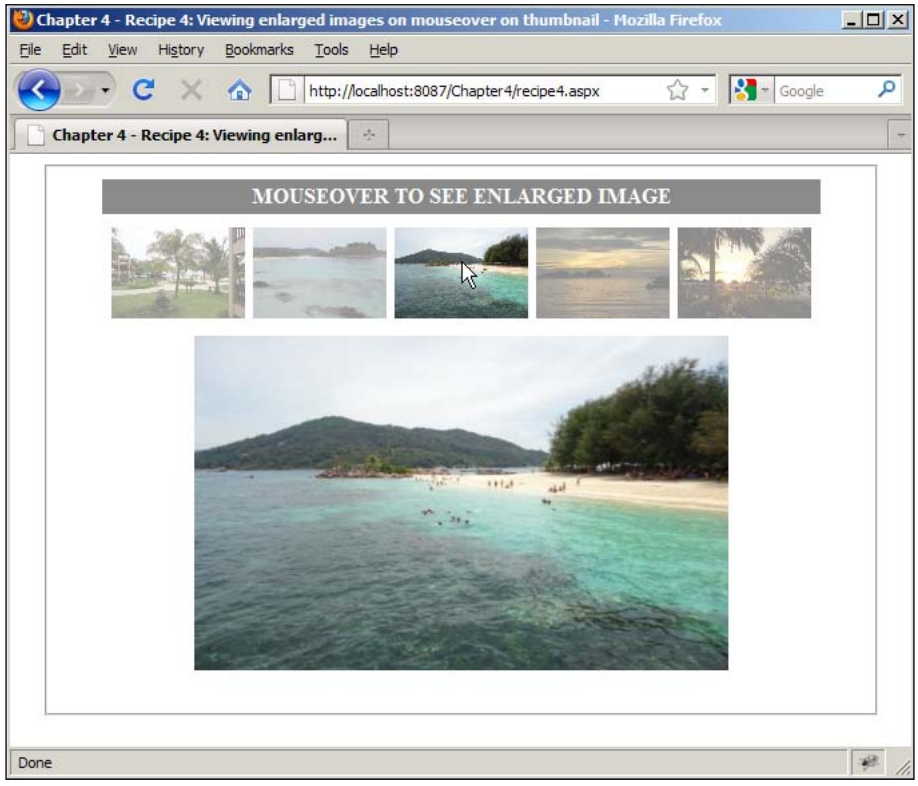
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        $(".thumbnail").hover(
```

```
            function() {
                $(".thumbnail").css("opacity", "0.5");
                $(this).animate({ opacity: 1.0 }, 200);
                $("#imgContainer").append("<img class='image' src='" +
$(this).attr("src") + "'/>");
            },
            function() {
            $(".thumbnail").css("opacity", "1.0");
                $(".image").remove();
            }

            );
        });
    </script>
```

How it works...

Run the web form. Mouseover on any thumbnail. You will see that the enlarged image appears in the display area below as shown in the following screen:



Note that on mouseover, only the target thumbnail is highlighted. The rest of the thumbnails appear faded.

On mouseout on the thumbnail, the image disappears from the display area.

See also

Creating an image gallery viewer

Swapping images on a page

In this recipe, we will swap an image on a page by updating its source property at runtime.

Getting ready

1. Add a new web form `Recipe5.aspx` to the current project.
2. Add a css class for the Image control as follows:

```
.image
    {
      position:relative;
      width:400px;
      height:250px;
    }
```

3 .Add an image control to the form and set its CssClass to the preceding class:

```
<asp:Image ID="Image1" ImageUrl="~/images/Image5.jpg"
CssClass="image" runat="server" />
```

4. Add a button control to trigger swapping of the image:

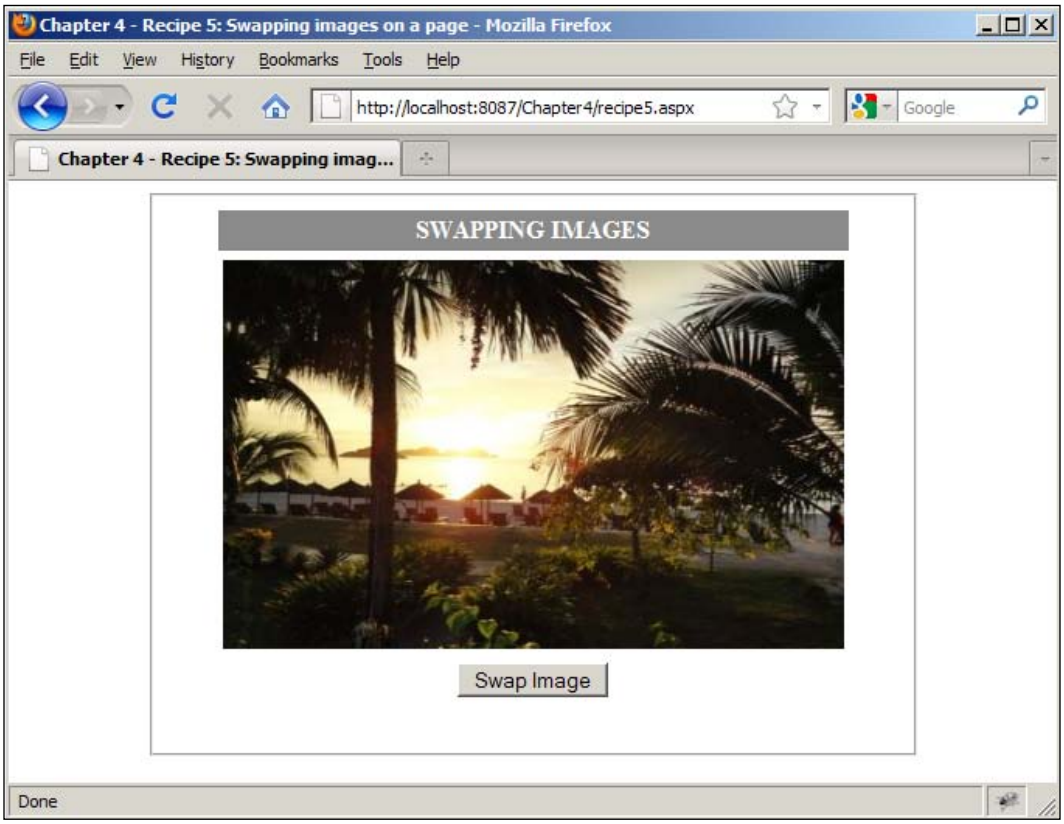
```
<asp:Button ID="btnSwap" Text="Swap Image" runat="server"></
asp:Button>
```

Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:470px;height:340px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">SWAPPING IMAGES</td></tr>
        <tr>
            <td colspan="2">
                <asp:Image ID="Image1" ImageUrl="~/images/Image5.jpg"
CssClass="image" runat="server" />
            </td>
```

```
            </tr>
            <tr><td colspan="2" align="center">
                <asp:Button ID="btnSwap" Text="Swap Image"
    runat="server"></asp:Button>
            </td></tr>
        </table>
        </fieldset>

    </div>
        </form>
```

On loading, the page displays as follows:



We will now swap the image on clicking the button control.

How to do it...

1. In the `document.ready()` function of the jQuery script block, enable the button control initially:

   ```
   $("#btnSwap").removeAttr("disabled");
   ```

2. Define a callback function on the `click` event of the button control using `bind`:

   ```
   $("#btnSwap").bind("click", function(e) {
   ```

3. Prevent default form submission:

   ```
   e.preventDefault();
   ```

4. Update the Image URL using the `src` attribute:

   ```
   $("#Image1").attr("src", "images/Image6.jpg");
   ```

5. Disable the button control by adding the attribute `disabled` to it:

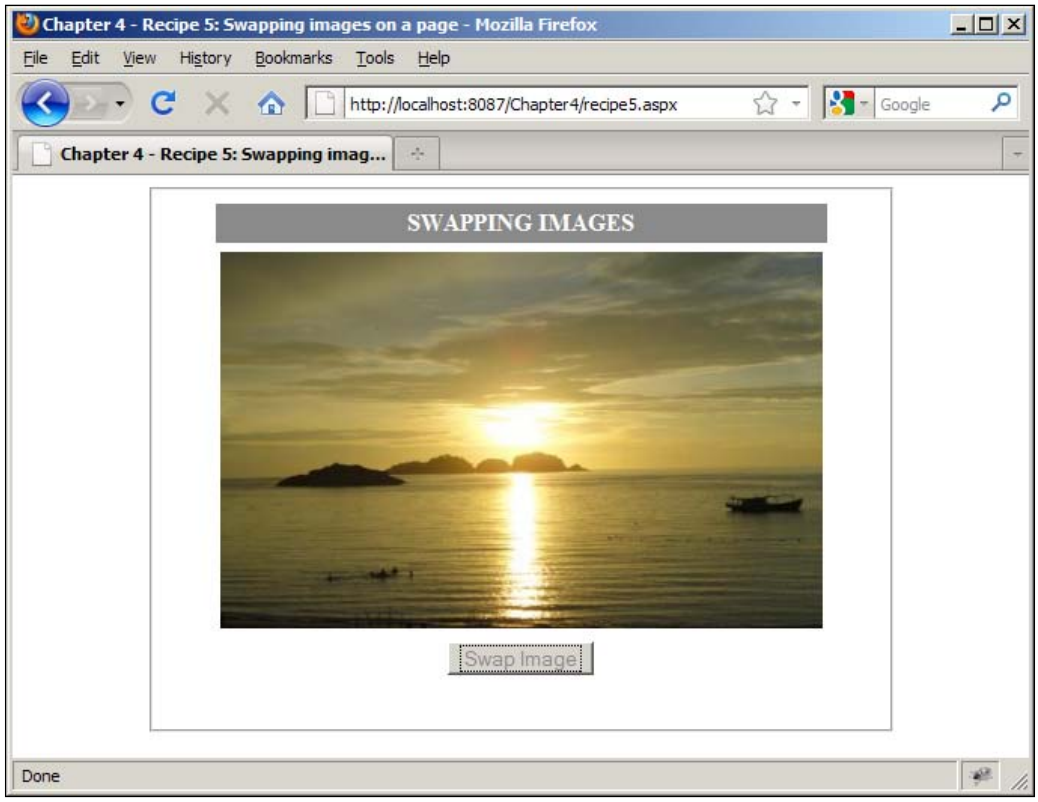
   ```
   $("#btnSwap").attr("disabled", "disabled");
               });
   ```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSwap").removeAttr("disabled");
            $("#btnSwap").bind("click", function(e) {
                e.preventDefault();
                $("#Image1").attr("src", "images/Image6.jpg");
                $("#btnSwap").attr("disabled", "disabled");
            });
        });
    </script>
```

How it works...

Run the web page. Click on the **Swap Image** button. The image source will update to display the following instead:



See also

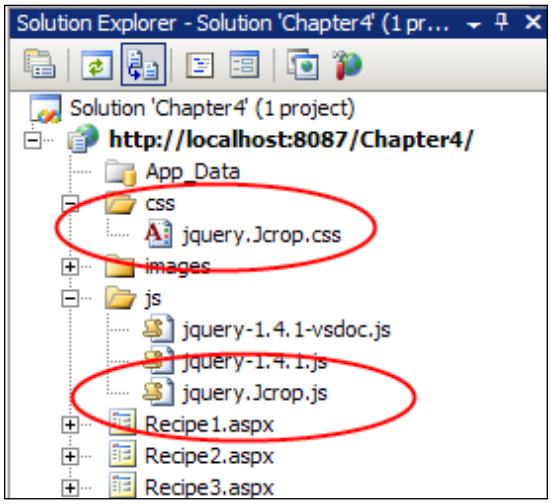
Changing image opacity on mouseover

Cropping images on a page

jQuery comes with a number of handy plugins to enhance the functionality of web controls. For applications requiring image cropping, a plugin called JCrop can be used with the Image control. In this recipe, we will use this plugin to crop images and display the co-ordinates of the crop area.

Getting ready

1. Add a new web form `Recipe6.aspx` to the current project.
2. Download the JCrop plugin from `http://deepliquid.com/content/Jcrop.html`.
3. Save the `jquery.Jcrop.js` file to the local `js` folder. Save the stylesheet file `jQuery.Jcrop.css` to the stylesheet folder `css`:



4. Add both the files we just saw to the page along with the jQuery library:

```
<link href="css/jquery.Jcrop.css" rel="stylesheet" type="text/css"
/>
    <script src="js/jquery-1.4.1.js" type="text/javascript"></
script>
    <script src="js/jquery.Jcrop.js" type="text/javascript"></
script>
```

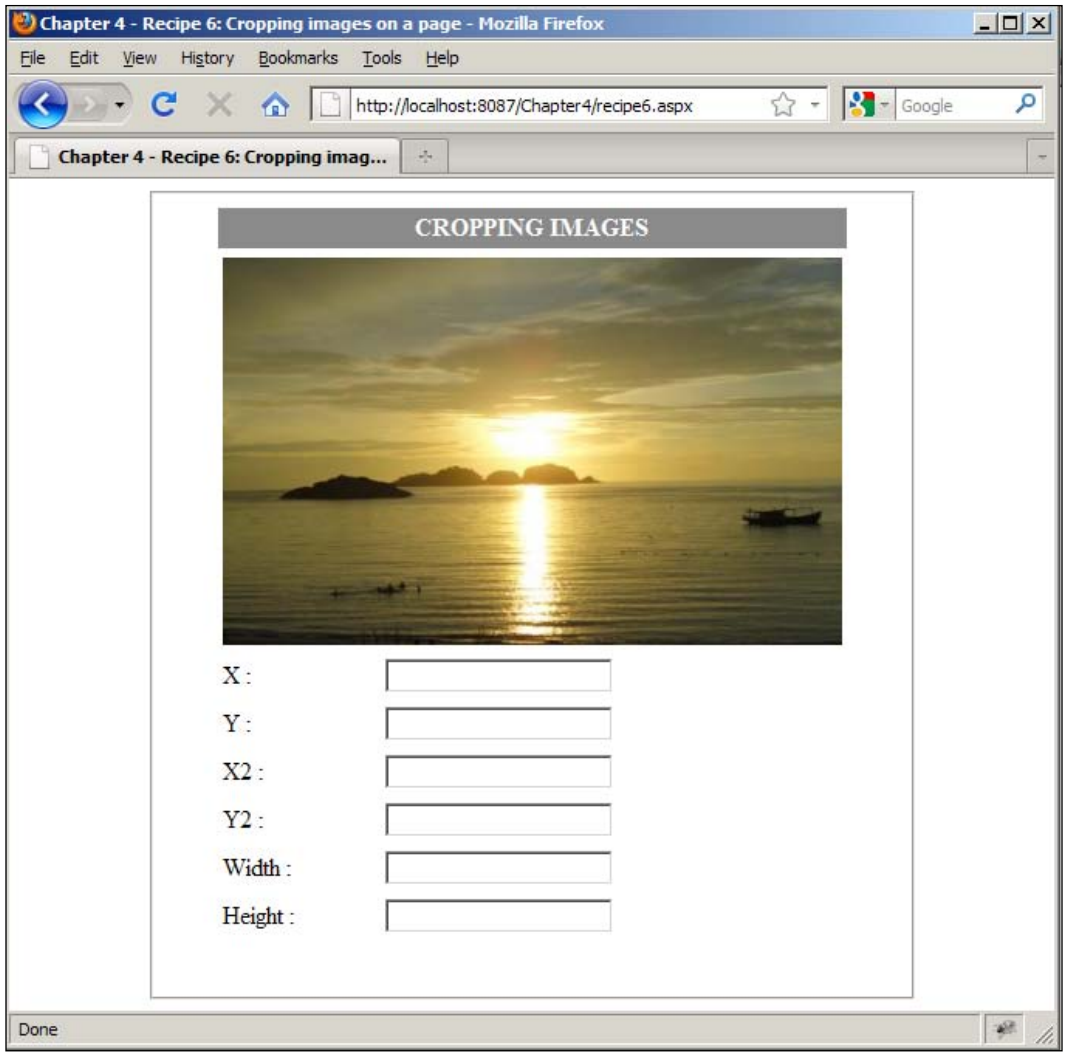
5. Add the following css class for the image:

```
.image
    {
      position:relative;
      width: 400px;
      height: 250px;
    }
```

6. Create the following ASPX markup for the web form:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:470px;height:500px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">CROPPING IMAGES</td></
tr>
        <tr>
            <td colspan="2">
                <asp:Image ID="Image1" CssClass="image"
ImageUrl="~/images/Image6.jpg" runat="server" />
            </td>
        </tr>
        <tr><td>X : </td>
        <td>
            <asp:TextBox ID="txtX" runat="server"></asp:TextBox></
td></tr>
         <tr><td>Y : </td>
        <td><asp:TextBox ID="txtY" runat="server"></asp:TextBox></
td></tr>
         <tr><td>X2 : </td>
        <td><asp:TextBox ID="txtX2" runat="server"></
asp:TextBox></td></tr>
         <tr><td>Y2 : </td>
        <td><asp:TextBox ID="txtY2" runat="server"></
asp:TextBox></td></tr>
         <tr><td>Width : </td>
        <td><asp:TextBox ID="txtWidth" runat="server"></
asp:TextBox></td></tr>
         <tr><td>Height : </td>
        <td><asp:TextBox ID="txtHeight" runat="server"></
asp:TextBox></td></tr>
    </table>
    </fieldset>

</div>
    </form>
```

Thus, on load, the page will display as follows:



How to do it...

1. In the `document.ready()` function of the jQuery script block, call the plugin method `JCrop` on the Image control:

   ```
   $("#Image1").Jcrop({
   ```

2. Specify the callback function on `onChange` event:

   ```
   onChange: showCoords,
   ```

3. Specify the callback function on `onSelect` event:

```
onSelect: showCoords
            });
```

4. Define the callback function. Pass the event object `e` in the callback function:

```
function showCoords(e) {
```

5. `(e.x,e.y)` retrieve the top-left co-ordinates of the crop area. Display these values in the (X,Y) TextBoxes:

```
$("#txtX").val(e.x);
$("#txtY").val(e.y);
```

6. `(e.x2,e.y2)` retrieve the bottom-right co-ordinates of the crop area. Display these in the (X2,Y2) TextBoxes:

```
$("#txtX2").val(e.x2);
$("#txtY2").val(e.y2);
```

7. `e.w` retrieves the width of the crop area. Display as follows:

```
$("#txtWidth").val(e.w);
```

8. `e.h` retrieves the height of the crop area. Display as follows:

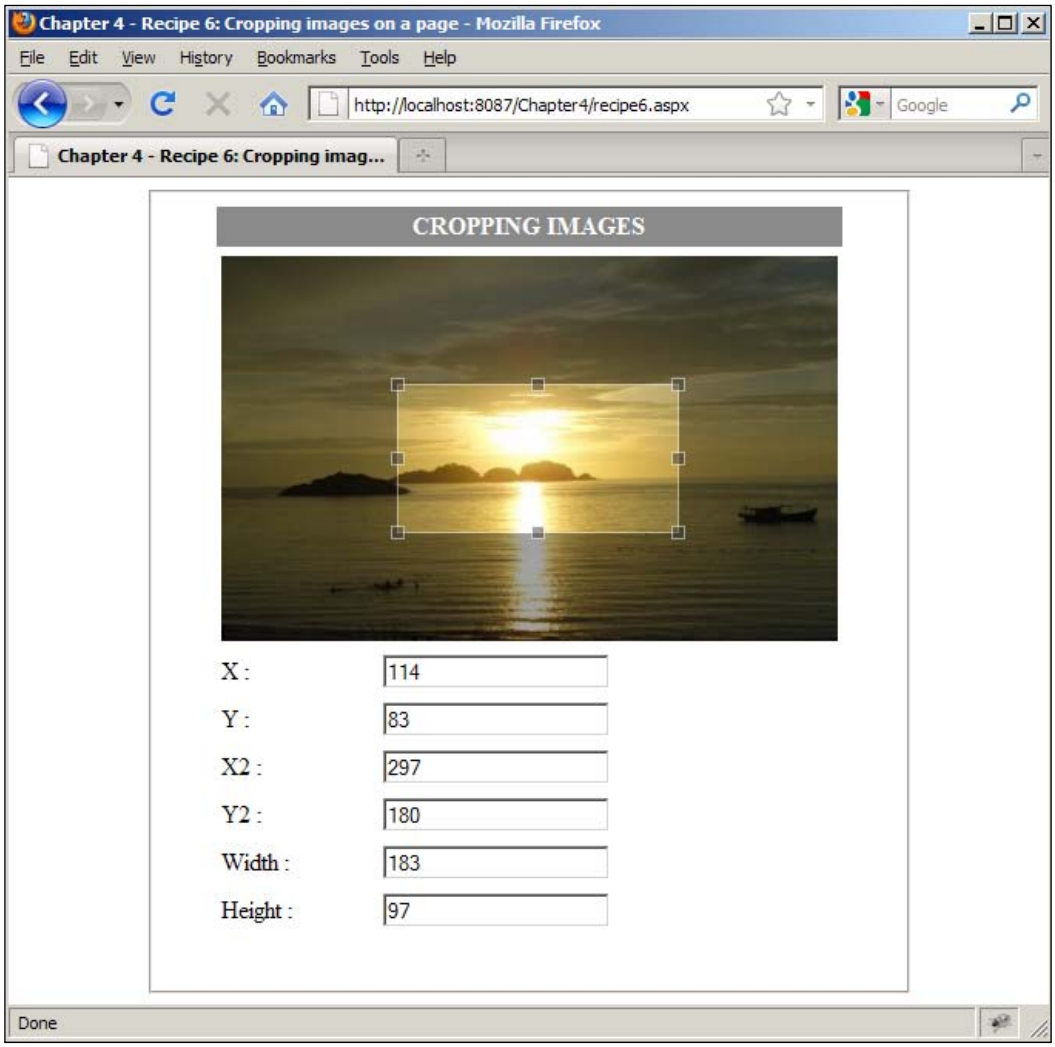
```
$("#txtHeight").val(e.h);
}
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#Image1").Jcrop({
                onChange: showCoords,
                onSelect: showCoords
            });
            function showCoords(e) {
                $("#txtX").val(e.x);
                $("#txtY").val(e.y);
                $("#txtX2").val(e.x2);
                $("#txtY2").val(e.y2);
                $("#txtWidth").val(e.w);
                $("#txtHeight").val(e.h);
            }
        });
    </script>
```

How it works...

Run the page. Click on any area of the image and drag the cursor to select the crop area as required. The page displays the co-ordinates as well as the dimensions of the crop area as follows:



See also

Swapping images on a page

Creating an image gallery viewer

In this recipe, we will create an image gallery viewer with the help of jQuery.

Getting ready

1. Add a new web form `Recipe7.aspx` to the current project.
2. Add a div area to the form and add a couple of Image controls to the div area:

```
<div class="imgcontainer">
                <asp:Image ID="Image1" ImageUrl="~/images/Image1.
jpg" runat="server" />
                <asp:Image ID="Image2" ImageUrl="~/images/Image2.
jpg" runat="server" />
                <asp:Image ID="Image3" ImageUrl="~/images/Image3.
jpg" runat="server" />
                <asp:Image ID="Image4" ImageUrl="~/images/Image4.
jpg" runat="server" />
                <asp:Image ID="Image5" ImageUrl="~/images/Image5.
jpg" runat="server" />
                <asp:Image ID="Image6" ImageUrl="~/images/Image6.
jpg" runat="server" />
                <asp:Image ID="Image7" ImageUrl="~/images/Image7.
jpg" runat="server" />
                <asp:Image ID="Image8" ImageUrl="~/images/Image8.
jpg" runat="server" />
            </div>
```

3. Attach the following css style to the div area:

```
.imgcontainer
    {
      position:relative;
      width:400px;
      height:250px;
    }
```

4. Attach the following css style to the Image controls. Note that the position of the images within the div area is defined as `absolute` and the top-left position is fixed:

```
.imgcontainer img
    {
      position:absolute;
      top:0;
      left:0;
      width:400px;
      height:250px;

    }
```

5. Add button controls for navigating to the `Previous` and `Next` images:

```
<asp:Button ID="btnPrevious" runat="server" Text="Previous" />
<asp:Button ID="btnNext" runat="server" Text="Next"/>
```

Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:470px;height:340px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">IMAGE GALLERY</td></tr>
        <tr>
            <td colspan="2">
            <div class="imgcontainer">
                <asp:Image ID="Image1" ImageUrl="~/images/Image1.jpg"
runat="server" />
                <asp:Image ID="Image2" ImageUrl="~/images/Image2.jpg"
runat="server" />
                <asp:Image ID="Image3" ImageUrl="~/images/Image3.jpg"
runat="server" />
                <asp:Image ID="Image4" ImageUrl="~/images/Image4.jpg"
runat="server" />
                <asp:Image ID="Image5" ImageUrl="~/images/Image5.jpg"
runat="server" />
                <asp:Image ID="Image6" ImageUrl="~/images/Image6.jpg"
runat="server" />
                <asp:Image ID="Image7" ImageUrl="~/images/Image7.jpg"
runat="server" />
                <asp:Image ID="Image8" ImageUrl="~/images/Image8.jpg"
runat="server" />
             </div>
            </td>
        </tr>
         <tr><td colspan="2" align="center">
            <asp:Button ID="btnPrevious" runat="server"
Text="Previous" /> 
            <asp:Button ID="btnNext" runat="server" Text="Next"/></
td></tr>
    </table>
    </fieldset>

</div>
    </form>
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define a variable for the z-index of an image:

```
var z = 0;
```

The z-index refers to an element's position on the Z-axis. It determines the stack order of an element. An element with a lower stack order will be overlapped by an element with a higher stack order.

2. Assuming the images are stacked one above the other, get the maximum possible z-index that is equivalent to the total number of Image controls on the page:

```
var max_z = $(".imgcontainer img").length;
```

Each Image control in the div area is assigned a unique z-index from 1 to max_z. The element with the highest index max_z will overlap all others and is the active image in the gallery.

3. Loop through the images in the div area:

```
$(".imgcontainer img").each(function() {
```

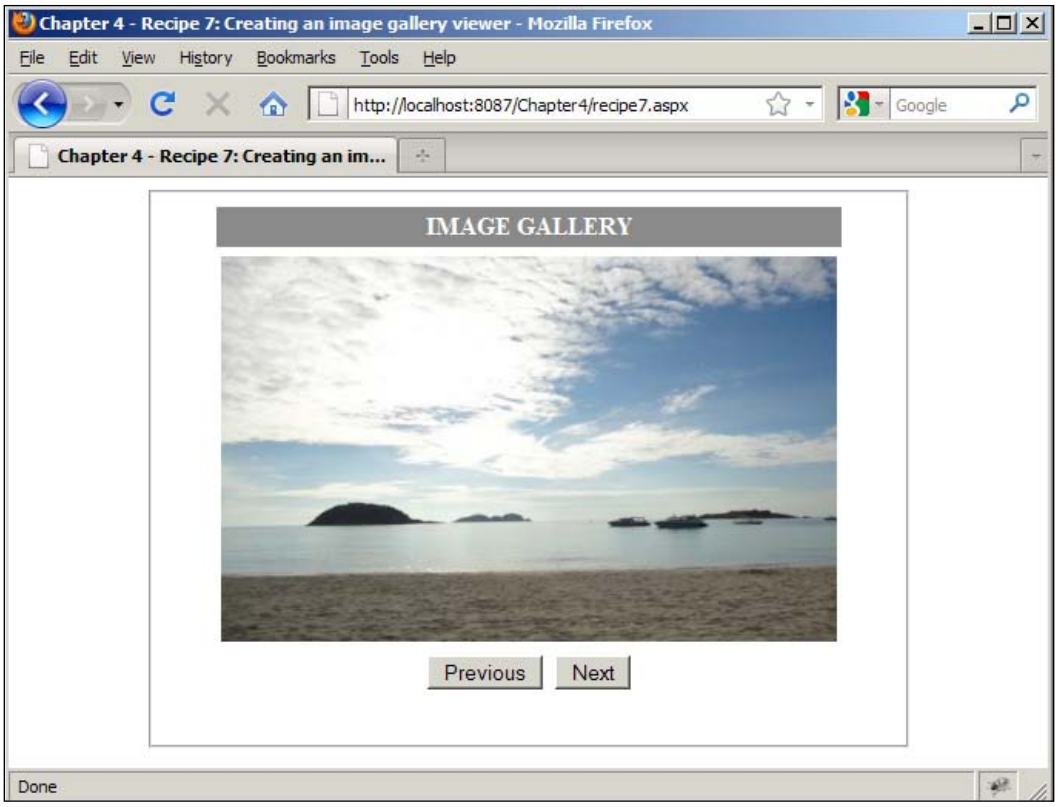
4. Increment the z-index for the next element:

```
z++;
```

5. Apply the respective z-index to the image:

```
$(this).css("z-index", z);
            });
```

6. Thus, on page load, the form displays the image with the highest z-index - `Image8`, as in the following screenshot:



On clicking the **Previous** button we will decrement the z-index of each Image control by 1. The image with z-index of 1 (the image at the bottom of the stack), will be assigned z-index of max_z, thus bringing it to the top of the stack and making it the active image.

7. Define the `click` event for the **Previous** button:
   ```
   $("#btnPrevious").bind("click", function(e) {
   ```
8. Prevent default form submission:
   ```
   e.preventDefault();
   ```
9. Loop through each Image control in the div area:
   ```
   $(".imgcontainer img").each(function() {
   ```
10. Get the z-index of the current Image control in the loop:
    ```
    var curr_z_index = parseInt($(this).css("z-index"));
    ```

11. If its z-index is the minimum i.e. 1) then reset to max_z:

```
if (curr_z_index == 1) {
                        $(this).css("z-index", max_z);
                    }
```

12. Otherwise, decrement the z-index of the Image control by 1:

```
else {
     $(this).css("z-index", (curr_z_index - 1));
                    }
                });
            });
```

On clicking the **Next** button we will increment the z-index of each Image control by 1. The active image i.e. the image with z-index of max_z), will be assigned a z-index of 1, thus putting it at the bottom of the stack.

13. Define the `click` event for the **Next** button:

```
$("#btnNext").bind("click", function(e) {
```

14. Prevent default form submission:

```
e.preventDefault();
```

15. Loop through each Image control in the div area:

```
$(".imgcontainer img").each(function() {
```

16. Get the z-index of the current Image control in the loop:

```
var curr_z_index = parseInt($(this).css("z-index"));
```

17. If its z-index is the maximum i.e. max_z) then reset to 1:

```
if (curr_z_index == max_z) {
          $(this).css("z-index", 1);
                    }
```

18. Otherwise, increment the z-index of the Image control by 1:

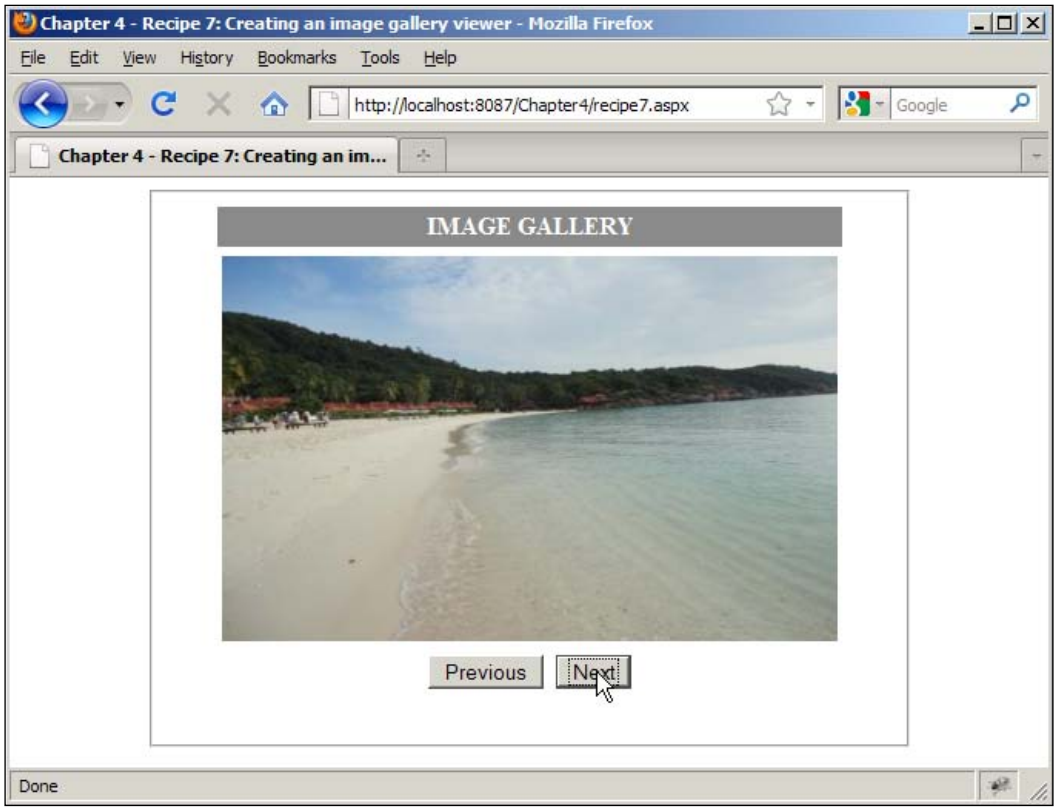
```
else {
      $(this).css("z-index", (curr_z_index + 1));
                    }
                });
            });
```

The complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            var z = 0;
            var max_z = $(".imgcontainer img").length;
            $(".imgcontainer img").each(function() {
                z++;
                $(this).css("z-index", z);
            });

            $("#btnPrevious").bind("click", function(e) {
                e.preventDefault();
                $(".imgcontainer img").each(function() {
                    var curr_z_index = parseInt($(this).css("z-
index"));
                    if (curr_z_index == 1) {
                        $(this).css("z-index", max_z);
                    }
                    else {
                        $(this).css("z-index", (curr_z_index - 1));
                    }
                });
            });

            $("#btnNext").bind("click", function(e) {
                e.preventDefault();
                $(".imgcontainer img").each(function() {
                    var curr_z_index = parseInt($(this).css("z-
index"));
                    if (curr_z_index == max_z) {
                        $(this).css("z-index", 1);
                    }
                    else {
                        $(this).css("z-index", (curr_z_index + 1));
                    }
                });
            });
        });
    </script>
```

How it works...

Run the web form. Click on the navigation buttons. The page will display the **Previous** or the **Next** images in the gallery as required.



See also

Viewing enlarged images on mouseover on a thumbnail.

Zooming images on mouseover

In this recipe, we will zoom an image on mouseover. The amount of zoom can be specified as required.

Getting ready

1. Create a new web form `Recipe8.aspx` in the current project.
2. Add a div area to the form. Add an Image control to the div area as follows:

```
<div class="imgcontainer">
                <asp:Image ID="Image1" CssClass="image"
ImageUrl="~/images/Image8.jpg" runat="server" />
                </div>
```

3 The css style for the div area is defined as follows:

```
.imgcontainer
    {
      width:400px;
      height:250px;
    }
```

Note that it is critical to specify the dimensions of the containing div area since we will use its dimensions to reset the image back to the original size.

4. The css style for the image is defined as follows:

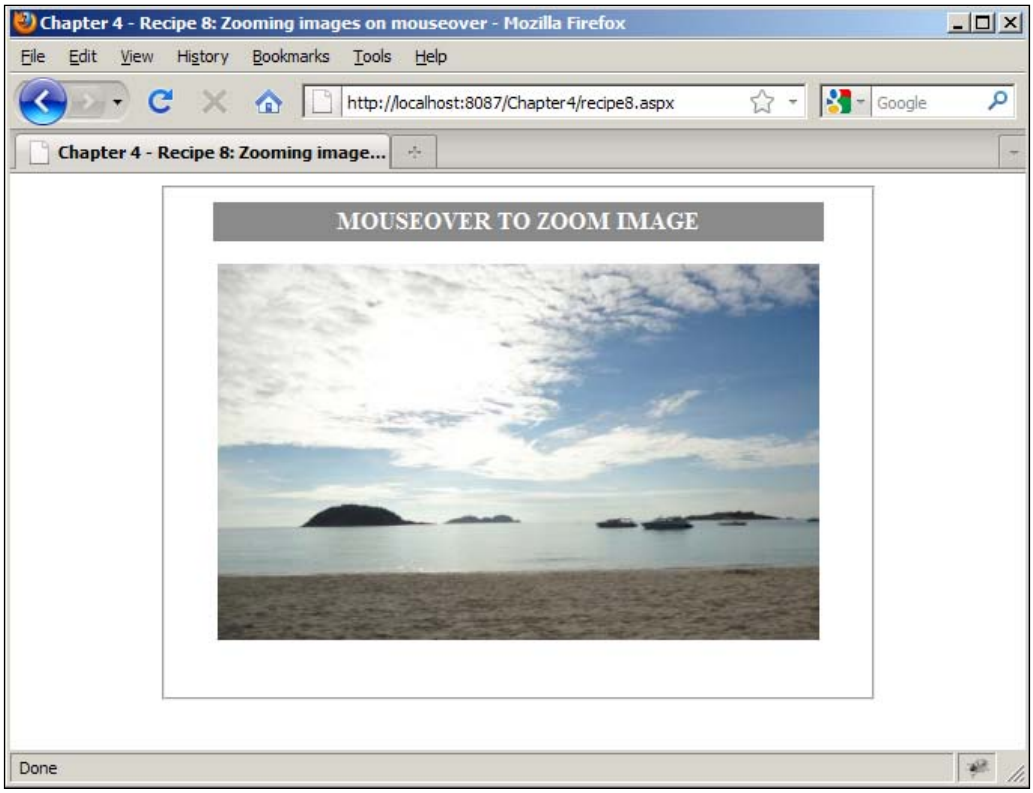
```
  .image
    {
      position:relative;
      top:0;
      left:0;
      width:400px;
      height:250px;
    }
```

Thus, the complete ASPX markup of the page is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:450px;height:320px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">MOUSEOVER TO ZOOM
IMAGE</td></tr>
       <tr><td colspan="2"></td></tr>
        <tr>
            <td colspan="2">
                <div class="imgcontainer">
                <asp:Image ID="Image1" CssClass="image"
ImageUrl="~/images/Image8.jpg" runat="server" />
```

```
                    </div>
                </td>
            </tr>
        </table>
        </fieldset>
    </div>
        </form>
```

On load, the page displays the image as follows:



How to do it...

1. In the `document.ready()` function of the jQuery script block, specify the amount of zoom required. In this example, we have used a zoom of 10 percent:

   ```
   var zoom = 1.1;
   ```

2. Specify the amount by which the image position i.e. the top and left positions can move on zooming:

   ```
   var move = -20;
   ```

3. Define the hover event for the image:

```
$("#Image1").hover(
```

4. In the callback function for the `mouseenter` event, find the zoomed width:

```
function() {
            var imgwidth = $("#Image1").width() * zoom;
```

5. Find the zoomed height:

```
var imgheight = $("#Image1").height() * zoom;
```

6. Animate the width, height, top, and left attributes of the image. The duration of the animation is 200 ms:

```
$(this).animate({ 'width': imgwidth, 'height': imgheight, 'top':
move, 'left': move }, { duration: 200 });
            },
```

7. In the callback function for the `mouseleave` event, restore the image back to the original dimensions. Use the container div area's dimensions to retrieve the original size of the image. Animate the width, height, top, and left properties in 100 ms:

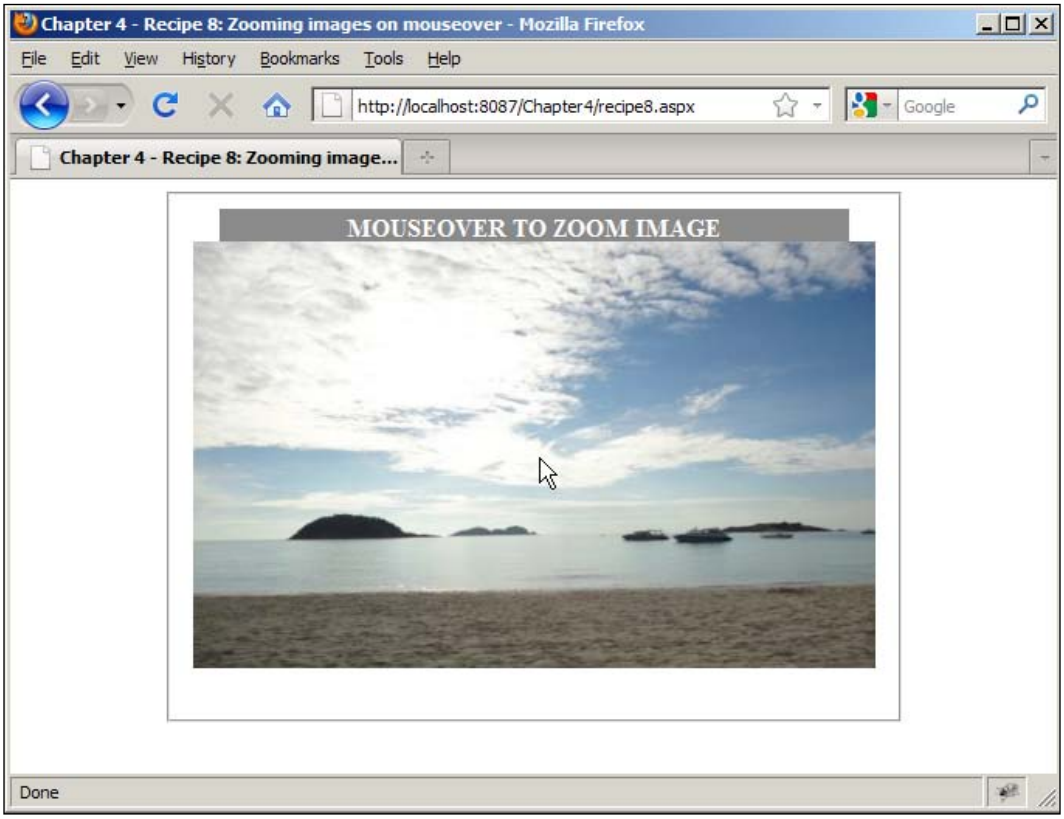
```
function() {
   $(this).animate({ 'width': $(".imgcontainer").width(),
'height': $(".imgcontainer").height(), 'top': '0', 'left': '0' },
{ duration: 100 });
            });
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            var zoom = 1.1;
            var move = -20;
            $("#Image1").hover(
            function() {
                var imgwidth = $("#Image1").width() * zoom;
                var imgheight = $("#Image1").height() * zoom;
                $(this).animate({ 'width': imgwidth, 'height':
imgheight, 'top': move, 'left': move }, { duration: 200 });
            },
            function() {
                //Reset the image
                $(this).animate({ 'width': $(".imgcontainer").width(),
'height': $(".imgcontainer").height(), 'top': '0', 'left': '0' }, {
duration: 100 });
            });
        });
    </script>
```

How it works...

Run the page. Mouseover on the image. You will notice that the image will zoom as shown below:



On mouseout, the image returns back to the original dimensions.

See also

Swapping images on a page

5
Animations in ASP.NET

In this chapter, we will cover:

- Enlarging text on hover
- Creating fade effect on hover
- Sliding elements on a page
- Preventing animation queue buildup
- Animating a panel
- Chaining animations together
- Hiding and displaying panels
- Creating disappearing effect

Introduction

jQuery offers many useful utilities to achieve animation effects, thus empowering developers to build rich animated pages for a better interactive experience for the web users.

Some useful inbuilt functions in jQuery that we will explore in this chapter for achieving animation effects are:

- **animate (properties, [duration], [easing], [complete])**:

  This method allows us to create custom animation effects on any numeric css property. The parameters supported by this method are:

  - **properties**: This is the map of css properties to animate, for e.g. width, height, fontSize, borderWidth, opacity, etc.

- **duration**: This is the duration of the animation in milliseconds. The constants `slow` and `fast` can be used to specify the durations, and they represent 600 ms and 200 ms respectively.
- **easing**: This is the easing function to use. Easing indicates the speed of the animation at different points during the animation. jQuery provides inbuilt `swing` and `linear` easing functions. Various plugins can be interfaced if other easing functions are required.
- **complete**: This indicates the callback function on completion of the animation.

- **fadeIn ([duration], [callback])**:

  This method animates the opacity of the matched elements from 0 to 1 i.e. transparent to opaque. The parameters accepted are:

  - **duration**: This is the duration of the animation
  - **callback**: This is the callback function on completion of the animation

- **fadeOut([duration], [callback])**:

  This method animates the opacity of the matched elements from 1 to 0 i.e. opaque to transparent. The parameters accepted are:

  - **duration**: This is the duration of the animation
  - **callback**: This is the callback function on completion of the animation

- **slideUp([duration], [callback])**:

  This method animates the height of the matched elements with an upward sliding motion. When the height of the element reaches 0, the css property `display` of the element is updated to `none` so that the element is hidden on the page. The parameters accepted are:

  - **duration**: This is the duration of the animation
  - **callback**: This is the callback function on completion of the animation

- **slideDown([duration], [callback])**:

  This method animates the height of the matched elements from 0 to the specified maximum height. Thus, the element appears to slide down on the page. The parameters accepted are:

  - **duration**: This is the duration of the animation
  - **callback**: This is the callback function on completion of the animation

- **slideToggle([duration], [callback])**:

  This method animates the height of the matched elements. If the element is initially hidden, it will slide down and become completely visible. If the element is initially visible, it will slide up and become hidden on the page. The parameters accepted are:

  - **duration**: This is the duration of the animation
  - **callback**: This is the callback function on completion of the animation

- **jQuery.fx.off**:

  If there is a need to disable animations because of a resource constraint or due to difficulties in viewing the animations, then this utility can be used to turn off the animation completely. This is achieved by setting all animated controls to their final state.

- **stop ([clearQueue], [jumpToEnd])**:

  This method stops the currently running animations on the page. The parameters accepted are:

  - **clearQueue**: This indicates whether any queued up animations are required to be cleared. The default value is false.
  - **jumpToEnd**: This indicates if the current animation is to be cleared immediately. The default value is false.

In this chapter, we will cover some of the animation effects that can be achieved in ASP.NET using the capabilities of jQuery.

Getting started

1. Let's start by creating a new ASP.NET website in Visual Studio and name it `Chapter5`.
2. Save the jQuery library in a script folder `js` in the project.
3. To enable jQuery on any web form, drag-and-drop to add the following to the page:

```
<scriptsrc="js/jquery-1.4.1.js" type="text/javascript"></script>
```

Now let's move on to the recipes where we will see different animation techniques using jQuery.

Enlarging text on hover

In this recipe, we will animate the font size of text content on hover.

Getting ready

1. Add a new web form `Recipe1.aspx` to the current project.
2. Create a Css class for the text content that we want to animate. The font size specified in the css Class is the original font size of the text before animation is applied to it:

```
.enlarge
    {
  font-size:12.5px;
  font-family:Arial,sans-serif;
    }
```

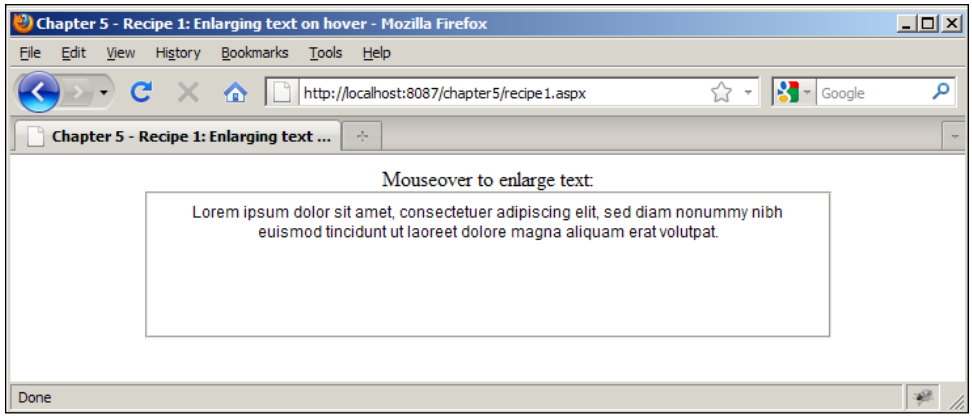
3. Add an ASP.NET Label control on the form and set its Css Class to the preceding style:

```
<asp:LabelCssClass="enlarge"runat="server">Lorem ipsum dolor sit
.....</asp:Label>
```

Thus, the ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
<div align="center">
    Mouseover to enlarge text:<br />
<fieldset id="content" style="width:500px;height:300px;">
<asp:LabelCssClass="enlarge" runat="server">Lorem ipsum dolor
sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</
asp:Label>
</fieldset>
</div>
</form>
```

Thus, initially, the page will display the Label control as follows:



We will now animate the font size of the Label on hover on the containing fieldset element.

How to do it...

1. In the `document.ready()` function of the jQuery script block, retrieve the original font size of the Label:

   ```
   var origFontSize = parseFloat($(".enlarge").css('font-size'));
   ```

   The `parseFloat()` function takes in an input string and returns the first floating point value in the string. It discards any content after the floating point value. For example, if the css property returns `12.5 px`, then the function will discard the `px`.

2. Define the hover event of the containing fieldset element:

   ```
   $("#content").hover(
   ```

3. In the `mouseenter` event handler of the hover method, update the cursor style to `pointer`:

   ```
   function() {
   $(".enlarge").css("cursor", "pointer");
   ```

4. Calculate the maximum font size that we want to animate to. In this example, we will set the maximum size to thrice the original:

   ```
   var newFontSize = origFontSize * 3;
   ```

5. Animate the `fontSize` css property of the Label in 300 ms:

   ```
   $(".enlarge").animate({ fontSize: newFontSize }, 300);
               },
   ```

6. In the `mouseleave` event handler of the hover method, animate the `fontSize` to the original value in 300 ms as shown:

   ```
   function() {
   $(".enlarge").animate({ fontSize: origFontSize }, 300);
   }
   );
   ```

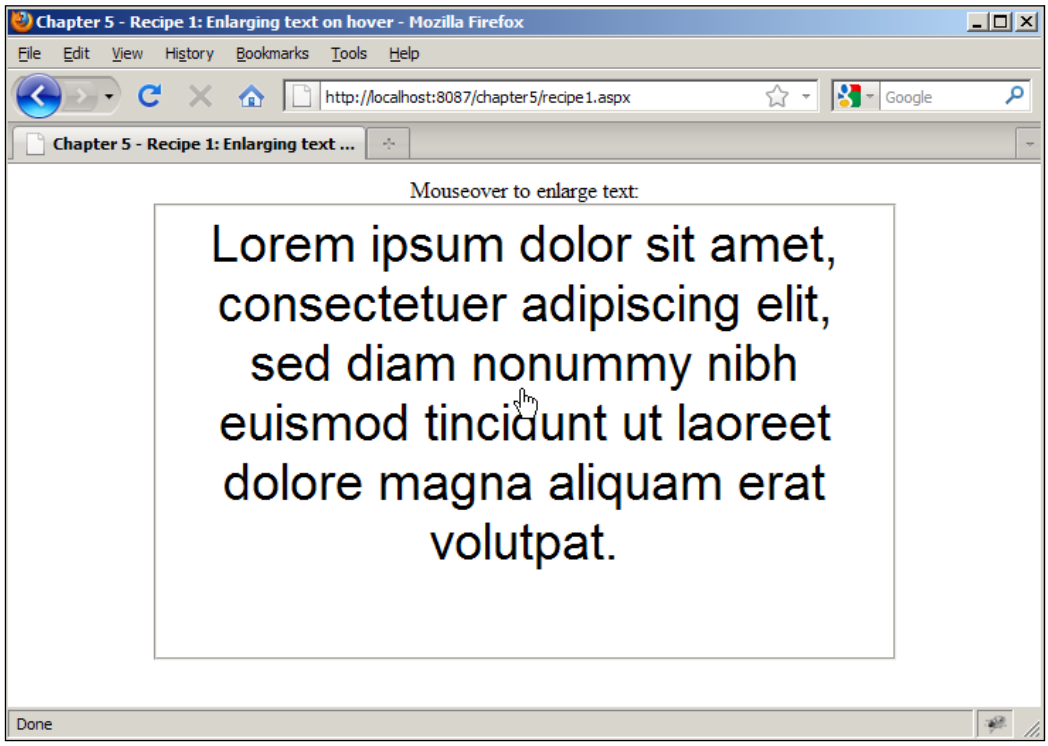
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            var origFontSize = parseFloat($(".enlarge").css('font-
size'));
            $("#content").hover(
            function() {
```

```
                $(".enlarge").css("cursor", "pointer");
                var newFontSize = origFontSize * 3;
                $(".enlarge").animate({ fontSize: newFontSize }, 300);
            },
            function() {
                $(".enlarge").animate({ fontSize: origFontSize },
300);
            }
);
        });
</script>
```

How it works...

Run the web form. Mouseover on the fieldset area. The text size will animate over the stated duration and change to the maximum specified font size as displayed in the following screenshot:



On removing the mouse from the fieldset area, the text size will return back to the original.

See also

Animating a panel.

Creating a fade effect on hover

In this recipe, we will create a fade effect on an ASP.NET Image control on hover. We will use the `fadeIn` and `fadeOut` methods to achieve the same.

Getting ready

1. Add a new web form `Recipe2.aspx` to the current project.
2. Add an image control to the form:

```
<asp:Image src="images/Image1.jpg" ID="Image1" runat="server" />
```

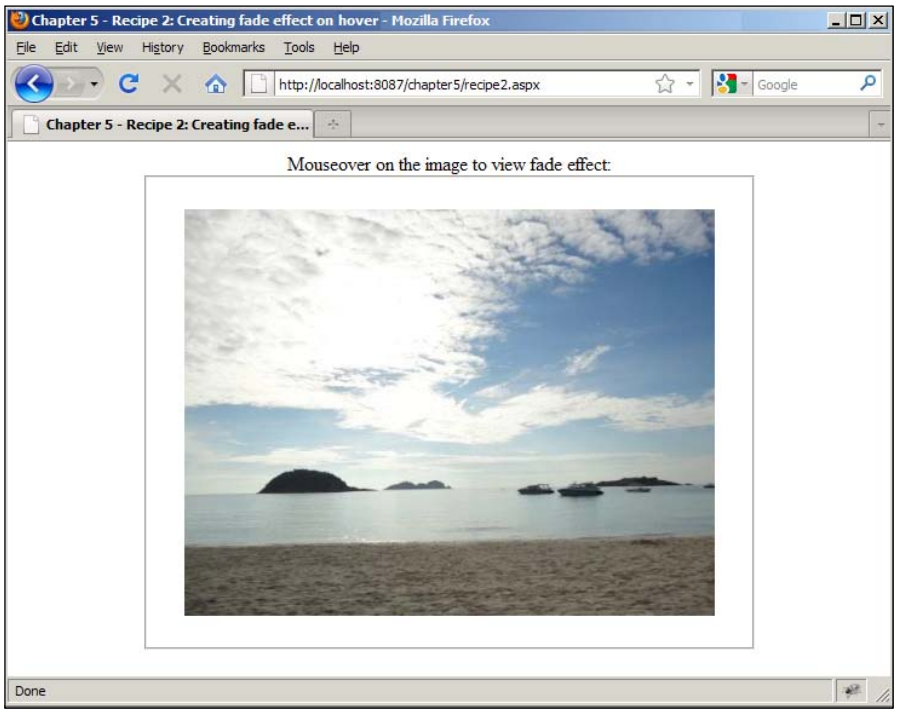
3. Define the properties of the image in the css:

```
#Image1
    {
      width:438px;
      height:336px;
    }
```

4. Thus, the complete ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
<div align="center">
    Mouseover on the image to view fade effect:
<fieldset id="content" style="width:480px;height:370px;">
<br />
<asp:Image src="images/Image1.jpg" ID="Image1" runat="server" />
</fieldset>
</div>
</form>
```

5. On page load, the image is displayed as follows:



We will now create a fade effect on the image on hover on the containing fieldset area.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the `hover` event on the containing fieldset area:

```
$("#content").hover(
```

2. In the `mouseenter` event handler of the hover method, update the cursor to pointer:

```
function() {
            $("#Image1").css("cursor", "pointer");
```

3. Apply the `fadeOut` method on the Image control with an animation duration of 1000 ms:

```
$("#Image1").fadeOut(1000);
          },
```

4. In the `mouseleave` event handler of the hover method, apply the `fadeIn` method on the Image control with an animation duration of 1000 ms:

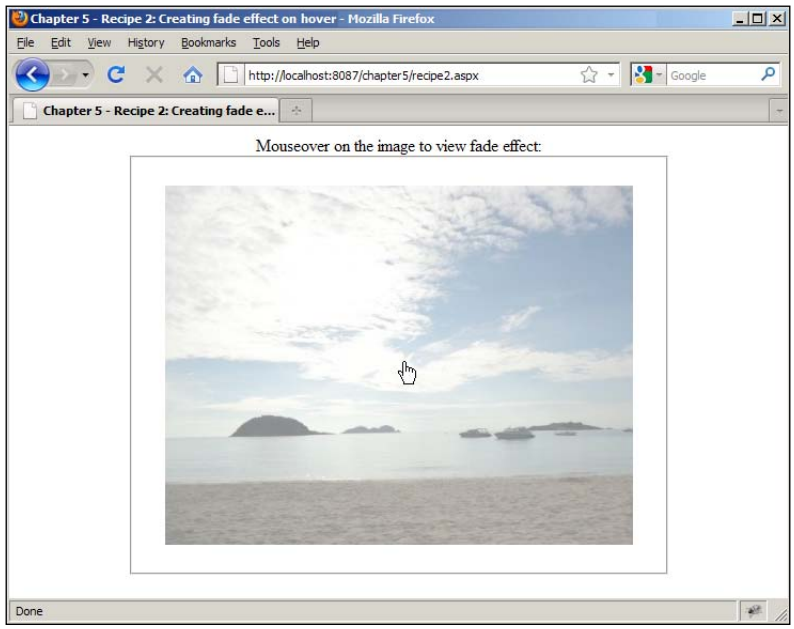
```
    function() {
                    $("#Image1").fadeIn(1000);
                }
    );
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        $("#content").hover(
             function() {
                $("#Image1").css("cursor", "pointer");
                $("#Image1").fadeOut(1000);
            },
             function() {
                 $("#Image1").fadeIn(1000);
            }
);
        });
</script>
```

How it works...

Run the web page. Mouseover on the Image control on the web page. The image will slowly fade away as shown in the following screenshot:



On mouseout from the containing fieldset area, the image reappears.

See also

Preventing animation queue buildup

Sliding elements on a page

In this recipe, we will use the `slideUp` and `slideDown` methods for achieving sliding effects on an ASP.NET panel.

Getting ready

1. Add a new web form `Recipe3.aspx` in the current project.
2. Add an ASP.NET panel to the page as follows:

```
<asp:Panel class="slide" runat="server">
            Sliding Panel
</asp:Panel>
```

3. The css class for the panel is defined as follows:

```
.slide
    {
  font-size:12px;
  font-family:Arial,sans-serif;
  display:none;
  height:100px;
  background-color:#9999FF;
    }
```

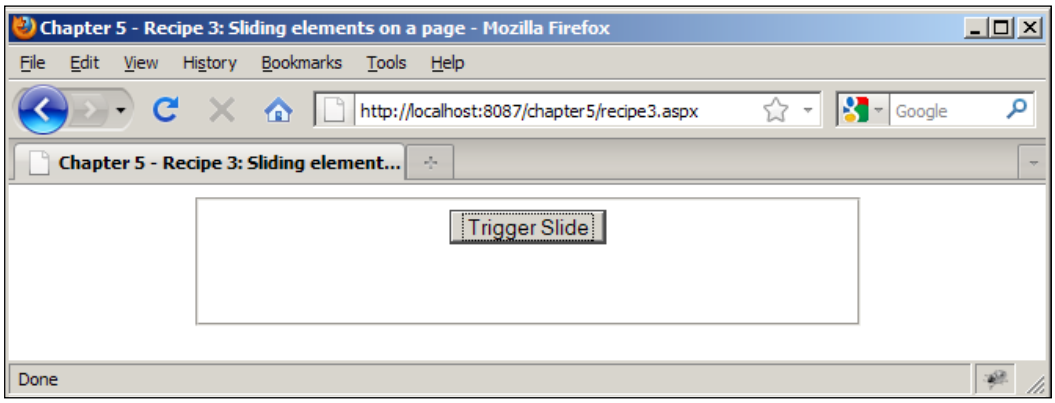
4. Add a button control to trigger the sliding effect on the panel:

```
<asp:Button ID="btnSubmit" runat="server" Text="Trigger Slide" />
```

5. Thus, the complete ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:400px;height:150px;">
<asp:Button ID="btnSubmit" runat="server" Text="Trigger Slide" />
<br /><br/>
<asp:Panel class="slide" runat="server">
            Sliding Panel
</asp:Panel>
</fieldset>

</div>
</form>
```

On page load, the page appears as shown in the following screenshot:



We will now use jQuery to slide up and slide down the panel .

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the `click` event of the button control:

   ```
   $("#btnSubmit").click(function(e) {
   ```

2. Prevent default form submission:

   ```
   e.preventDefault();
   ```

3. Check if the ASP.NET panel control is hidden:

   ```
   if ($(".slide").is(":hidden"))
   ```

   The jQuery selector `:hidden` selects matched elements that are hidden on the page.

4. If yes, then slide down the panel until its height reaches the maximum (`100 px`) defined in the css property.

   ```
   $(".slide").slideDown("slow");
   ```

5. If the panel is initially visible then slide up so that its height slowly reduces until it becomes 0 and the panel disappears from the page:

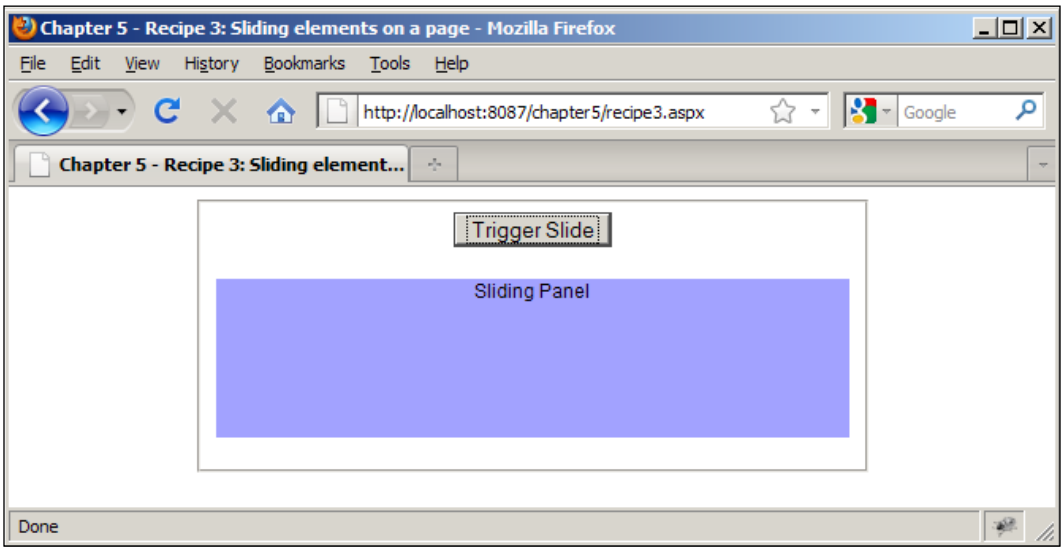
   ```
   else
     $(".slide").slideUp("slow");
   });
   ```

Thus, the complete jQuery solution is as follows:

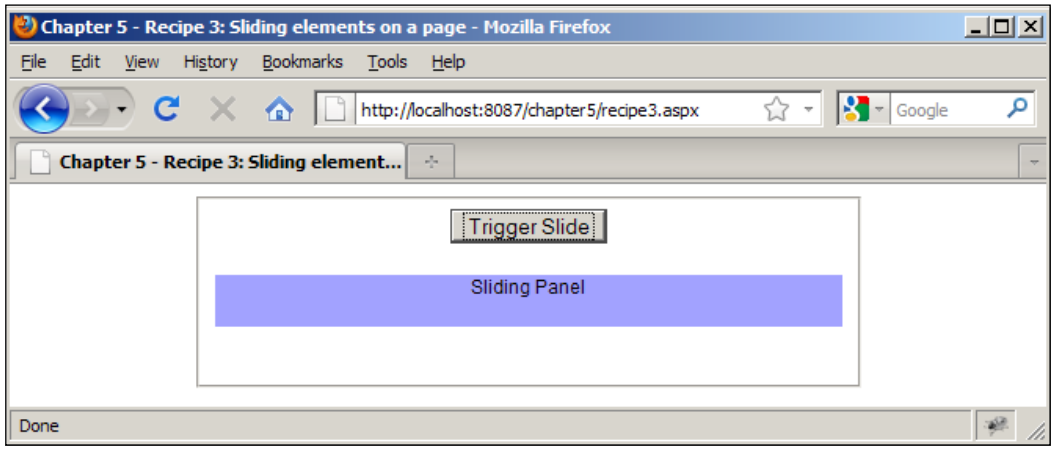
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                if ($(".slide").is(":hidden"))
                    $(".slide").slideDown("slow");
                else
                    $(".slide").slideUp("slow");
            });
        });
</script>
```

How it works...

Run the page. Initially, the panel is invisible. Now click the **Trigger Slide** button. The panel height is animated until it becomes completely visible as follows:



On clicking the button once again, the panel height is animated again so that it decreases until it disappears from the page.



There's more...

The same effect can also be achieved using the slideToggle method on the panel as follows:

```
$(".slide").slideToggle("slow");
```

See also

Hiding and displaying panels

Preventing animation queue buildup

If multiple animation events are triggered on an ASP.NET form simultaneously, jQuery queues the animations and executes the complete queue. This might sometimes lead to undesirable results. In this recipe, we will see the impact of an animation queue buildup and how we can prevent the same.

Getting ready

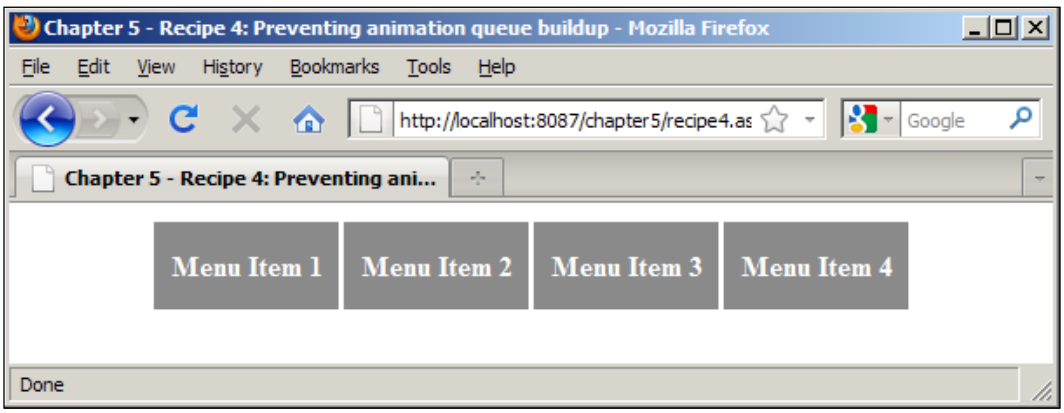
1. Add a new web form `Recipe4.aspx` to the current project.
2. Define a css style for a simple menu control:

```
.menuitem
    {
  background-color:Gray;
  font-weight:bold;
  color:White;
  text-align:center;
  width:100px;
  height:50px;
  cursor:pointer;
    }
```

The complete ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<table border="0" cellpadding="3" cellspacing="3">
<tr>
<td class="menuitem">Menu Item 1</td>
<td class="menuitem">Menu Item 2</td>
<td class="menuitem">Menu Item 3</td>
<td class="menuitem">Menu Item 4</td>
</tr>
</table>
</div>
</form>
```

On page load, the web form displays the menu items as shown in the following screenshot:



We will now animate the opacity of the menu items on mouseover:

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the `hover` event on the menu items using jQuery's css selector:

   ```
   $(".menuitem").hover(function(){
   ```

2. In the `mouseenter` event handler of the hover method, animate the opacity of the menu items to 0.5:

   ```
   $(this).animate({opacity:0.5},"slow");
                },
   ```

3. In the `mouseleave` event handler of the hover method, animate the opacity back to 1 i.e. opaque):

   ```
   function(){
   $(this).animate({opacity:1},"slow");
                });
   ```

Thus, the complete jQuery script is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
//PART 1 - ANIMATION QUEUE BUILDUP
              $(".menuitem").hover(function(){
                       $(this).animate({opacity:0.5},"slow");
              },
              function(){
                      $(this).animate({opacity:1},"slow");
              });
});
</script>
```

Run the page and observe the behavior of the menu items.

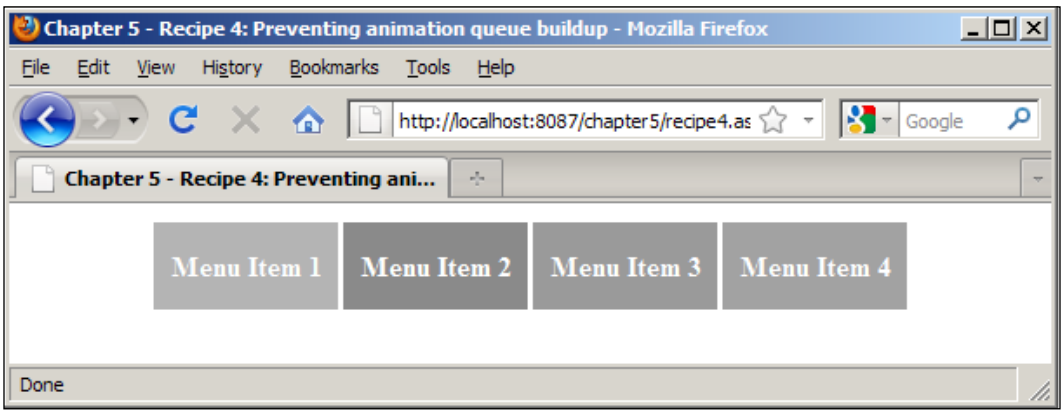
Now, update the preceding script slightly to include the `stop()` method in the animation as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
              //PART 2 - PREVENTING ANIMATION QUEUE BUILDUP
              $(".menuitem").hover(function(){
$(this) .stop().animate({opacity:0.5},"slow");
              },
              function(){
           $(this) .stop().animate({opacity:1},"slow");
              });
});
</script>
```

Run the page again and observe the behavior of the menu items.

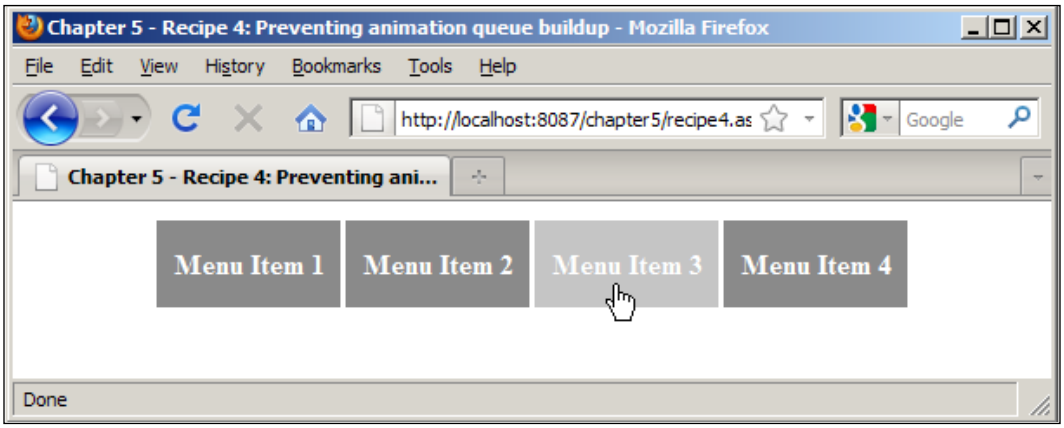
How it works...

During the first run, when the mouse is moved randomly over the menu items, the opacity level changes on `mouseenter` and `mouseleave` events on the respective target items. You will also notice that this effect queues up for each menu item that had `mouseenter`/`mouseleave` events triggered as shown in the following screenshot:



Even after moving the cursor away, jQuery continues to run the animations in the queue.

However, during the second run when the `stop()`method is called before initiating a call to the `animate()` method, `stop()` terminates all currently running animations by bringing the animated objects to the final state. As a result, only the opacity level of the target menu item changes on hover and the rest of the menu items are returned to an opaque state as demonstrated in the following screenshot:



See also

Chaining animations together

Animating a panel

In this recipe, let's animate different properties of an ASP.NET panel such as width, height, opacity, font size, and border width simultaneously on a web page.

Getting ready

1. Add a new web form `Recipe5.aspx` to the current project.
2. Add an ASP.NET panel control to the form:

```
<asp:Panel class="contentArea" runat="server">
            Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat.
</asp:Panel>
```

3. Set its css class as shown:

```
.contentArea
    {
  font-size:12px;
  font-family:Arial,sans-serif;
  width:200px;
  height:100px;
  background-color:#9999FF;
  border:solid;
    }
```

4. Add a button control to trigger the animation effect:

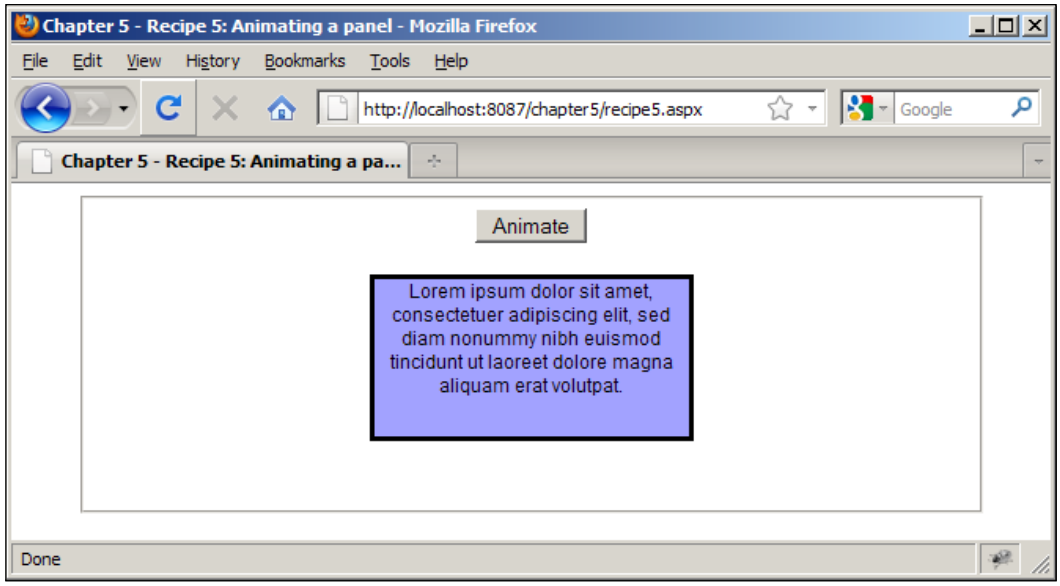
```
<asp:Button ID="btnSubmit" runat="server" Text="Animate" />
```

Thus, the complete ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:550px;height:370px;">
<asp:Button ID="btnSubmit" runat="server" Text="Animate" />
<br /><br />
<asp:Panel class="contentArea" runat="server">
```

```
            Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna
aliquam erat volutpat.
</asp:Panel>
</fieldset>
</div>
</form>
```

Thus, on page load, the page appears as follows:



How to do it...

1. In the `document.ready()` function of the jQuery script block, enable the button control explicitly on page load:

   ```
   $("#btnSubmit").removeAttr("disabled", "disabled");
   ```

2. Define the click event of the button control:

   ```
   $("#btnSubmit").click(function(e) {
   ```

3. Prevent default form submission:

```
e.preventDefault();
```

4. Use jQuery's css selector to retrieve the panel control and call the animate method providing a map of css properties to animate. Define the animation duration as `slow`:

```
$(".contentArea").animate({
                    opacity: 0.5,
                    width: "500px",
                    height: "280px",
                    fontSize: "36px",
                    borderWidth: "15px"
                }, "slow");
```

The above map of css properties causes the panel control to animate to an opacity of 0.5, width of 500 px, height of 280 px, font size of 36 px, and border width of 15 px in a duration of 600 ms.

5. Disable the button control at the end of the animation:

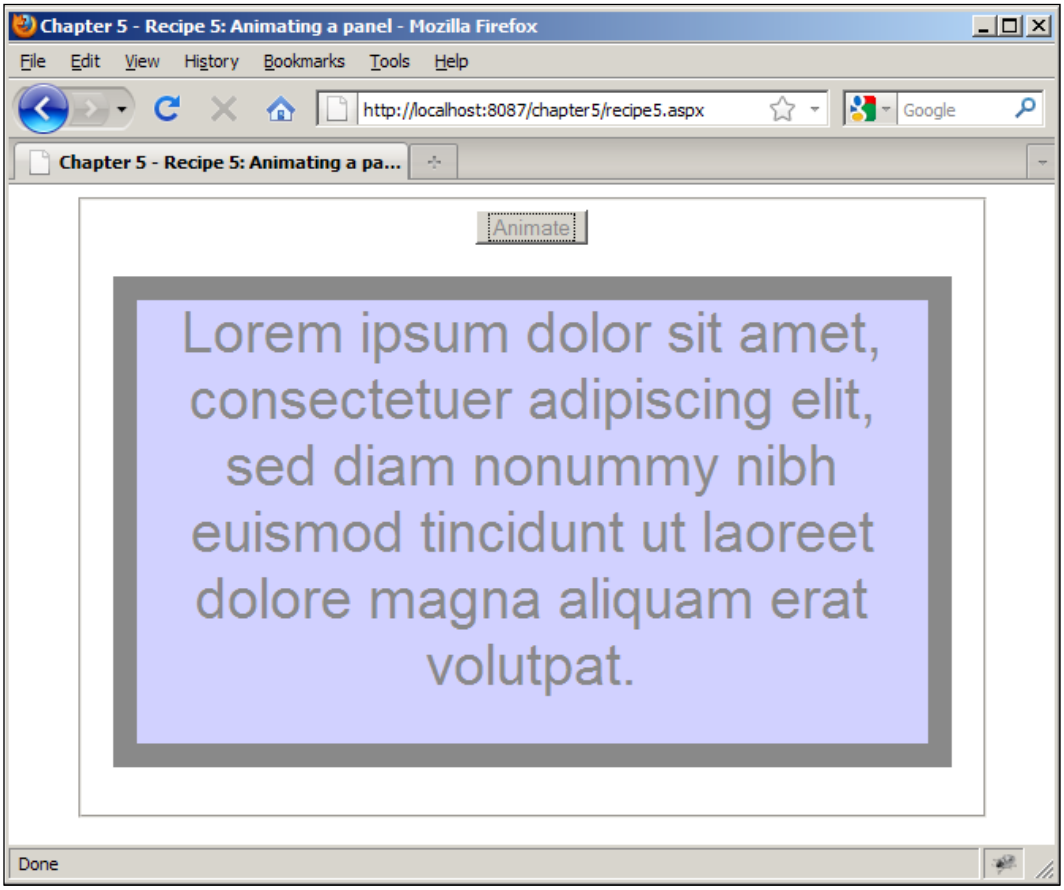
```
$(this).attr("disabled", "disabled");
            });
```

Thus the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").removeAttr("disabled", "disabled");
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                $(".contentArea").animate({
                    opacity: 0.5,
                    width: "500px",
                    height: "280px",
                    fontSize: "36px",
                    borderWidth: "15px"
                }, "slow");
                $(this).attr("disabled", "disabled");
            });
        });
</script>
```

How it works...

Run the web page. Click on the **Animate** button to kick start the animation effect. The page displays the panel control as follows at the end of the animation:



See also

Creating disappearing effect

Chaining animations together

In this example, let's see how a sequence of animations can be chained together to create a continuous animation effect. We will also vary the speed of each animation in the sequence.

Getting ready

1. Add a new web form `Recipe6.aspx` to the current project.
2. Add an ASP.NET panel control to the form:

```
<asp:Panel class="contentArea" runat="server">
</asp:Panel>
```

3. Sets its css class as follows:

```
.contentArea
    {
  width:50px;
  height:50px;
  background-color:#9999FF;
  border:solid;
  position:relative;
    }
```

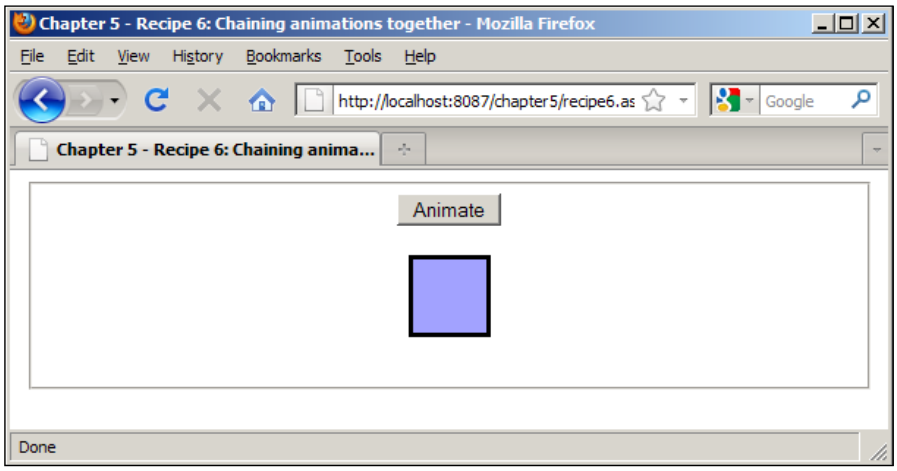
4. Add a button control to the form to trigger the animation sequence:

```
<asp:Button ID="btnSubmit" runat="server" Text="Animate" />
```

Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:550px;height:250px;">
<asp:Button ID="btnSubmit" runat="server" Text="Animate" />
<br /><br />
<asp:Panel class="contentArea" runat="server">
</asp:Panel>
</fieldset>
</div>
</form>
```

On load, the page will display the panel control as shown in the following screenshot:



We will now apply a series of sliding motion to the panel control by chaining the animation functions.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the `click` event of the button control:

   ```
   $("#btnSubmit").click(function(e){
   ```

2. Prevent default form submission:

   ```
   e.preventDefault();
   ```

3. Retrieve the panel control using the jQuery css selector and animate the left offset and opacity of the panel control in 2000 ms:

   ```
   $('.contentArea').animate({left:"+=200", opacity: 0.8}, 2000)
   ```

4. Chain another animation method to the one we just saw to animate the top offset and opacity of the panel control in 700 ms:

   ```
   .animate({top:"+=100", opacity:0.5},700)
   ```

5. Chain another animation method to the one we just saw to animate the left offset and opacity of the panel control in 1200 ms:

   ```
   .animate({left:"-=400", opacity: 0.2},1200)
   ```

6. Chain another animation method to the one we just saw to animate the top offset and opacity of the panel control in 700 ms:

   ```
   .animate({top:"-=100", opacity: 0.5},700)
   ```

7. Chain another animation method to the one we just saw to animate the left offset and opacity of the panel control in 2000 ms:

```
.animate({left:"+=200", opacity: 0.8},2000);
          });
```

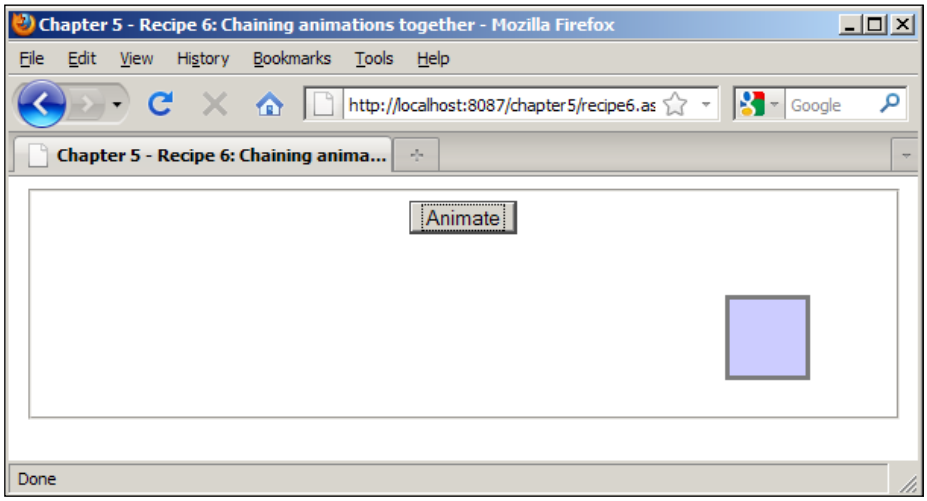
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
           $("#btnSubmit").click(function(e){
               e.preventDefault();
              $('.contentArea').animate({left:"+=200", opacity: 0.8},
2000)
                 .animate({top:"+=100", opacity:0.5},700)
            .animate({left:"-=400", opacity: 0.2},1200)
            .animate({top:"-=100", opacity: 0.5},700)
                 .animate({left:"+=200", opacity: 0.8},2000);
            });
        });
</script>
```

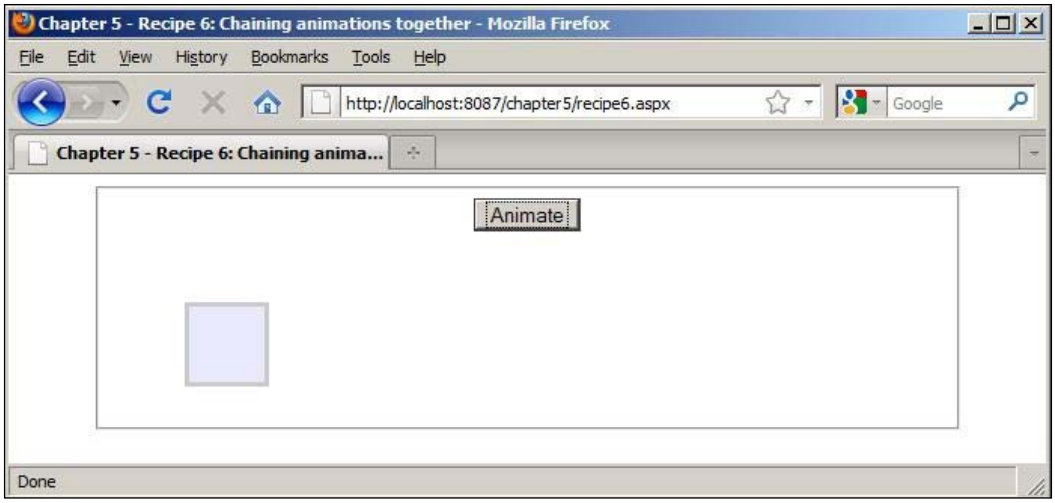
How it works...

Run the web page. Click on the **Animate** button. You will notice that the panel control goes through the following sequence of animations:

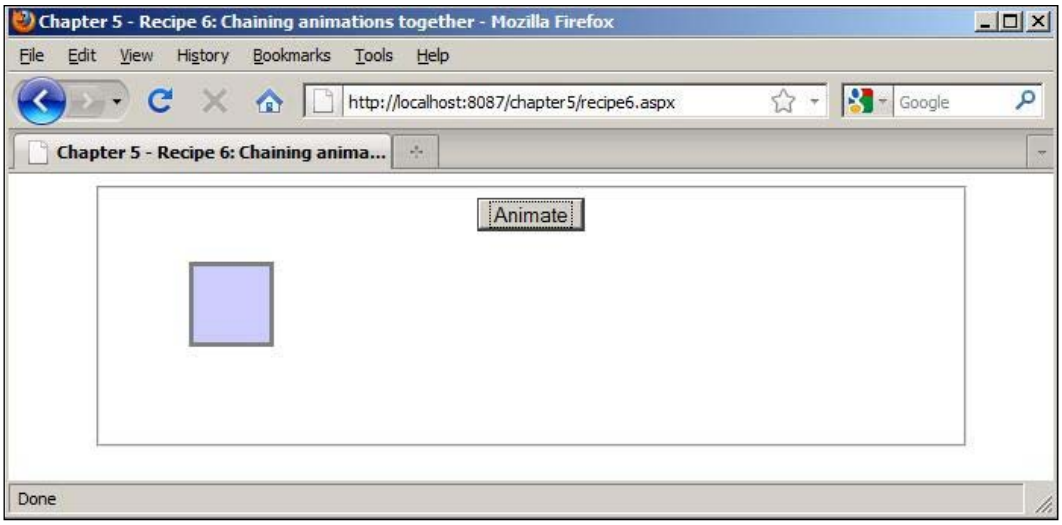
1. Slides to the right by 200 px.
2. Slides to the bottom by 100 px as captured:



3. Slides to the left by 400 px as displayed next:



4. Slides to the top by 100 px as displayed next:



5. Slides to the right by 200 px.

Thus, the panel returns back to the original position after the sequence of animation is completed. Notice that the opacity levels and the duration of the animation differs from one step to the next.

See also

Sliding elements on a page

Hiding and displaying panels

Often in web applications, there is a need to dynamically display or hide objects based on certain events triggered by the web user. One such example is a dynamic navigation bar where a sub navigation is to be turned on/off. In this recipe, we will see how to achieve this using jQuery.

Getting ready

1. Add a new web form `Recipe7.aspx` to the current project.
2. Create a couple of main menu and sub-menu items in a vertical navigation bar using ASP.NET panel controls as shown in the following example:

```
<asp:Panel class="content_head" runat="server">Menu Item 1</
asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 11
<br />Sub Menu Item 12</asp:Panel>
```

3. Define the css style of the main menu item as follows:

```
.content_head
    {
        width:150px;
        height:30px;
        background-color:#CCCCCC;
        cursor:pointer;
    }
```

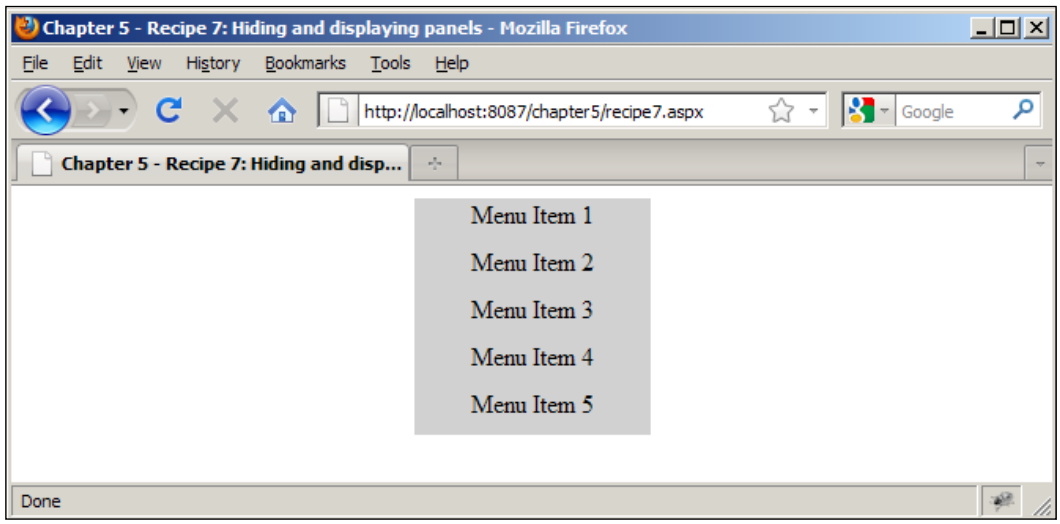
4. Define the css style of the sub-menu item as follows:

```
.content_body
    {
        display:none;
        width:150px;
        height:50px;
        background-color:#9999FF;
    }
```

Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<asp:Panel class="content_head" runat="server">Menu Item 1</asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 11
<br />Sub Menu Item 12</asp:Panel>
<asp:Panel class="content_head" runat="server">Menu Item 2</asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 21
<br />Sub Menu Item 22</asp:Panel>
<asp:Panel class="content_head" runat="server">Menu Item 3</asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 31
<br />Sub Menu Item 32</asp:Panel>
<asp:Panel class="content_head" runat="server">Menu Item 4</asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 41
<br />Sub Menu Item 42</asp:Panel>
<asp:Panel class="content_head" runat="server">Menu Item 5</asp:Panel>
<asp:Panel class="content_body" runat="server">Sub Menu Item 51
<br />Sub Menu Item 52</asp:Panel>
</div>
</form>
```

Thus, on page load, the web form displays only the main menu items as shown in the following screenshot:



We will now hide/display the sub-menu items on clicking the corresponding main menu items.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the callback function on the `click` event of the main menu item:

   ```
   $(".content_head").click(function(){
   ```

2. Use the jQuery `.next()` selector to retrieve the immediately next sub-menu item and apply the `slideToggle` function on the same:

   ```
   $(this).next(".content_body").slideToggle("slow");
   });
   ```

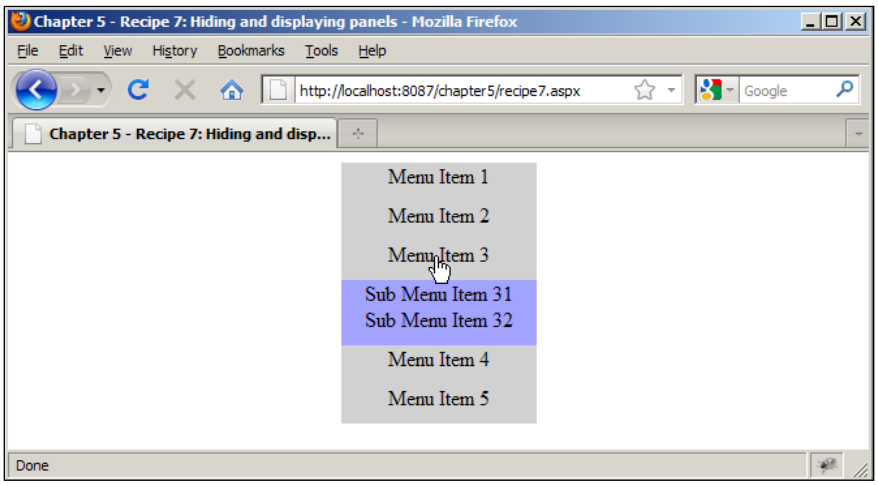
The `.next()` selector retrieves the immediately next matched element from the DOM tree.

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
             $(".content_head").click(function(){
                  $(this).next(".content_body").slideToggle("slow");
             });
        });
</script>
```

How it works...

Run the web page. Click on any main menu item. The corresponding sub-menu item will expand as follows:



On clicking the target main menu item again, the sub-menu collapses.

See also

Creating disappearing effect

Creating disappearing effect

In this recipe, we will use the animate method to animate the opacity of an ASP.NET panel control.

Getting ready

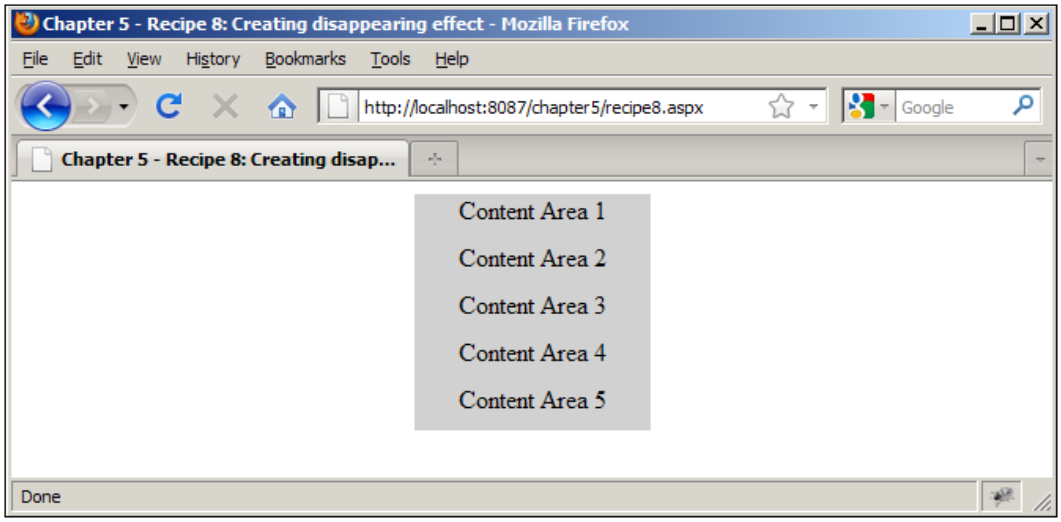
1. Create a new web form `Recipe8.aspx` in the current project.
2. Define a css class for the panel control:

```
.content_head
    {
        width:150px;
        height:30px;
        background-color:#CCCCCC;
        cursor:pointer;
    }
```

3. Add a couple of panel controls to the form as shown:

```
<form id="form1" runat="server">
<div align="center">
<asp:Panel class="content_head" runat="server">Content Area 1</
asp:Panel>
<asp:Panel class="content_head" runat="server">Content Area 2</
asp:Panel>
<asp:Panel class="content_head" runat="server">Content Area 3</
asp:Panel>
<asp:Panel class="content_head" runat="server">Content Area 4</
asp:Panel>
<asp:Panel class="content_head" runat="server">Content Area 5</
asp:Panel>
</div>
</form>
```

4. Thus, on page load, the web form displays the controls as captured in the following screenshot:



We will now use the animate function on the opacity of the panel controls to create a disappearing effect.

How to do it...

1. In the `document.ready()` function of the jQuery script block, use the jQuery css selector to retrieve the panels. Define the callback function on their `click` event:

```
$(".content_head").click(function(){
```

2. Apply the animate method on the matched elements as shown:

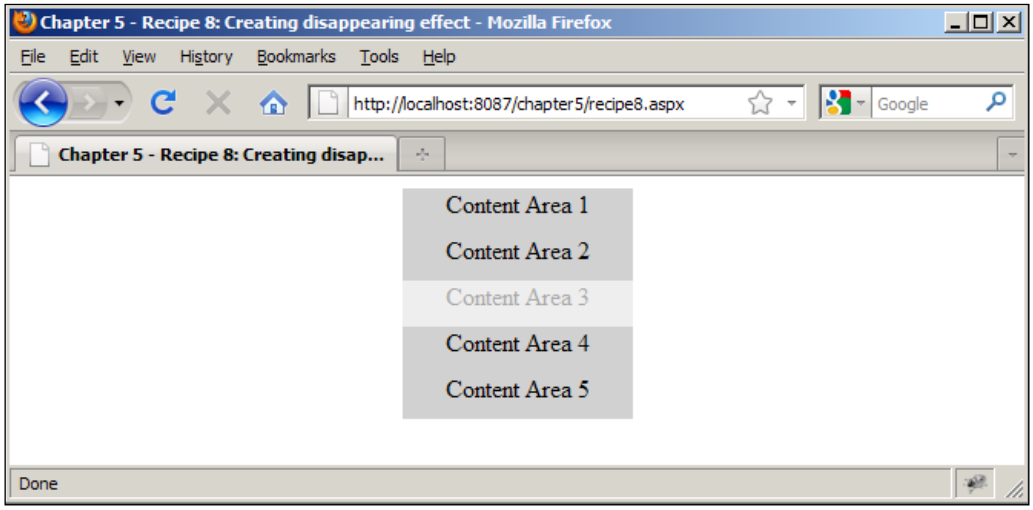
```
$(this).animate({opacity: "hide"},"slow");
            });
```

Thus, the complete jQuery solution is as follows:

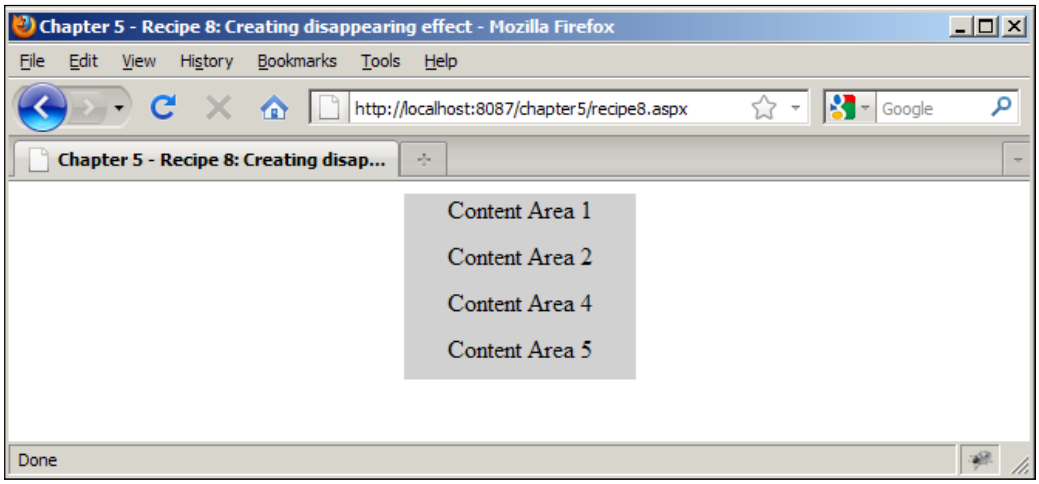
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
             $(".content_head").click(function(){
                  $(this).animate({opacity: "hide"},"slow");
             });
        });
</script>
```

How it works...

Run the web page. Click on any panel on the form. You will notice that the target area will slowly fade as captured in the following screenshot:



After 600 ms (`slow` animation effect), the target area will disappear completely as follows:



See also

Hiding and displaying panels

6
AJAX and ASP.NET (Part I)

In this chapter, we will cover:

- Setting up AJAX with ASP.NET using jQuery
- Using Firebug to view AJAX request/response
- Consuming page methods with AJAX
- Consuming web services with AJAX
- Populating ASP.NET DropDownList using AJAX
- Creating an auto complete search box

Introduction

AJAX (Asynchronous JavaScript and XML), a term coined by Jesse James Garrett of Adaptive Path, stands for a combination of different technologies that help to communicate seamlessly with the server without the need for a page refresh. AJAX applications involve the following technologies:

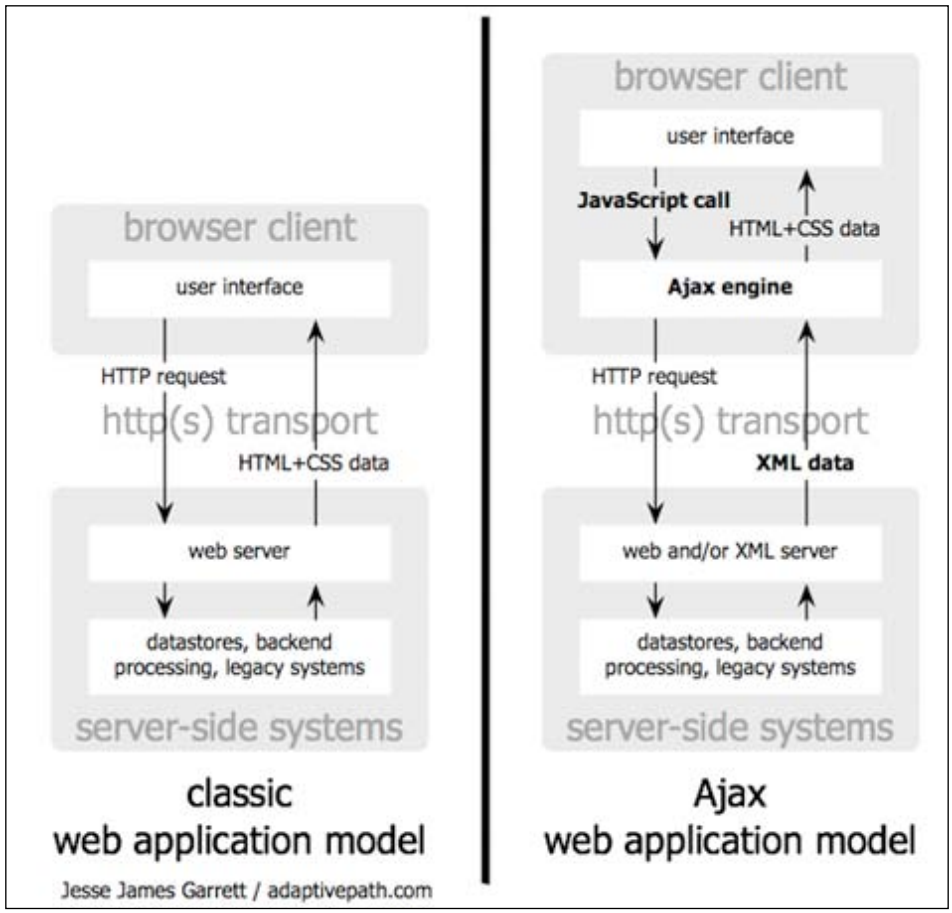
- JavaScript for running the core AJAX engine
- XmlHttpRequest object to communicate with the server
- Web presentation using XHTML and CSS or XSLT
- DOM to work with the HTML structure
- XML and JSON for data interchange

The XmlHttpRequest object is used for posting HTTP/HTTPS requests to the server. Most modern browsers have a built-in XmlHttpRequest object.

JSON (JavaScript Object Notation) is a lightweight data interchange format and is increasingly used in AJAX applications today. It is basically a collection of name/value pairs.

In classic web applications, the client submits data to the server for processing and the server sends back refreshed content to the client. This causes a visible page refresh and the web user must wait for a page reload before further interaction with the web application.

AJAX, however, eliminates the need for an explicit page refresh by communicating with the server behind the scenes. It uses the power of XmlHttpRequest object to post a request to the server. Thus, the backend communication with the server is transparent to the end user. In addition, using AJAX, only the data that is required to be updated can be selectively refreshed on the page.



The previous figure is the traditional model for web applications (left) compared to the AJAX model (right).

(The previous figure is from Jesse James Garrett, *AJAX: A New Approach to Web Applications*, `http://adaptivepath.com/ideas/essays/archives/000385.php`).

The previous figure shows the basic difference between traditional and AJAX-enabled applications. In traditional web applications, the client sends requests directly to the server and waits to receive the corresponding response. In AJAX-based applications, however, this is replaced by a JavaScript call to the AJAX engine instead, which sends the request asynchronously to the server. As a result, web users' interaction with the application is not interrupted and users can continue to work with the application.

The jQuery library provides many methods for working with AJAX. In this chapter, we will explore the use of the following methods:

- **$.ajax(settings)**: This is a generic low level function that helps to create any type of AJAX request. There are a number of configuration settings that can be applied using this function to customize an AJAX call. It helps to set the type of HTTP request (GET/POST), the URL, parameters, datatype, as well as the callback functions to execute successful/unsuccessful invocation of the AJAX call.
- **$.ajaxSetup(options)**: This method helps to define default settings for making AJAX calls on the page. The setup is done one time and all the subsequent AJAX calls on the page are made using the default settings.

Getting started

1. Let's start by creating a new ASP.NET website in Visual Studio and name it `Chapter6`.
2. Save the jQuery library in a script folder `js` in the project.
3. To enable jQuery on any web form, drag-and-drop the library to add the following to the page:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
```

Now let's move on to the recipes where we will see how jQuery can be used to make AJAX calls to ASP.NET code-behind. Basically, in this chapter, we will explore three possible approaches of communicating with the server:

- Using page methods
- Using web services
- Using HTTP Handlers

So, let's get started.

Setting up AJAX with ASP.NET using jQuery

In this recipe, we will set up our ASP.NET web page to execute an AJAX call. We will also see how to set the default AJAX properties.

Getting ready

1. Create an HTML file `Content.html` in the project and add the following contents:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Content Page</title>
    <link href="css/content.css" rel="stylesheet" type="text/css"
/>
</head>
<body>
<p>
<table cellpadding="3" cellspacing="3" id="box-table-a">
<tr><td>Title</td><td>Author</td><td>Genre</td></tr>
<tr><td>Alchemist</td><td>Paulo Coelho</td><td>Fiction</td></tr>
<tr><td>Heart of Darkness</td><td>Joseph Conrad</td><td>Classic</
td></tr>
<tr><td>David Copperfield</td><td>Charles Dickens</
td><td>Classic</td></tr>
<tr><td>Have a Little Faith</td><td>Mitch Albom</td><td>Fiction</
td></tr>
 </table>
 </p>
</body>
</html>
```

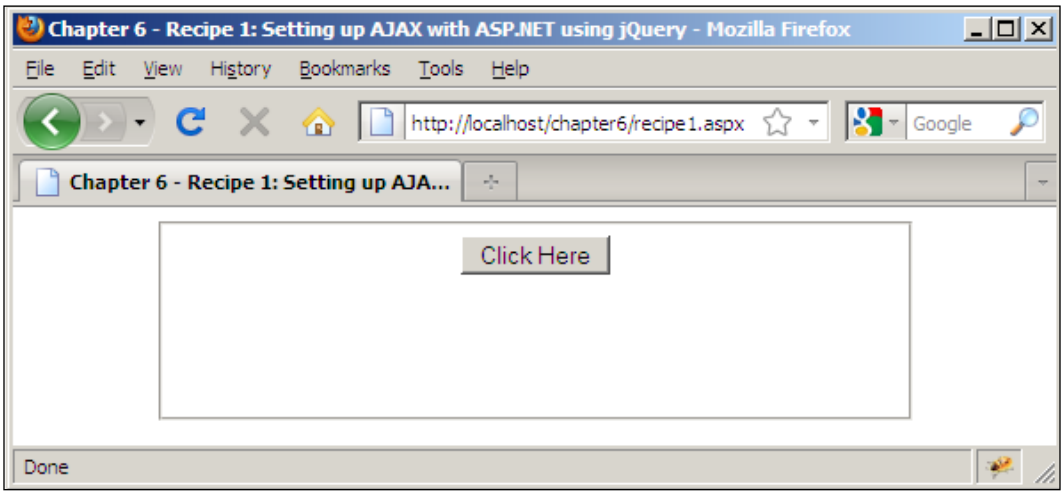
2. Add a new web form `Recipe1.aspx` to the current project.
3. Add a button control to the web form to trigger the AJAX request.

```
<asp:Button ID="btnSubmit" runat="server" Text="Click Here" />
```

4. Thus, the ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
     <div align="center">
    <fieldset style="width:400px;height:200px;">
    <div id="contentArea">
    <asp:Button ID="btnSubmit" runat="server" Text="Click Here" />
    </div>
    </fieldset>
    </div>
    </form>
```

5. On page load, the form will appear as shown in the following screenshot:



When the button is clicked, we will retrieve the contents of the HTML file using AJAX and display it on the page.

How to do it...

1. In the `document.ready()` function of the jQuery script block, call the `ajaxSetup()` method to set the default properties of all AJAX calls on the page:

```
$.ajaxSetup({
```

2. Turn off the cache so that the contents are not cached by the browser:

```
cache: false,
```

3. Set the data type of the response. In this case, since we are going to load the contents from the HTML file, the dataType is HTML:

```
dataType: "html",
```

4. Define the callback function if the AJAX request fails. The callback function takes in three parameters: the XMLHttpRequest object, the error status, and an exception object:

```
error: function(xhr, status, error) {
                    alert('An error occurred: ' + error);
                },
```

5. Define the global timeout in milliseconds:

```
timeout: 30000,
```

6. Set the type of HTTP request (GET/POST):

```
type: "GET",
```

7. Define the callback function to be called before the AJAX request is initiated. This function can be used to modify the Xml HttpRequest object:

```
beforeSend: function() {
                    console.log('In Ajax beforeSend function');
                },
```

8. Define the function to be called when the AJAX request is completed:

```
complete: function() {
                console.log('In Ajax complete function');
                }
            });
```

Now, having set the default properties in the previous code block, we will now invoke the actual AJAX call on clicking the button control on the form.

1. Attach a handler for the `click` event of the button control:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Initiate the AJAX call using the `.ajax()` method:

```
$.ajax({
```

4. Specify the URL to send the request to. In this case, we're sending the request to the HTML file:

```
url: "Content.htm",
```

5. Define the callback function for successful execution of the AJAX call:

```
success: function(data) {
```

The HTML response from the server is received in the `data` parameter in the preceding callback function.

6. Clear the contents of the containing div area to remove the button control and append the received HTML response:

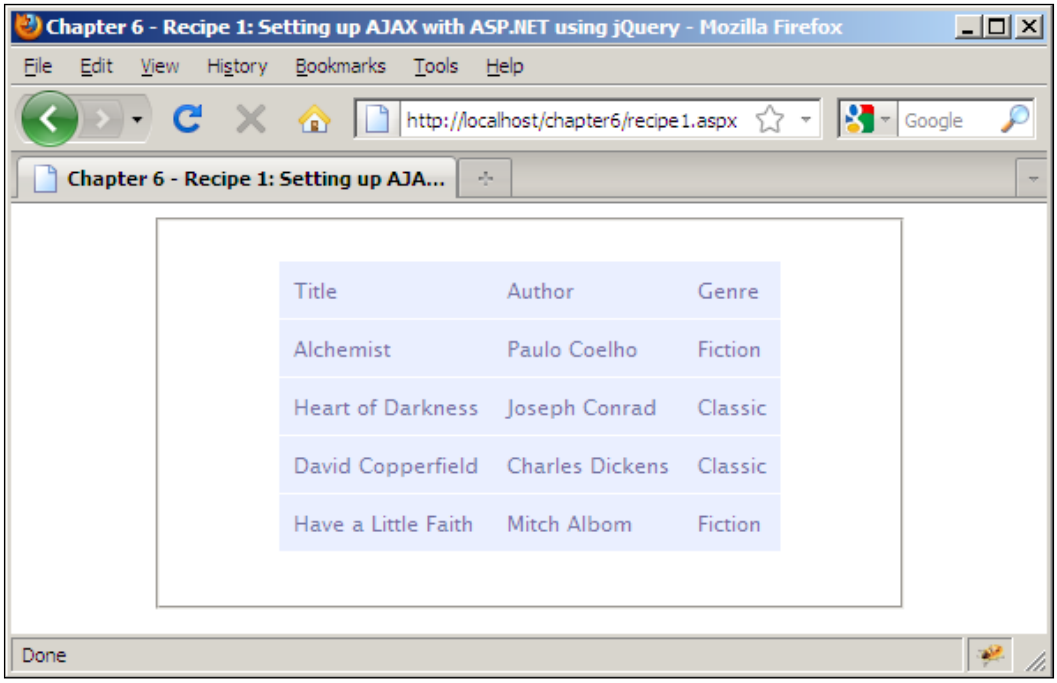
```
$("#contentArea").html("").append(data);
                }
            });
        });
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
       $(document).ready(function() {
           $.ajaxSetup({
               cache: false,
               dataType: "html",
               error: function(xhr, status, error) {
                   alert('An error occurred: ' + error);
               },
               timeout: 30000,
               type: "GET",
               beforeSend: function() {
                   console.log('In Ajax beforeSend function');
               },
               complete: function() {
               console.log('In Ajax complete function');
               }
           });

       $("#btnSubmit").click(function(e) {
           e.preventDefault();
           $.ajax({
               url: "Content.htm",
               success: function(data) {
               $("#contentArea").html("").append(data);
               }
           });
       });
   });
   </script>
```

How it works...

Run the web form. Click on the button to initiate the AJAX request. You will see that the page content is updated without any visible page refresh as follows:



See also

Using Firebug to view AJAX request/response

Using Firebug to view AJAX request/response

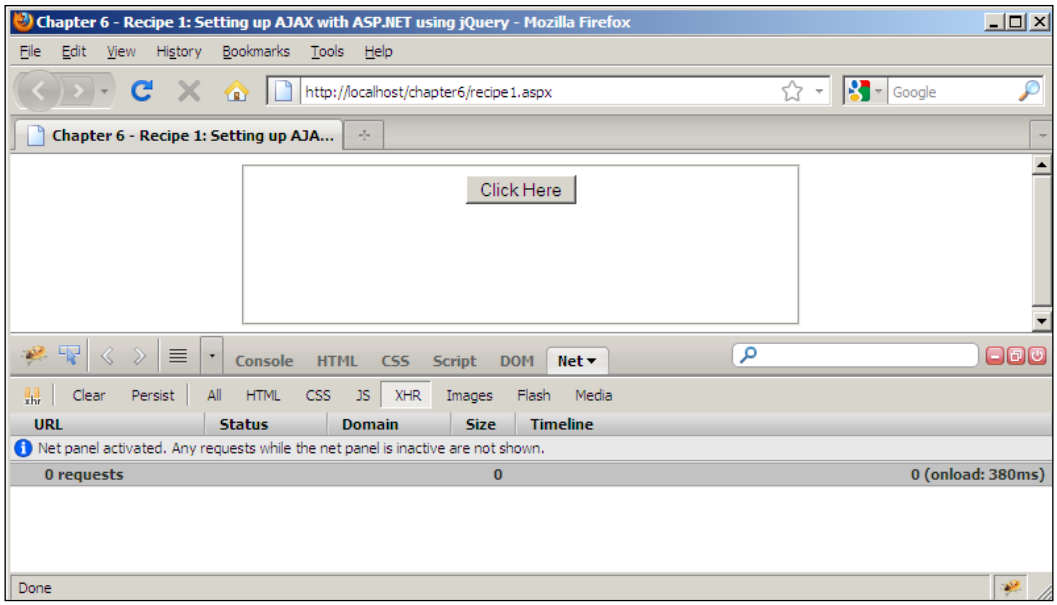
Most modern browsers provide Developer Tools for editing and debugging different types of content such as .html, .css, and JavaScript in real time.

In Internet Explorer 8 and above, the Developer Tools can be launched by pressing *F12*. In Chrome, you can access the same using **Tools | Developer Tools** or alternatively pressing **Control + Shift + I**.

Mozilla Firefox browser provides an add-on web development tool called Firebug. In this recipe, we will see how it can be used for XmlHttpRequest monitoring, thus enabling the logging of AJAX request/response for troubleshooting. We will demonstrate using the web form created in the *Setting up AJAX with ASP.NET using jQuery* recipe.

Getting ready

1. Download and install the latest version of Firebug from `http://getfirebug.com/`.
2. Run `Recipe1.aspx` in Mozilla Firefox browser. Launch Firebug from the Tools menu.
3. The page displays the Firebug window as follows:



Now, let's use Firebug step-by-step to trace the AJAX request/response.

How to do it...

1. Click on the button control on the web form to initiate the AJAX request. As a result, the page will retrieve the HTML contents and display in the div area of the form.
2. Click on the **Console** tab in the Firebug window. The window shows the console logging triggered by the callback functions defined in `beforeSend` and `complete`. It also displays the `GET` request initiated by the page as captured in the following screenshot:

Firebug supports the following console functions:

`console.log`: To log messages to the console at runtime

`console.debug`: To log debug messages with a hyperlink to the calling line

`console.error`: To log error messages

`console.info`: To log information messages

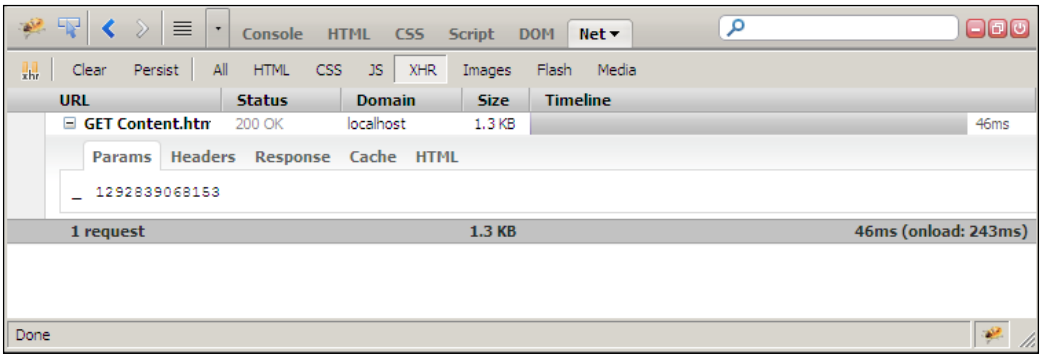
`console.warning`: To log warning messages

3. To view the different types of console logging, update the `complete` callback function in `ajaxSetup()` method as follows:

```
complete: function() {
     console.log('In Ajax complete function - Log Message');
     console.debug('In Ajax complete function - Debug Message');
     console.error('In Ajax complete function - Error Message');
     console.info('In Ajax complete function - Info Message');
     console.warn('In Ajax complete function - Warning Message');
  }
```

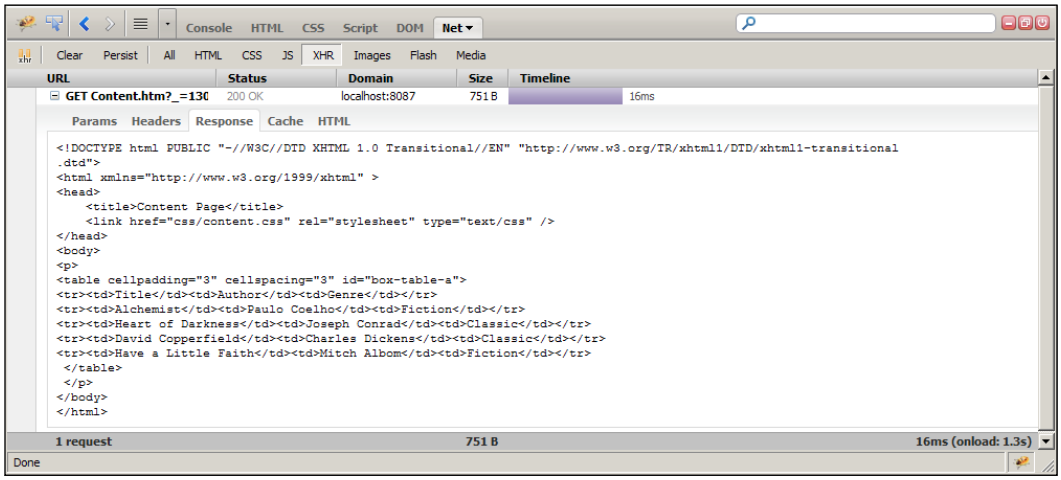
Rerun the page. The logging will appear in the **Console** tab as follows:

4. Next, click the **Net** tab in the Firebug window. This is the place where we can monitor the network activity triggered by the page. The traffic is segregated into various types such as `HTML`, `CSS`, `JS`, `XHR`, `Images`, `Flash`, and `Media`.
5. Click the `XHR` sub-tab to monitor the XmlHttpRequest object. The window displays the request URL, response status, response size, and execution time. It also contains the following sub-tabs:
    - **Params**: Displays name/value pairs in the request.
    - **Headers**: Displays the request/response headers.
    - **Response**: Displays the AJAX response as received by the browser. The response can be either JSON, XML, HTML, text, or script.
    - **Cache**: Displays the cache details.
    - **HTML**: Displays the HTML response.

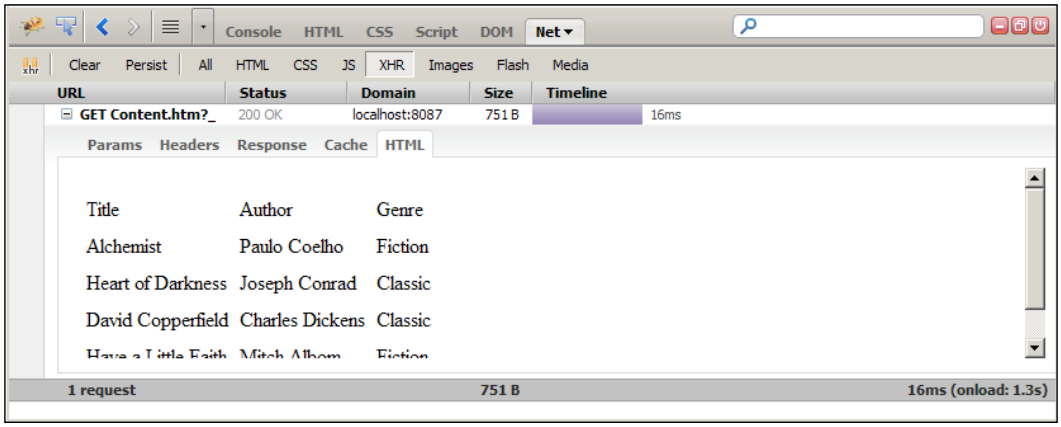


Notice that the `Params` sub-tab contains a string with a _ prefix. This is the mechanism used by jQuery to control caching. For every request, jQuery generates a new string to ensure that the browser considers the request as a new one and does not cache the response.

6. Next, click the Response sub-tab. This screen will display the AJAX response as received by the page. The format can be either JSON, XML, HTML, text, or script. In our case, the response is in HTML format as shown:



7. Next, click on the HTML sub-tab. Firebug will display the AJAX response in HTML format as follows:



See also

Consuming page methods with AJAX

Consuming page methods with AJAX

There are two primary ways of using jQuery to communicate with ASP.NET code-behind based on events triggered on a web page:

- using page methods
- using web services

Page methods provide a handy way of invoking code-behind from client side script when simple server-side processing is required and when the application scope does not necessitate the calling of web services.

In this recipe, we will use AJAX to call a page method to check if the user name entered by the end user is valid.

Getting ready

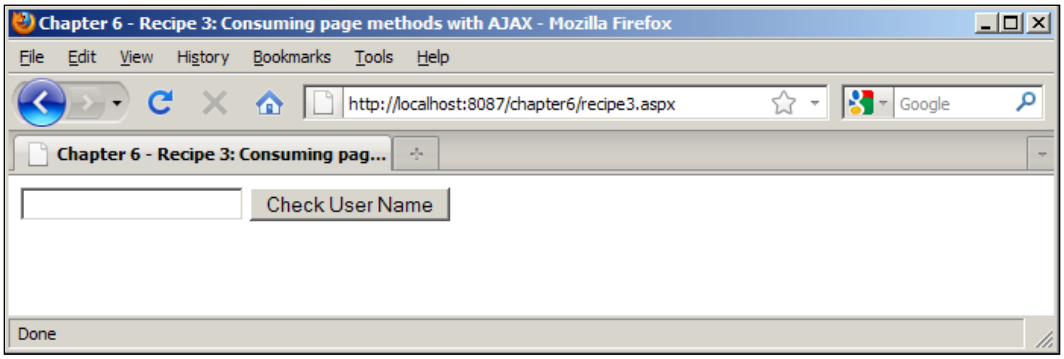
1. Add a new web form `Recipe3.aspx` to the current project.
2. In the code-behind, add the following namespace to enable page methods:
   ```
   using System.Web.Services;
   ```
3. Define an internal array object to store some UserNames in the system:
   ```
   public static string[] UserNameArray;
   ```
4. Populate the array in the `Page_Load` method as follows:
   ```
   protected void Page_Load(object sender, EventArgs e)
       {
           UserNameArray = new string[7] { "testid01", "testid02",
   "testid03", "testid04", "testid05", "testid06", "testid07" };
       }
   ```
5. Write a page method with a single input parameter `sUserName`. The page method will check if the input user name exists in the system and will return `false` if the user name is not found in the system.
   ```
   [WebMethod()]
       public static bool CheckUserName(string sUserName)
       {
           foreach (string strUsername in UserNameArray){
               if (sUserName == strUsername)
                   return true;
           }

           return false;
       }
   ```

Note that the page method is defined with a label `[WebMethod()]`. Also, to be able to call the method directly, we must define it as `static`.

6. On the web form, add a TextBox control to enter the username. Also add a button control to trigger the AJAX request as follows:

```
<form id="form1" runat="server">
    <div>
        <asp:TextBox ID="txtUserName" runat="server"></
asp:TextBox>
        <asp:Button ID="btnSubmit" runat="server" Text="Check User
Name" />
</div>
    </form>
```

7. Thus, on page load, the web form will appear as shown below:



We will now trigger an AJAX call to the page method on clicking the button control.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the button control:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Check if the TextBox is empty and alert the user:

```
if ($("#txtUserName").val() == '')
          alert("Please enter the UserName");
```

4. If the TextBox is not empty, call the `sendData()` function and pass the input data as a parameter to the function:

```
else
    sendData($("#txtUserName").val());
            });
```

5. Now we will define the `sendData()` function that takes in a single parameter i.e. `sUserName`):

```
function sendData(sUserName) {
```

6. Get the base path of the page:

```
var loc = window.location.href;
loc = (loc.substr(loc.length - 1, 1) == "/") ? loc + "Recipe3.
aspx" : loc;
```

7. Call the `.ajax()` method:

```
$.ajax({
```

8. Specify the request type as HTTP POST:

```
type: "POST",
```

9. Specify the complete URL of the page method to which the request needs to be sent:

```
url: loc + "/CheckUserName",
```

10. Pass the input parameter `sUserName` in `.json` format:

```
data: '{"sUserName":"' + sUserName + '"}',
```

Note that JSON serialised data consists of name/value pairs in the format `{"name1" : "value1", "name2" : "value2", "name3" : "value3"}`.

11. Specify the content type of the request/response:

```
contentType: "application/json; charset=utf-8",
```

12. Specify the response data type:

```
dataType: "json",
```

13. Define the callback function for successful invocation:

```
success: function(msg) {
                        if (msg.d)
                            alert("The User Name is valid");
                        else
                            alert("The User Name is invalid")
                    },
```

Note that the AJAX response is wrapped in a `d` object, e.g. `{"d":true}`. Hence, to access the Boolean response from the page method, we use `msg.d`.

14. Specify the callback function in case of failure:

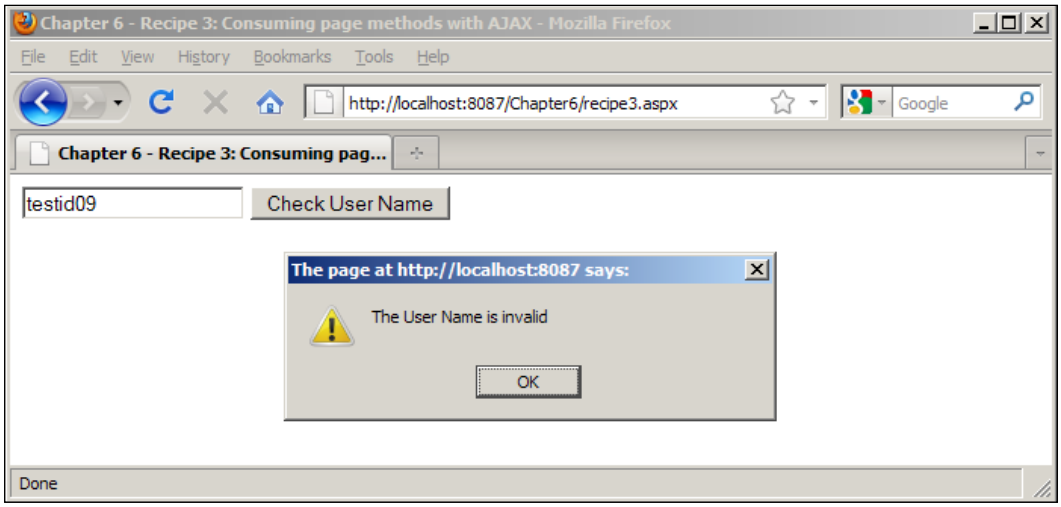
```
error: function() {
                        alert("An unexpected error has occurred
during processing.");
                    }
                });
            }
```

Thus, the complete jQuery solution is as follows:

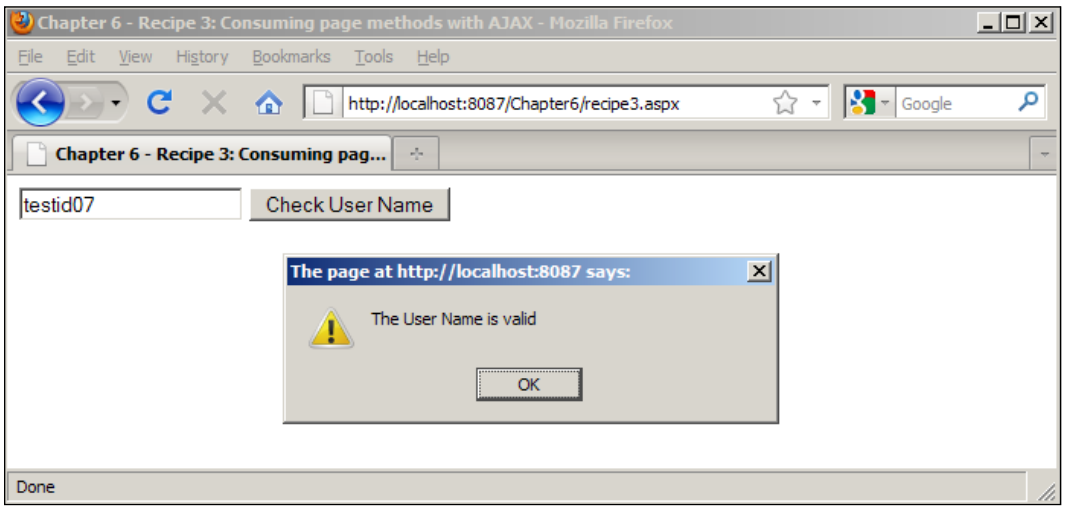
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                if ($("#txtUserName").val() == '')
                    alert("Please enter the UserName");
                else
                    sendData($("#txtUserName").val());
            });

            function sendData(sUserName) {
                var loc = window.location.href;
                loc = (loc.substr(loc.length - 1, 1) == "/") ? loc +
                "Recipe3.aspx" : loc;
                $.ajax({
               type: "POST",
               url: loc + "/CheckUserName",
               data: '{"sUserName":"' + sUserName + '"}',
               contentType: "application/json; charset=utf-8",
               dataType: "json",
               success: function(msg) {
                        if (msg.d)
                            alert("The User Name is valid");
                        else
                            alert("The User Name is invalid")
                    },
               error: function() {
 alert("An unexpected error has occurred during processing.");
                    }
            });
      }

   });
</script>
```

How it works...

Run the web page. Enter any random data in the TextBox and click the button control. The page sends an AJAX request to the page method. On receiving a response, it throws an error as shown below:



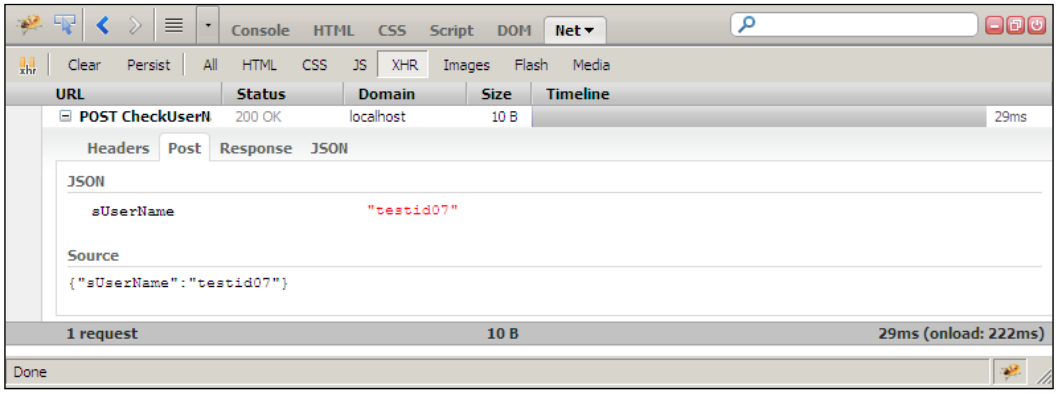
On entering a username that exists in the system, the page displays the following message:



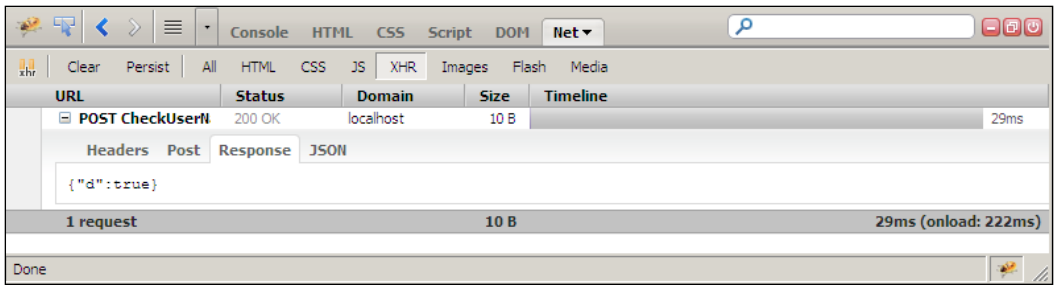
There's more...

Let's inspect the AJAX request/response using Firebug. Launch Firebug in Mozilla Firefox and submit a username for verification.

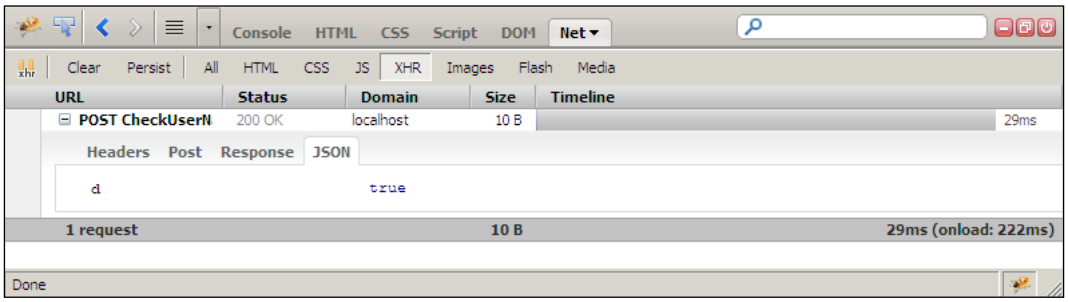
The Firebug window will log the JSON request as follows:



The corresponding response received from the page method is logged in the `Response` sub-tab as shown next:



Switch to the `JSON` sub-tab to view the JSON deserialised response:



See also

Consuming web services using AJAX

Consuming web services with AJAX

In addition to page methods, jQuery also enables the making of AJAX calls to web services. In this recipe, we will validate the username keyed in by the end user by making a call to a web service instead.

Getting ready

1. Add a web service `UserNameWS.asmx` to the current project.
2. Add the following namespace to the code-behind:

```
using System.Web.Services;
```

3. To enable the web service to be called using AJAX, add the following:

```
[System.Web.Script.Services.ScriptService]
```

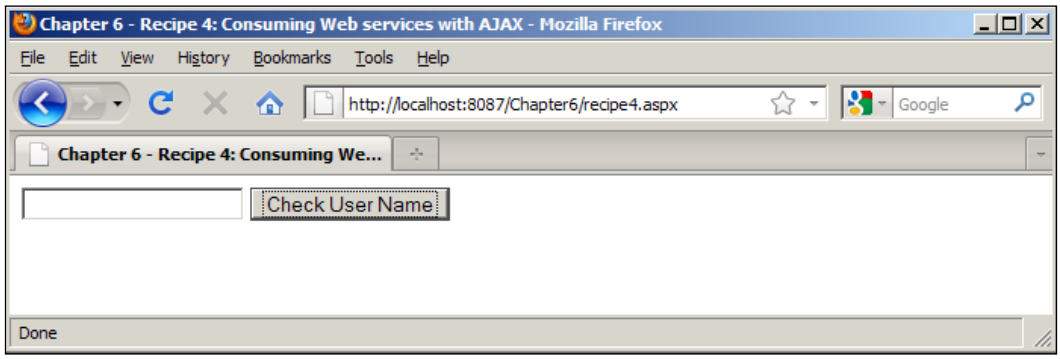
4. Add a web method to validate the username as follows:

```
[WebMethod]
    public bool CheckUserName(string sUserName)
    {
        string[] UserNameArray;
        UserNameArray = new string[7] { "testid01", "testid02",
"testid03", "testid04", "testid05", "testid06", "testid07" };

        foreach (string strUsername in UserNameArray)
        {
            if (sUserName == strUsername)
                return true;
        }

        return false;
    }
```

5. Add a new web form `Recipe4.aspx` to the current project.
6. Add a TextBox and a Button control to the form as follows:

```
<form id="form1" runat="server">
    <div>
        <asp:TextBox ID="txtUserName" runat="server"></
asp:TextBox>
        <asp:Button ID="btnSubmit" runat="server" Text="Check User
Name" />
</div>
    </form>
```

7. Thus, the page appears as shown:



We will now invoke the web service using AJAX when the button is clicked.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the button control:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Check if the TextBox is empty and alert the user:

```
if ($("#txtUserName").val() == '')
          alert("Please enter the UserName");
```

4. If the TextBox is not empty, call the `sendData()` function and pass the input data as a parameter to the function:

```
else
    sendData($("#txtUserName").val());
            });
```

5. Now we will define the `sendData()` function that takes in a single parameter i.e. `sUserName`):

```
function sendData(sUserName) {
```

6. Call the `.ajax()` method:

```
$.ajax({
```

7. Specify the request type as HTTP POST:

```
type: "POST",
```

8. Specify the web service and the corresponding web method to call:

```
url: "UserNameWS.asmx/CheckUserName",
```

9. Pass the input parameter `sUserName` in JSON format:

```
data: '{"sUserName":"' + sUserName + '"}',
```

Note that JSON serialised data consists of name/value pairs in the format `{"name1" : "value1", "name2" : "value2", "name3" : "value3"}`.

10. Specify the content type of the request/response:

```
contentType: "application/json; charset=utf-8",
```

11. Specify the response data type:

```
dataType: "json",
```

12. Define the callback function for successful invocation:

```
success: function(msg) {
                        if (msg.d)
                            alert("The User Name is valid");
                        else
                            alert("The User Name is invalid")
                    },
```

Note that the AJAX response is wrapped in a `d` object, e.g. `{"d":true}`. Hence, to access the Boolean response from the page method, we use `msg.d`.

13. Specify the callback function in case of failure:

```
error: function() {
       alert("An unexpected error has occurred during
processing.");
                    }
                });
            }
```

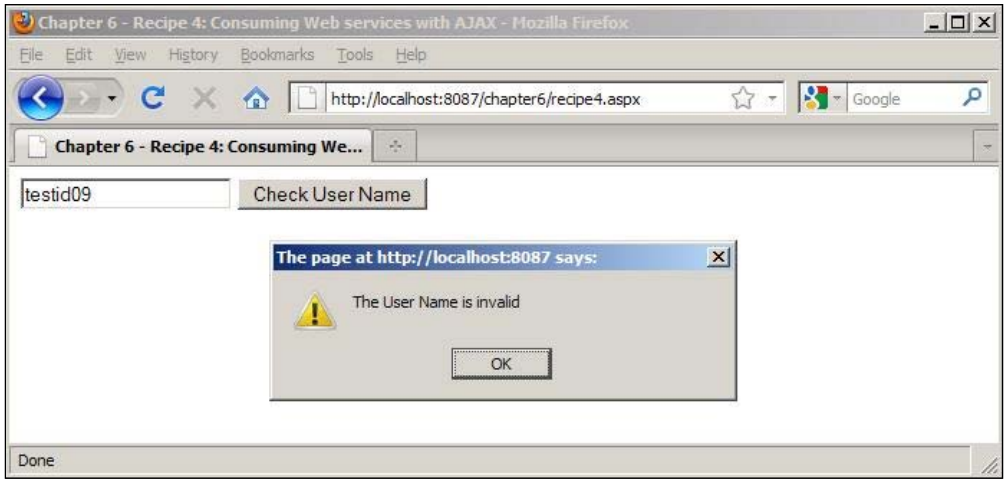
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                if ($("#txtUserName").val() == '')
                    alert("Please enter the UserName");
```

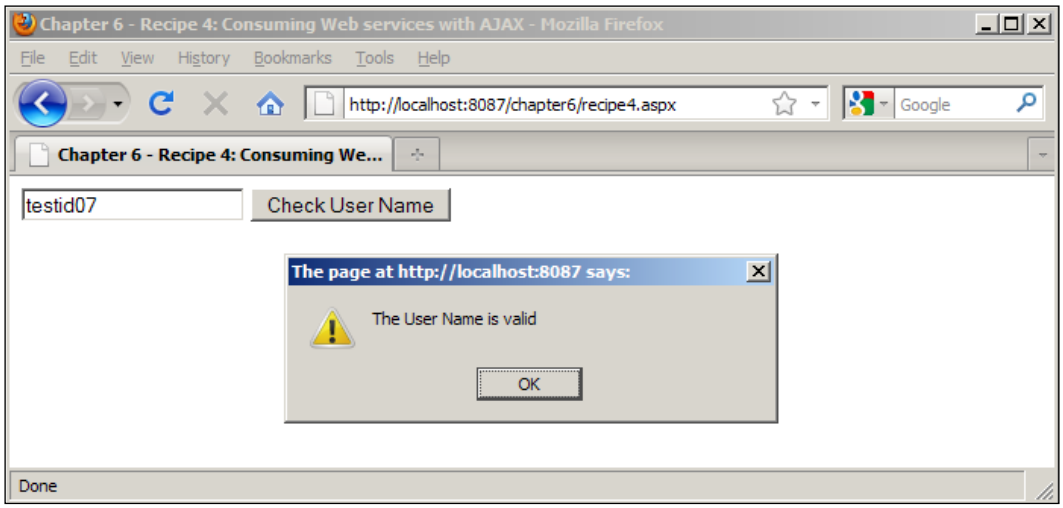
```
                else
                    sendData($("#txtUserName").val());
            });
            function sendData(sUserName) {
                 $.ajax({
                    type: "POST",
                    url: "UserNameWS.asmx/CheckUserName",
                    data: '{"sUserName":"' + sUserName + '"}',
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function(msg) {
                        if (msg.d)
                            alert("The User Name is valid");
                        else
                            alert("The User Name is invalid")
                    },
                    error: function() {
                        alert("An unexpected error has occurred during
processing.");
                    }
                });
            }

        });
    </script>
```

How it works...

Run the web page. Enter any random text in the input field and click the button control. The page will make an AJAX call to the web service. On receiving the response, it displays the following message:



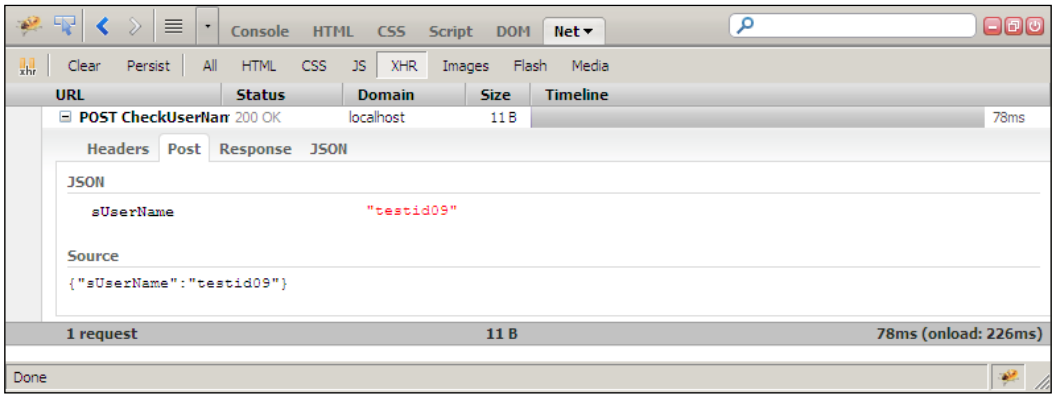
On keying in a username that is valid, the page displays the following:



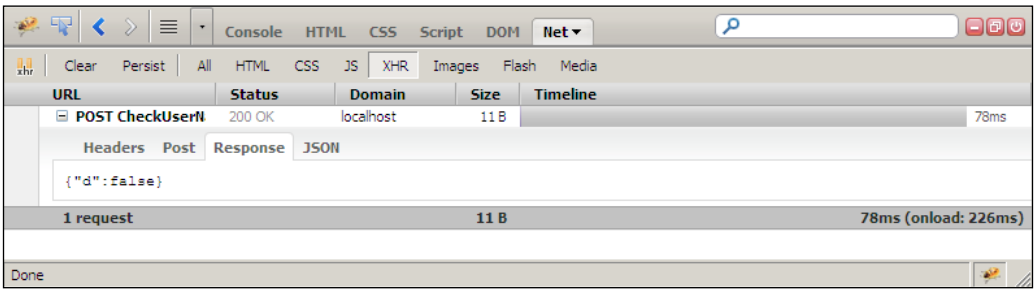
There's more...

Let's inspect the AJAX communication using Firebug in Mozilla Firefox. Run Firebug from the **Tools** menu. Load the web page and click the **Net** tab.

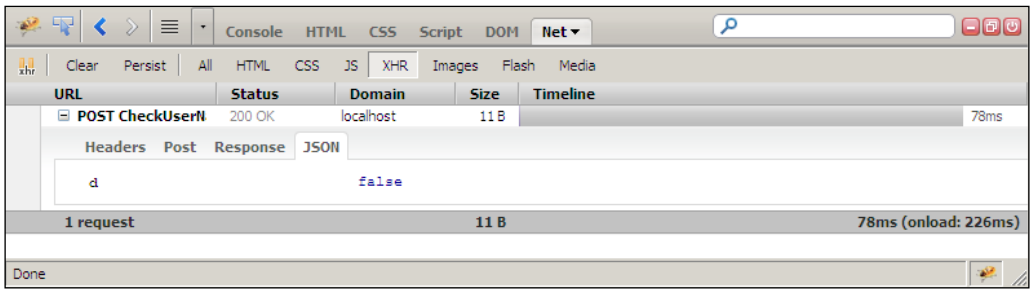
Firebug logs the JSON request as shown in the following screen:



The corresponding response is captured in the `Response` sub-tab as follows:



On clicking the JSON sub-menu, we can view the JSON deserialised output as seen in the following screenshot:



See also

Consuming page methods using AJAX

Populating ASP.NET DropDownList using AJAX

In this recipe, we will create a cascading dropdown. We will populate the values in the second DropDownList based on the selected value in the first DropDownList. We will make an AJAX call to a page method to retrieve the contents of the second DropDownList.

Getting ready

1. Add a new web form `Recipe5.aspx` to the current project.
2. Add the following DropDownList to the form:

```
<asp:DropDownList ID="ddlUserID" runat="server">
<asp:ListItem Text="Please select" Value="[-]"></asp:ListItem>
```

```
<asp:ListItem Text="UserID1" Value="UserID1"></asp:ListItem>
<asp:ListItem Text="UserID2" Value="UserID2"></asp:ListItem>
<asp:ListItem Text="UserID3" Value="UserID3"></asp:ListItem>
</asp:DropDownList>
```

3. Add another empty DropDownList to the form:

```
<asp:DropDownList ID="ddlModuleList" CssClass="hidden"
runat="server">
</asp:DropDownList>
```

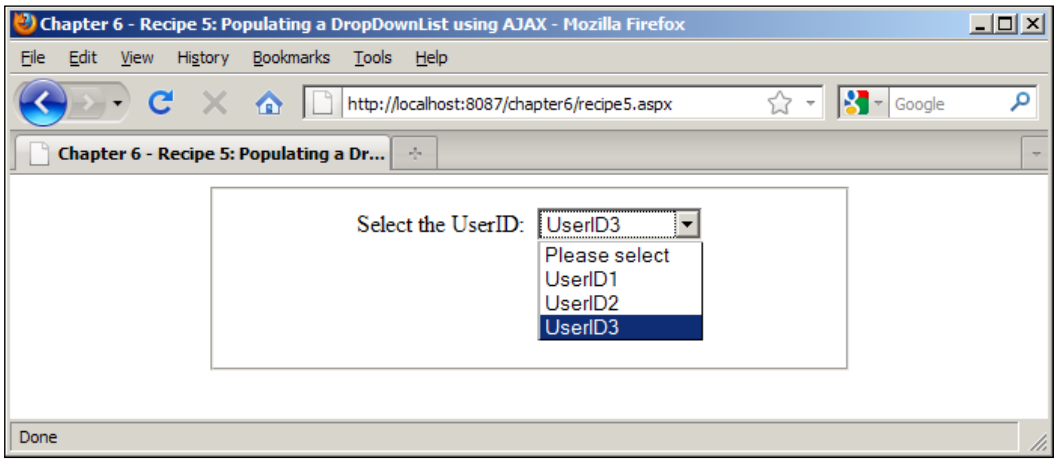
4. The css style for the second DropDownList is defined as follows:

```
.hidden
    {
     display:none;
    }
```

Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:400px;height:100px;">
<table cellpadding="3" cellspacing="3" border="0">
<tr><td>Select the UserID: </td>
<td align="left">
<asp:DropDownList ID="ddlUserID" runat="server">
<asp:ListItem Text="Please select" Value="[-]"></asp:ListItem>
<asp:ListItem Text="UserID1" Value="UserID1"></asp:ListItem>
<asp:ListItem Text="UserID2" Value="UserID2"></asp:ListItem>
<asp:ListItem Text="UserID3" Value="UserID3"></asp:ListItem>
</asp:DropDownList>
</td></tr>
<tr><td>
<label id="lblModuleList" class="hidden" runat="server">List of
Modules available: </label>
</td>
<td align="left">
<asp:DropDownList ID="ddlModuleList" CssClass="hidden" runat="server">
</asp:DropDownList>
</td></tr>
</table>
</fieldset>
</div>
</form>
```

On page load, the web form appears as follows:



1. In the code-behind of the web form, add the following namespace to enable page methods:

   ```
   using System.Web.Services;
   ```

2. Create a `static` page method that returns an ArrayList object consisting of Text/Value pairs. The page method takes in a single parameter i.e. `sUserID`):

   ```
   [WebMethod()]
       public static ArrayList GetModuleList(string sUserID)
       {
           ArrayList listArr = new ArrayList();

           if (sUserID == "UserID1")
           {
               listArr.Add(new ListItem("Dashboard", "Dashboard"));
               listArr.Add(new ListItem("Edit Profile", "Edit
   Profile"));
               listArr.Add(new ListItem("Change Password", "Change
   Password"));
           }
           else if (sUserID == "UserID2")
           {
               listArr.Add(new ListItem("Dashboard", "Dashboard"));
               listArr.Add(new ListItem("Account Settings", "Account
   Settings"));
               listArr.Add(new ListItem("Privacy Settings", "Privacy
   Settings"));
           }
   ```

```
        else if (sUserID == "UserID3")
        {
            listArr.Add(new ListItem("Dashboard", "Dashboard"));
            listArr.Add(new ListItem("Redeem Points", "Redeem
Points"));
            listArr.Add(new ListItem("Contact Us", "Contact Us"));
        }

        return listArr;
    }
```

When the end user selects a UserID from the first DropDownList, we will trigger an AJAX call to the page method and pass the selected UserID as a parameter.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define an event handler for the `change` event of the first DropDownList:

   ```
   $("#ddlUserID").bind('change', function(e) {
   ```

2. Check if the selected value is empty:

   ```
   if ($("#ddlUserID").val() != '[-]')
   ```

3. If a valid item is selected then call the `sendData()` function. Pass the selected value of the DropDownList as a parameter to the function:

   ```
   sendData($("#ddlUserID").val());
   ```

4. If no value is selected from the DropDownList then hide the second DropDownList and the corresponding label control:

   ```
   else {
           $("#lblModuleList").addClass("hidden");
           $("#ddlModuleList").addClass("hidden");
       }
    });
   ```

5. Now let's define the `sendData()` function:

   ```
   function sendData(sUserID) {
   ```

6. Get the base path of the current page:

   ```
   var loc = window.location.href;
   loc = (loc.substr(loc.length - 1, 1) == "/") ? loc + "Recipe5.
   aspx" : loc;
   ```

7. Invoke the `.ajax()` method:

   ```
   $.ajax({
   ```

8. Set the HTTP request type to POST:

```
type: "POST",
```

9. Specify the page method to send the AJAX request to:

```
url: loc + "/GetModuleList",
```

10. Pass the selected UserID from the first DropDownList to the page method in JSON format:

```
data: '{"sUserID":"' + sUserID + '"}',
```

11. Define the content type of the data transfer:

```
contentType: "application/json; charset=utf-8",
```

12. Specify the response data type:

```
dataType: "json",
```

13. Define the callback function on successful invocation of the page method where the AJAX response is received in the object `msg`. This is the ArrayList object consisting of Text/Value pairs:

```
success: function(msg) {
```

14. Unhide the second DropDownList and the corresponding label control:

```
$("#lblModuleList").removeClass("hidden");
$("#ddlModuleList").removeClass("hidden");
```

15. Clear the second DropDownList and add a default item as follows:

```
$("#ddlModuleList").empty().append($("<option></option>").val("[-
]").html("Please select"));
```

The DropDownList can be cleared by using `.empty()`. Thereafter, we chain the `.append()` function to add the `<option>` tags for the ListItem and set its Text using `.html()` and Value using `.val()`.

16. For each ListItem in the response object `msg`, add the Text/Value pairs to the second DropDownList:

```
$.each(msg.d, function() {
                          $("#ddlModuleList").append($("<option></
option>").val(this['Value']).html(this['Text']));
                        });
},
```

`this['Text']` returns the Text of the current ListItem whereas `this['Value']` returns the corresponding Value.

17. Define the callback function in case of failure:

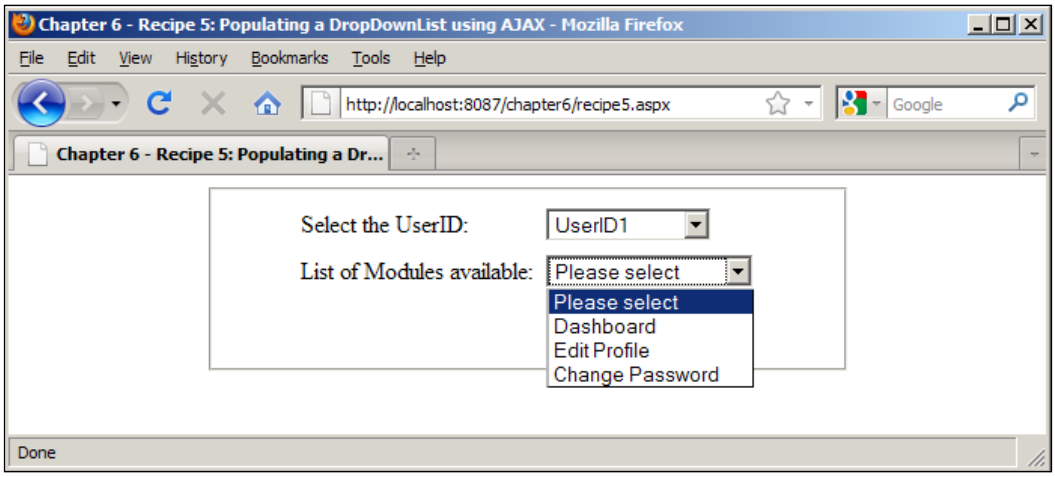
```
error: function() {
                        alert("An unexpected error has occurred
during processing.");
                    }
                });
            }
```

Thus, the complete jQuery solution is as follows:

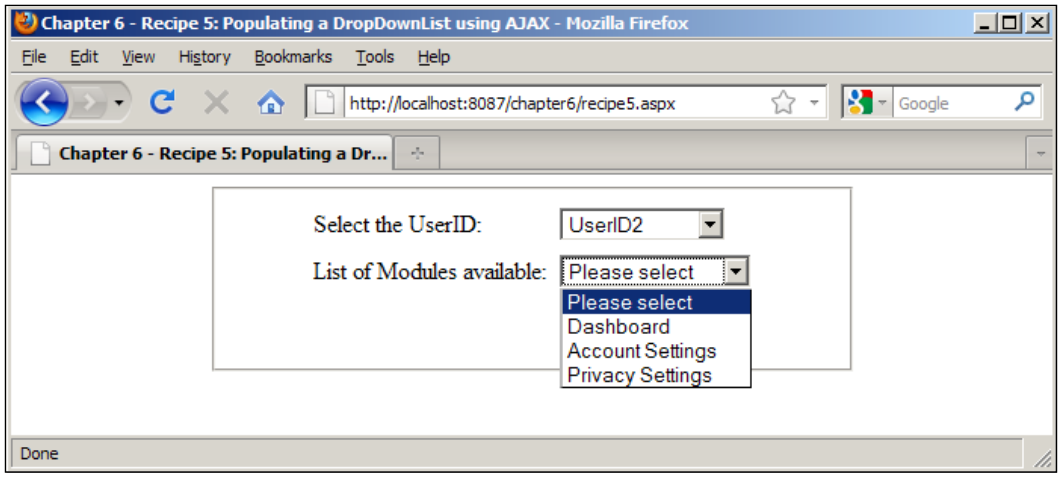
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#ddlUserID").bind('change', function(e) {
                if ($("#ddlUserID").val() != '[-]')
                    sendData($("#ddlUserID").val());
                else {
                    $("#lblModuleList").addClass("hidden");
                    $("#ddlModuleList").addClass("hidden");
                }
            });

            function sendData(sUserID) {
                var loc = window.location.href;
                loc = (loc.substr(loc.length - 1, 1) == "/") ? loc +
"Recipe5.aspx" : loc;
                $.ajax({
                    type: "POST",
                    url: loc + "/GetModuleList",
                    data: '{"sUserID":"' + sUserID + '"}',
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    success: function(msg) {
                        $("#lblModuleList").removeClass("hidden");
                        $("#ddlModuleList").removeClass("hidden");
                        $("#ddlModuleList").empty().
append($("<option></option>").val("[-]").html("Please select"));
                        $.each(msg.d, function() {
                            $("#ddlModuleList").append($("<option></
option>").val(this['Value']).html(this['Text']));
                        });
                    },
                    error: function() {
                        alert("An unexpected error has occurred during
processing.");
                    }
                });
            }
        });
    </script>
```

How it works...

Run the web page. Initially, the second DropDownList is hidden. Now select any UserID from the first DropDownList. The second DropDownList will be populated as shown:



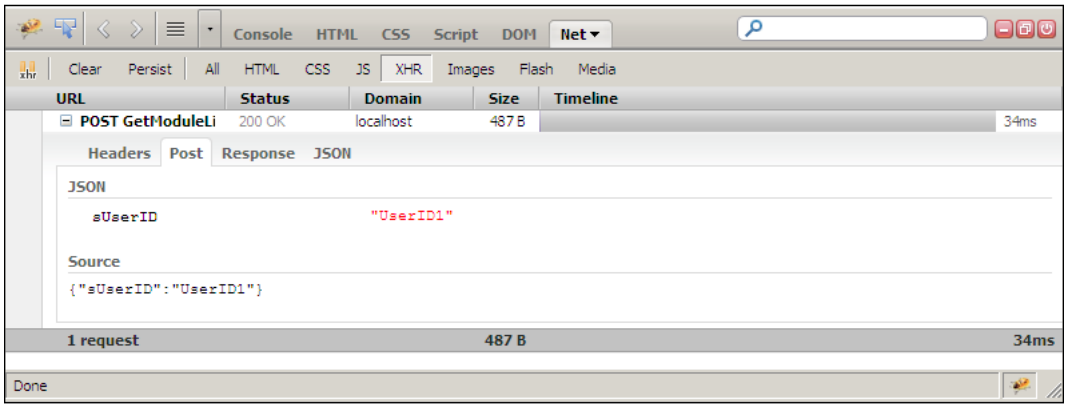
Now select a different UserID from the first DropDownList. You will notice that the contents of the second DropDownList will be updated as shown next:



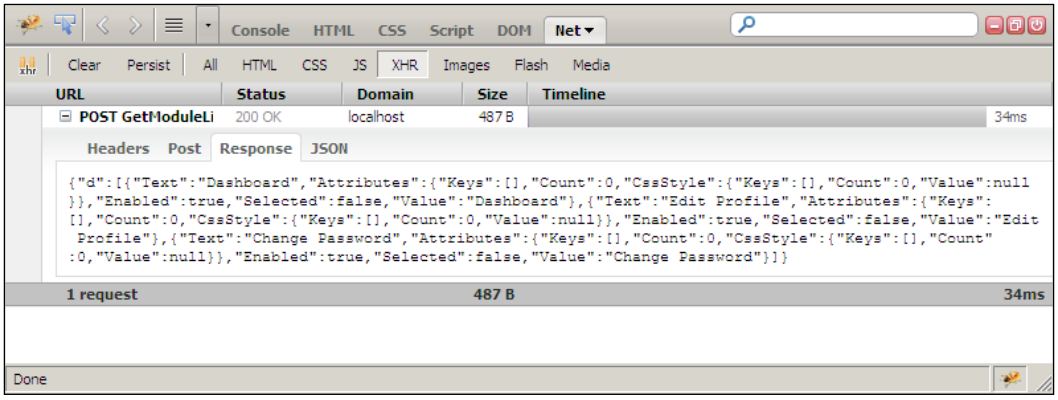
There's more...

Let's take a look at the AJAX request/response generated by the page. Launch Firebug in Mozilla Firefox and load the ASPX page.

Under the **Net** tab, click on **XHR** and then on the **Post** sub-tab. Firebug will display the JSON request as shown:



The corresponding JSON response returned to the calling script is as follows:



Note that each item returned in the ArrayList contains a `Text` and a corresponding `Value` field.

See also

Creating an auto complete search box

Creating an auto complete search box

With the widespread use of AJAX in web programming, many online search applications now provide auto complete search boxes. The retrieval of the matching search keywords is done by posting AJAX calls to server side code that links to a backend database.

In this recipe, we will use the auto complete widget provided by jQuery UI to demonstrate an auto complete search box. We will post the AJAX request to an ASP.NET HTTP handler for the retrieval of the matching search keywords.

Getting ready

1. Add a HTTP handler `SearchKeys.ashx` to the project.
2. Add the following namespaces to the code-behind:

```
using System.Collections.Generic;
using System.Web.Script.Serialization;
```

3. The HTTP Handler implements `IHttpHandler` and contains two primary functions:
   - `ProcessRequest` that takes in a single `HttpContext` parameter
   - `IsResuable` that returns a Boolean value
4. Define the `IsReusable()` function as follows:

```
public bool IsReusable {
        get {
            return false;
        }
    }
```

5. Create a function that will return a list of matching search keywords from an internal collection based on the target keyword:

```
public static string[] GetFilteredList(string keyword)
    {
        List<string> countryList = new List<string> { "Australia",
"Indonesia", "Philippines", "New Zealand", "Singapore",
"Thailand", "Korea", "Vietnam" };
       List<string> filteredList = new List<string>();

       foreach (string keyval in countryList)
       {
          if (keyval.ToUpper().Contains(keyword.ToUpper()))
              filteredList.Add(keyval);
       }

       return filteredList.ToArray();
    }
```

6. Assuming the target keyword is passed as a QueryString parameter to the handler, it can be retrieved using the following:

```
string keyword;
keyword = context.Request.QueryString["keyword"];
```

7. Thus, the `ProcessRequest()` function can be defined as follows:

```
public void ProcessRequest (HttpContext context)
    {
        string keyword;
        keyword = context.Request.QueryString["keyword"];
        if (keyword != null)
        {
            JavaScriptSerializer serializer = new
JavaScriptSerializer();
            string jsonString = serializer.Serialize(GetFilteredLi
st(keyword));
            context.Response.Write(jsonString);
        }
   }
```

Note that we have used a Serialisation function to format the output list in JSON. The response is sent back to the client using `Context.Response.Write`.

8. Next, we will add a new web form `Recipe6.aspx` to the project.
9. Download jQuery UI library from `http://jqueryui.com/download`. You can customise the download to include the following widgets:
   - Core
   - Widget
   - Position
   - Autocomplete
10. Save the downloaded script files in the local folder `js`. Save the style sheets in the local folder `css`.
11. Include the previous files on the page along with the jQuery library:

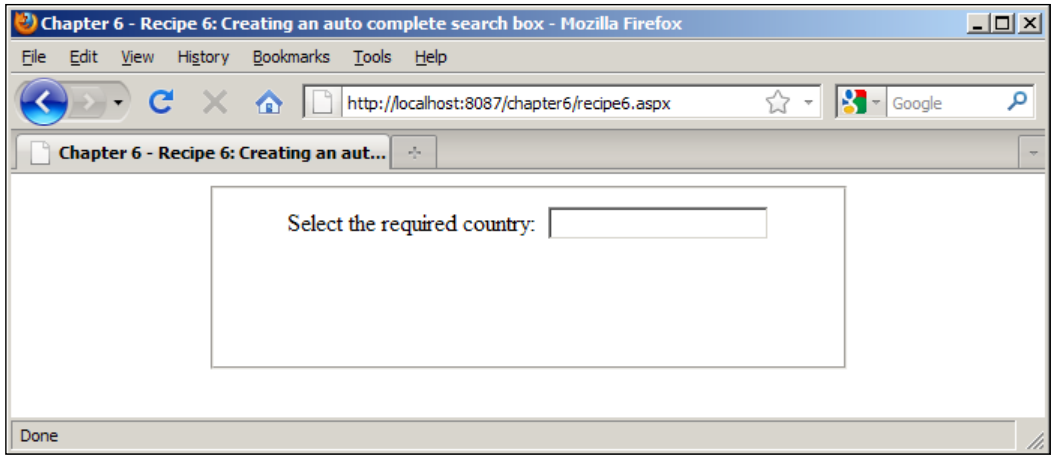
```
<link href="css/jquery-ui-1.8.6.custom.css" rel="stylesheet"
type="text/css" />
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
<script src="js/jquery.ui.core.js" type="text/javascript"></
script>
<script src="js/jquery.ui.position.js" type="text/javascript"></
script>
```

```
<script src="js/jquery.ui.widget.js" type="text/javascript"></
script>
<script src="js/jquery.ui.autocomplete.js" type="text/
javascript"></script>
```

12. Add a TextBox control to the form. Thus, the ASPX markup of the web form is as follows:

```
<form id="form1" runat="server">
     <div align="center">
    <fieldset style="width:400px;height:100px;">
    <table cellpadding="3" cellspacing="3" border="0">
    <tr><td>
        <label id="lblCountry" runat="server">Select the required
country: </label>
    </td><td>
        <asp:TextBox ID="txtSearch" runat="server"></asp:TextBox>
    </td></tr>
    </table>
    </fieldset>
    </div>
    </form>
```

13. On page load, the web form will appear as shown:



Now, we will use jQuery to add auto complete functionality to the TextBox. We will retrieve the matching keywords by posting an AJAX request to the HTTP handler.

How to do it...

1. In the `document.ready()` function of the jQuery script block, call the `autocomplete()` function on the TextBox control:

   ```
   $("#txtSearch").autocomplete({
   ```

2. Set the minimum length of the search keyword to trigger the retrieval of matching keywords:

   ```
   minLength: 1,
   ```

3. Set the source of the auto complete list to a callback function that takes in two parameters: request and response:

   ```
   source:
           function(request, response) {
   ```

The data `source` of the auto complete list can be either of the following:

Local Array

URL returning JSON data

Callback function

4. Call the `.ajax()` function in the preceding callback function:

   ```
   $.ajax({
   ```

5. Set the HTTP request type to `POST`:

   ```
   type: "POST",
   ```

6. Set the URL to the HTTP Handler. Pass the target keyword as a QueryString parameter to the Handler. The target keyword can be retrieved in the callback function using `request.term`:

   ```
   url: "SearchKeys.ashx?keyword=" + request.term,
   ```

7. Specify the content type of the data transfer:

   ```
   contentType: "application/json; charset=utf-8",
   ```

8. Specify the response data type. In this case, the HTTP Handler returns JSON data:

   ```
   dataType: "json",
   ```

9. Define the callback function on successful AJAX invocation:

```
success: function(data) {
            response($.map(data, function(item) {
                                    return {
                                        value: item
                                    }
                                }))
                            },
```

The jQuery function `.map(array, callback(elementOfArray, indexInArray))` takes in an array object and applies a callback function on each item of the array returning a new array object.

10. Define the callback function in case of failure:

```
error: function() {
    alert("An unexpected error has occurred during processing.");
                                }
                            });
                        }
            });
```

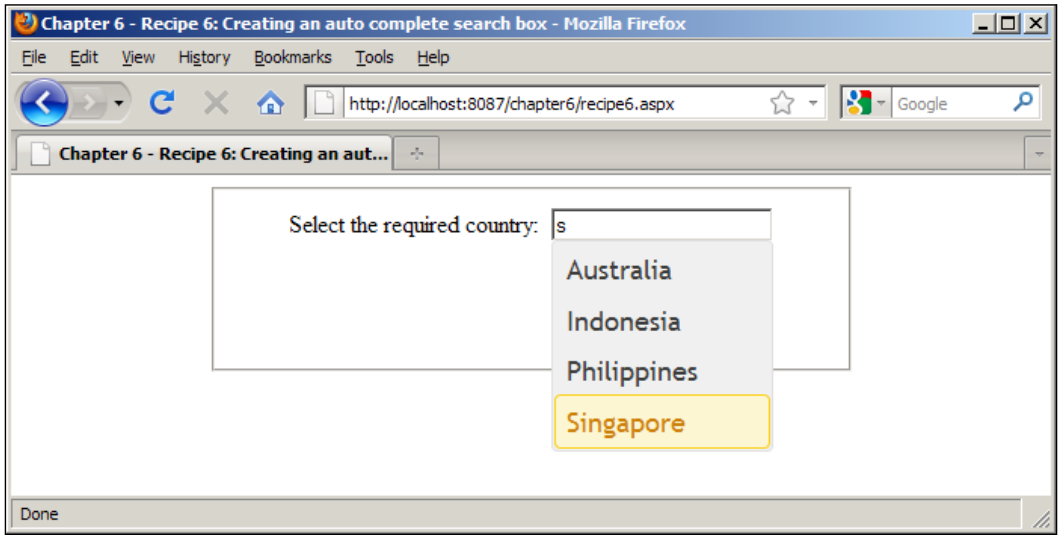
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        $("#txtSearch").autocomplete({
               minLength: 1,
            source:
                   function(request, response) {
                       $.ajax({
                           type: "POST",
                           url: "SearchKeys.ashx?keyword=" + request.
term,
                           contentType: "application/json;
charset=utf-8",
                           dataType: "json",
                           success: function(data) {
                             response($.map(data, function(item) {
                                 return {
                                     value: item
                                 }
                             }))
                         },
                           error: function() {
```

```
                                alert("An unexpected error has occurred
during processing.");
                            }
                        });
                    }
                });
            });
        </script>
```

How it works...

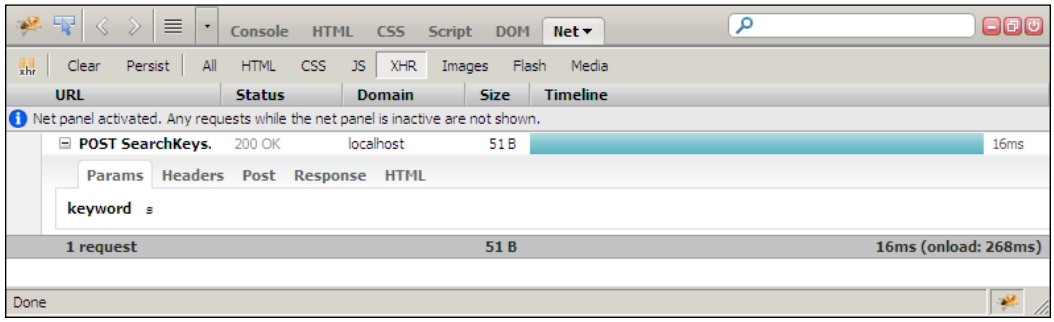
Run the web page. Enter in any character in the input TextBox. You will notice that the system pulls out matching keywords as shown:



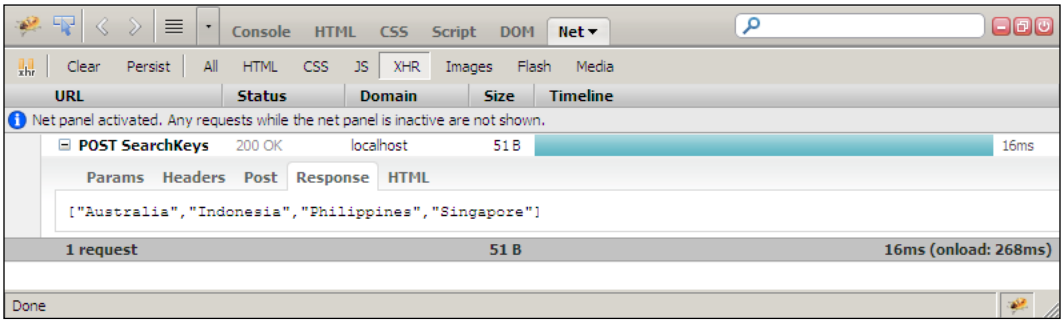
There's more...

Let's monitor the AJAX request/response using Firebug in Mozilla Firefox. Launch Firebug from the Tools menu.

Load the page and enter any character in the search TextBox to post the AJAX call. Click on **Net** tab and then on the **XHR** sub-tab. The `Params` will display the search keyword posted to the HTTP Hander as shown in the following screen:



Under `Response`, Firebug logs the JSON serialised result string as displayed next:



See also

Populating ASP.NET DropDownList using AJAX

7
AJAX and ASP.NET (Part II)

In this chapter, we will cover:

- Displaying a progress indicator during AJAX calls
- Reading XML data with AJAX
- Catching and displaying AJAX errors
- Using AJAX to load scripts in web pages
- Cross site AJAX querying using jQuery
- Using complex data types with AJAX

Introduction

In this chapter, we will continue to explore the applications of jQuery AJAX in ASP.NET and focus on the different types of data that we can work with. We will see examples that use HTML, XML, script, JSON, and JSONP data types in different scenarios.

Before going into the recipes, let's take a quick look at the most commonly used jQuery AJAX functions:

- **$.ajax(settings)**: This is a generic low level function that helps to create any type of AJAX request. There are a number of configuration settings that can be applied using this function to customize an AJAX call. It helps to set the type of HTTP request (i.e. GET/POST), the URL to send the request to, the parameters, data type, as well as the callback functions to execute on successful/unsuccessful invocation of the AJAX call.

- **Global AJAX Functions/Event Handlers**: jQuery provides a number of global functions for event handling. The most commonly used function is $.ajaxSetup(options), which helps to define default settings for making AJAX calls on the page.

  Other event handlers that centrally manage AJAX requests are:

  - **$.ajaxStart(handler())**: This handler is executed prior to making the first AJAX request on the page
  - **$.ajaxStop(handler())**: This handler is executed when all the AJAX requests on the page are complete
  - **$.ajaxSend(handler(event, XMLHttpRequest, ajaxOptions))**: This handler is executed before an AJAX request is sent
  - **$.ajaxSuccess(handler(event, XMLHttpRequest, ajaxOptions))**: This handler is executed whenever an AJAX request completes successfully
  - **$.ajaxError(handler(event, XMLHttpRequest, ajaxOptions, thrownError))**: This handler is executed whenever an AJAX request completes an error
  - **$.ajaxComplete(handler(event, XMLHttpRequest, ajaxOptions))**: This handler is executed whenever an AJAX request is complete

- **$.post(url, [data], [success(data, textStatus, XMLHttpRequest)], [dataType])**: This function helps to load data using HTTP POST request. It's a shortcut for the following AJAX function:

```
$.ajax({
  type: 'POST',
  url: url,
  data: data,
  success: success
  dataType: dataType
});
```

- **$.get (url, [data], [callback(data, textStatus, XMLHttpRequest)], [dataType])**: This function helps to load data using HTTP GET request. It's a shortcut for the following AJAX function:

```
$.ajax({
  url: url,
  data: data,
  success: success
  dataType: dataType
});
```

  where the default `type` of HTTP request is `GET`.

- **$.getJSON(url, [data], [callback(data, textStatus, xhr)])**: This function helps to load JSON data from the server using HTTP GET request. It's a shortcut for the following AJAX call:

```
$.ajax({
  url: url,
  dataType: 'json',
  data: data,
  success: callback
});
```

- **$.getScript(url, [success(data, textStatus)])**: This function helps to load a script file from the server using HTTP GET request. It also provides a callback function that is executed on successful load.

  This function is a shortcut for the following:

```
$.ajax({
  url: url,
  dataType: 'script',
  success: success
});
```

- **$.load(url, [data], [complete(responseText, textStatus, XMLHttpRequest)])**: This function helps to load HTML data from the server into the matched element.

Getting started

1. Let's start by creating a new ASP.NET website in Visual Studio and name it `Chapter7`.
2. Save the jQuery library in a script folder `js` in the project.
3. To enable jQuery on any web form, drag-and-drop the library to add the following to the page:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
```

Now, let's move on to the recipes where we will see different applications of jQuery AJAX in ASP.NET.

Displaying a progress indicator during AJAX calls

During AJAX calls, the client communicates with the server and at times there is a processing time involved. In this recipe, we will display a progress indicator on the page during the call while the client waits for the response to return from the server.

Getting ready

1. For this recipe, you can use any animated progress indicator image. To download a custom image visit `http://www.ajaxload.info/`.

2. Select the type of image as well as the required background/foreground colors. Generate and save the image file to the `images` folder in the project.
3. Add a new web form `Recipe1.aspx` to the current project.
4. Add a css class to display the progress indicator:

```
.Progress
    {
      background-image: url(images/ajax-loader.gif);
      background-position: center;
      background-repeat: no-repeat;
      cursor: wait;
      padding: 10px;
```

```
    width: 200px;
    height: 100px;
}
```

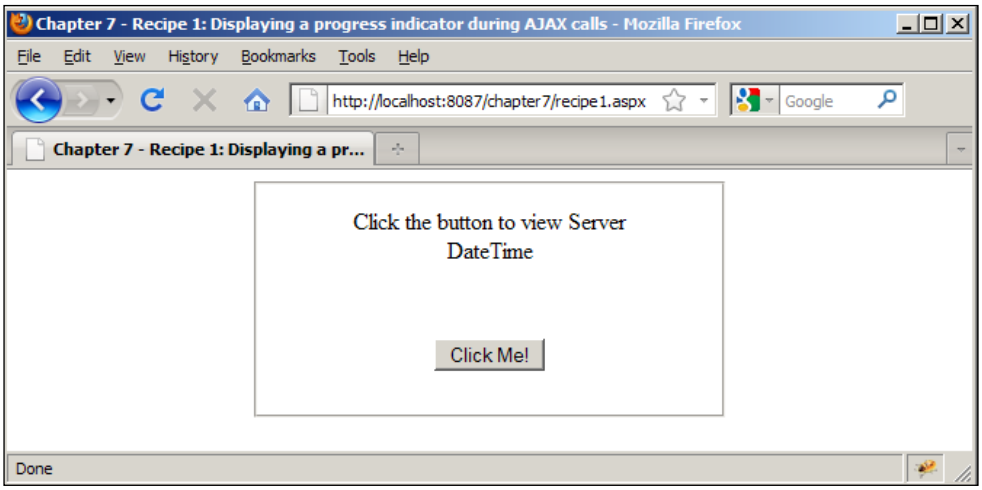
5. Add a button control to the form to trigger the AJAX request. Thus, the ASPX markup is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:300px;height:200px;">
        <div id="contentArea">
        Click the button to view Server DateTime
        </div>
        <br /><br />
        <asp:Button ID="btnSubmit" runat="server" Text="Click
Me!" />
    </fieldset>
    </div>
    </form>
```

6. Define the css style of the containing div area as follows:

```
#contentArea
    {
      padding: 10px;
      width: 200px;
      height: 100px;
      text-align:center;
    }
```

7. Thus, on page load, the form appears as shown:



On clicking the button control we will call a page method. We will introduce a time lag in the method. During this time lag, we will display the progress indicator image on the page. Hence, let's define a page method that returns the server time with an artificial delay:

1. In the code-behind file `Recipe1.aspx.cs`, include the following namespace to enable page methods:

   ```
   using System.Web.Services;
   ```

2. Write a page method that returns the current server time in string format after a delay of 5000 ms:

   ```
   [WebMethod]
       public static string GetServerTime()
       {
           System.Threading.Thread.Sleep(5000);
           return DateTime.Now.ToLongDateString() + " " + DateTime.
   Now.ToLongTimeString();

       }
   ```

Note that the page method should be defined as `static` so that it can be accessed directly by the client.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the button control:

   ```
   $("#btnSubmit").click(function(e) {
   ```

2. Prevent default form submission:

   ```
   e.preventDefault();
   ```

3. Clear the containing div area on the form:

   ```
   $("#contentArea").html("");
   ```

4. Add the css class `Progress` to the containing div area. Note that this will cause the progress indicator to display on the page:

   ```
   $("#contentArea").addClass("Progress");
   ```

5. Call the `sendData()` function. The AJAX call is made in this function:

   ```
   sendData();
   ```

6. Define the `sendData()` function as follows:

   ```
   function sendData() {
   ```

7. Get the complete path of the current page:

```
var loc = window.location.href;
loc = (loc.substr(loc.length - 1, 1) == "/") ? loc + "Recipe1.
aspx" : loc;
```

8. Make the AJAX call:

```
$.ajax({
```

9. Specify the HTTP request type:

```
type: "POST",
```

10. Specify the URL of the page method:

```
url: loc + "/GetServerTime",
```

11. Specify the content type of the data transfer:

```
contentType: "application/json; charset=utf-8",
```

12. Specify the response data type:

```
dataType: "json",
```

13. Define the callback function for successful invocation. On receiving the response, remove the css class `Progress` from the containing div area. As a result, the progress indicator image will be removed from the page. Display the AJAX response in the div area instead:

```
success: function(msg) {
                        $("#contentArea").removeClass("Progress");
$("#contentArea").html(msg.d);
},
```

Note that the AJAX response is wrapped in a `d` object e.g. `{"d":true}`. Hence, to access the response from the page method, we use `msg.d`.

14. Define the callback function for failed invocation:

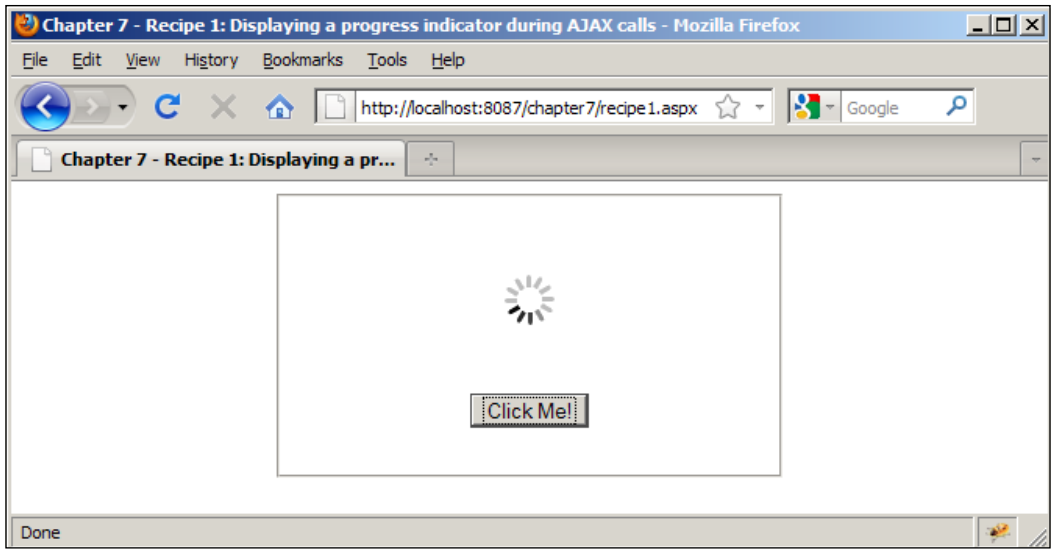
```
error: function() {
     alert("An unexpected error has occurred during processing.");
                 }
            });
        }
    });
```

Thus, the complete jQuery solution is as follows:

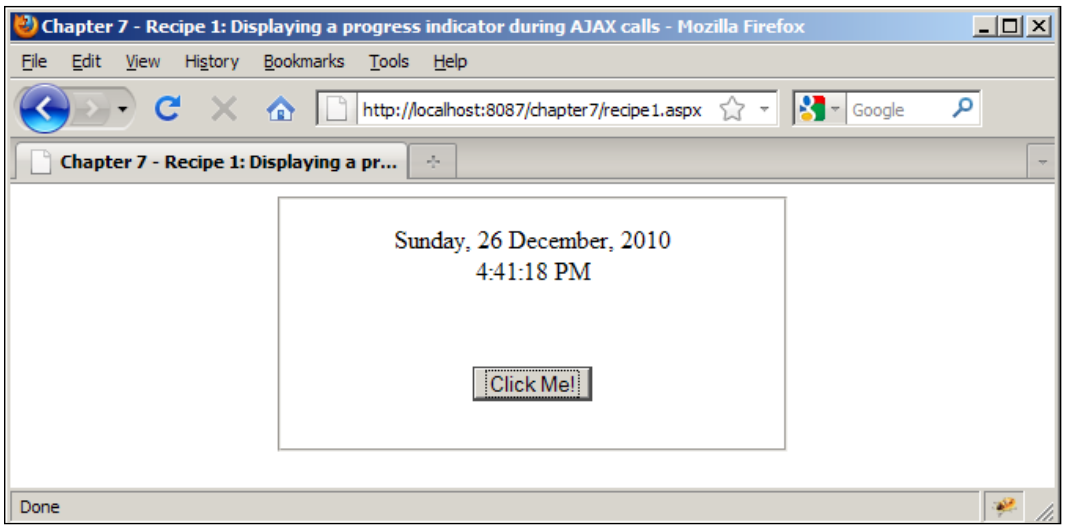
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                $("#contentArea").html("");
                $("#contentArea").addClass("Progress");
                sendData();

                function sendData() {
                    var loc = window.location.href;
                    loc = (loc.substr(loc.length - 1, 1) == "/") ? loc
                    + "Recipe1.aspx" : loc;
                    $.ajax({
                        type: "POST",
                        url: loc + "/GetServerTime",
                        contentType: "application/json;
                        charset=utf-8",
                        dataType: "json",
                        success: function(msg) {
                            $("#contentArea").removeClass("Progress");
                            $("#contentArea").html(msg.d);
                        },
                        error: function() {
                            alert("An unexpected error has occurred
                            during processing.");
                        }
                    });
                }

            });
        });
    </script>
```

How it works...

Run the web page. Click on the button control. You will notice that the page displays a progress indicator as shown next:



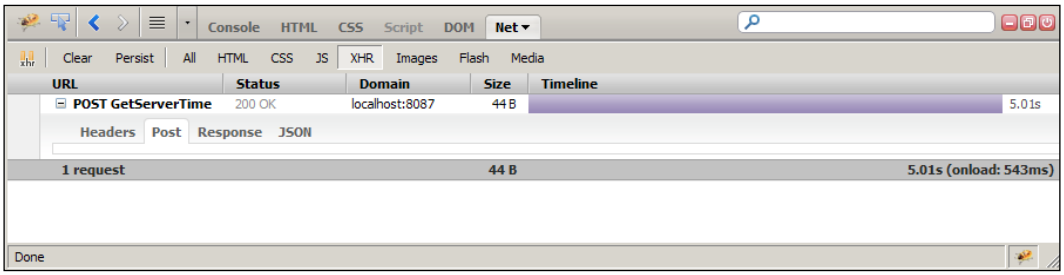
After the time delay, the client receives the response from the server and displays the current server datetime as shown next:



There's more...

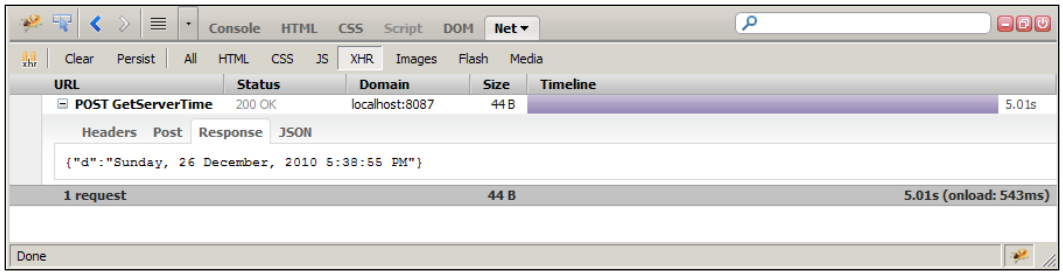
Let's see the AJAX request/response triggered by the page using Firebug. Load the page in Mozilla Firefox and launch the Firebug tool.

Click on the **Net** tab and then on the **XHR** sub-tab. Click on the **Post** tab to view the Ajax request as follows:



Note that as displayed in the preceding screen, the browser makes AJAX request to the `GetServerTime` page method. No parameters are passed to the page method.

Click on the **Response** sub-tab. Firebug will display the response as follows:



See also

Reading XML data with AJAX

Reading XML data with AJAX

AJAX can be used to work with different types of data such as JSON, JSONP, XML, HTML, text, or script. In this recipe, let's see how to load an XML DOM object using an AJAX call.

Getting ready

1. Create an XML document `BookList.xml` and save in the root directory of the current project. Populate the following contents to the document:

```
<?xml version="1.0" encoding="utf-8" ?>
<BookList>
  <Book>
    <Title>Alchemist</Title>
    <Author>Paulo Coelho</Author>
    <Genre>Fiction</Genre>
  </Book>
  <Book>
    <Title>Heart of Darkness</Title>
    <Author>Joseph Conrad</Author>
    <Genre>Classic</Genre>
  </Book>
  <Book>
    <Title>David Copperfield</Title>
    <Author>Charles Dickens</Author>
    <Genre>Classic</Genre>
  </Book>
  <Book>
    <Title>Have a Little Faith</Title>
    <Author>Mitch Albom</Author>
    <Genre>Fiction</Genre>
  </Book>
</BookList>
```

2. Add a new web form `Recipe2.aspx` to the project.
3. Add the following markup to the form:

```
<form id="form1" runat="server">
<div align="center">
<fieldset style="width:350px;height:100px;">
 <div id="contentArea" align="left">
 </div>
</fieldset>
</div>
</form>
```

We will use jQuery AJAX to display the contents of the XML document in the preceding containing div area on the page. We will display each XML node as a list item of an unordered HTML list. For each node `BookList/Book`, we will retrieve the following sub-nodes and display it on the page:

- Title
- Author
- Genre

How to do it...

1. In the `document.ready()` function of the jQuery script block, append the containing tags of the unordered list:
   ```
   $("#contentArea").append("<ul></ul>");
   ```
2. Call the `.ajax()` method:
   ```
   $.ajax({
   ```
3. Set the HTTP request type to `GET`:
   ```
   type: "GET",
   ```
4. Specify the URL of the XML document:
   ```
   url: "BookList.xml",
   ```
5. Specify the data type of the response i.e. in this case `xml`:
   ```
   dataType: "xml",
   ```
6. Define the callback function for successful invocation:
   ```
   success: function(xml){
   ```
7. Loop through each node `BookList/Book` in the XML document:
   ```
   $(xml).find('Book').each(function(){
   ```
8. Retrieve the content of the `'Title'` sub-node:
   ```
   var sTitle = $(this).find('Title').text();
   ```
9. Retrieve the content of the `'Author'` sub-node:
   ```
   var sAuthor = $(this).find('Author').text();
   ```
10. Retrieve the content of the `'Genre'` sub-node:
    ```
    var sGenre = $(this).find('Genre').text();
    ```

11. Append the retrieved contents we just saw as a list item to the unordered list on the form:

```
$("<li></li>").html(sTitle + ", " + sAuthor + ", " + sGenre).
appendTo("#contentArea ul");
                  });
               },
```

12. Define the callback function in case of error:

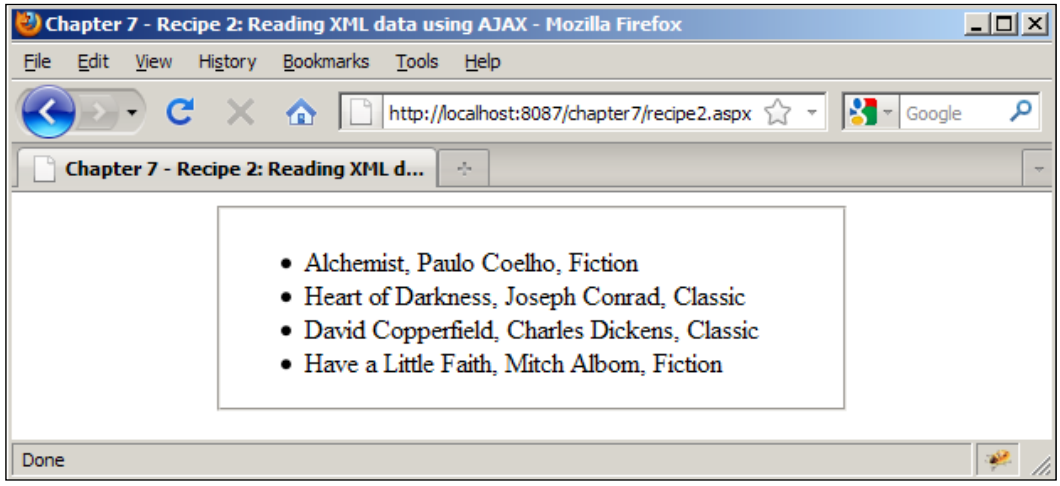
```
error: function() {
                        alert("An unexpected error has occurred
during processing.");
                     }
           });
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
          $("#contentArea").append("<ul></ul>");
           $.ajax({
               type: "GET",
               url: "BookList.xml",
               dataType: "xml",
               success: function(xml){
                  $(xml).find('Book').each(function(){
                      var sTitle = $(this).find('Title').text();
                      var sAuthor = $(this).find('Author').text();
                      var sGenre = $(this).find('Genre').text();
                      $("<li></li>").html(sTitle + ", " + sAuthor + ",
                      " + sGenre).appendTo("#contentArea ul");
                  });
               },
               error: function() {
                        alert("An unexpected error has occurred during
                        processing.");
                    }
           });
        });
    </script>
```

How it works...

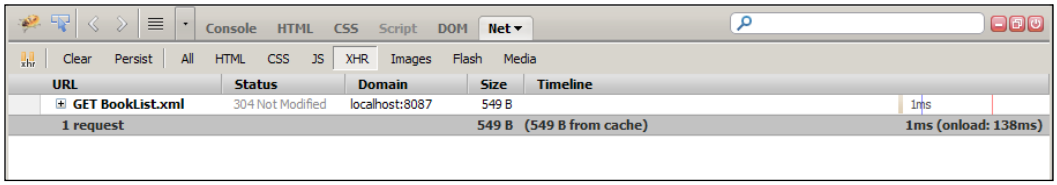
Load the page in the browser. The contents of the XML document will display on the page as an unordered HTML list as follows:



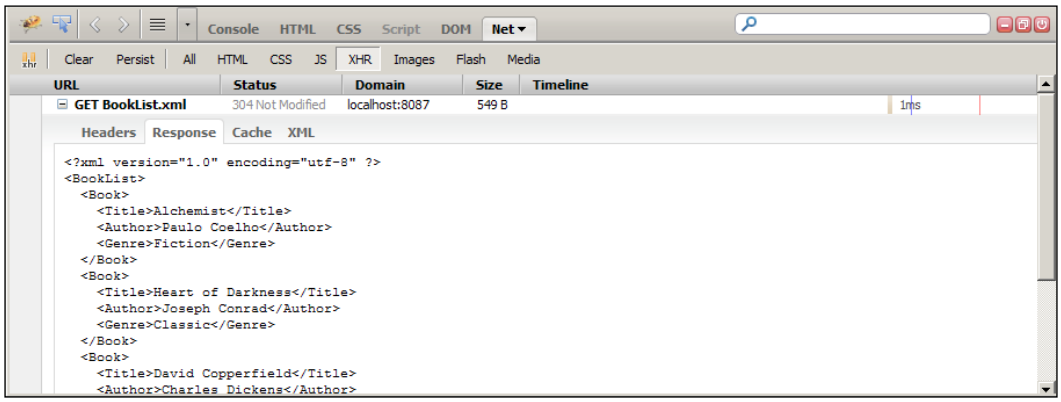
There's more...

Let's use Firebug in Mozilla Firefox browser to view the AJAX request/response.

Load the page in Mozilla Firefox browser and launch Firebug. Click on the **Net** tab and then on the **XHR** sub-tab. Firebug displays the GET AJAX call to `BookList.xml` as follows:



The resulting XML data can be seen in the **Response** sub-tab as follows:



See also

Displaying a progress indicator during AJAX calls

Catching and displaying AJAX errors

In certain applications, there might be a need to display the server exception, if any, raised during an AJAX call, for information/troubleshooting on the client. In this recipe, we will see how to retrieve the server exception and the XmlHTTPRequest statuses on the client.

We will go about by posting an AJAX request for a file retrieval from the server. We will intentionally throw an IO exception on the server and capture the same on the client.

Getting ready

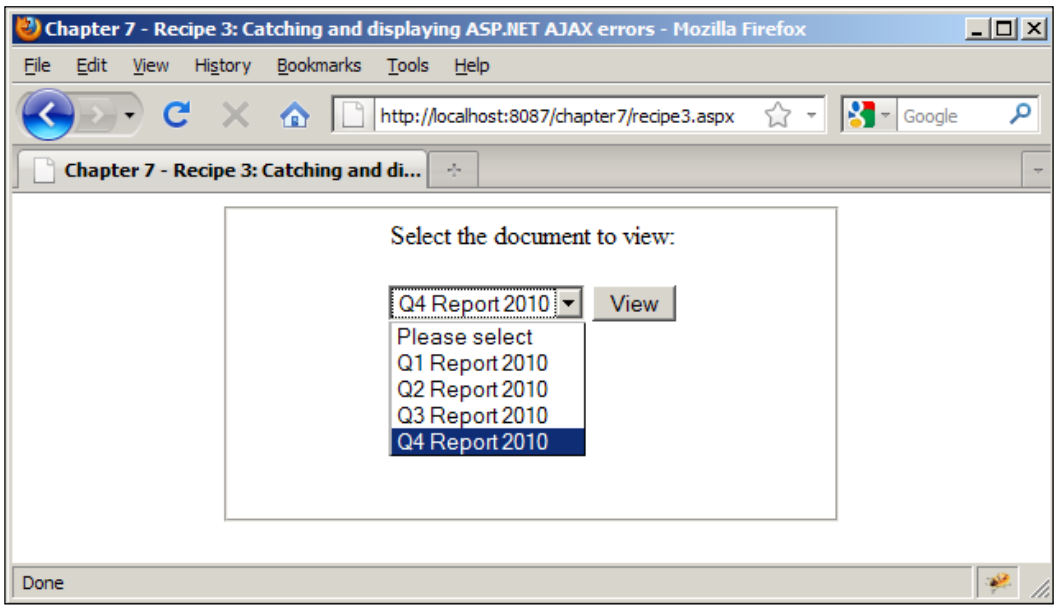
1. Add a web form `Recipe3.aspx` to the current project.
2. Add a DropDownList to the form. Set the value of each item of the DropDownList to a filename for retrieval from the server.
3. Thus, the complete ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:350px;height:250px;">
     <div id="contentArea">
         Select the document to view:<br /><br />
         <asp:DropDownList ID="ddDocumentList" runat="server">
         <asp:ListItem Text="Please select" Value="[-]"></
asp:ListItem>
```

```
            <asp:ListItem Text="Q1 Report 2010" Value="Q1Report2010.
pdf"></asp:ListItem>
            <asp:ListItem Text="Q2 Report 2010" Value="Q2Report2010.
pdf"></asp:ListItem>
            <asp:ListItem Text="Q3 Report 2010" Value="Q3Report2010.
pdf"></asp:ListItem>
            <asp:ListItem Text="Q4 Report 2010" Value="Q4Report2010.
pdf"></asp:ListItem>
            </asp:DropDownList>
            <asp:Button ID="btnSubmit" runat="server" Text="View" />
            <br /><br />
            <div id="message"></div>
            </div>
        </fieldset>
        </div>
        </form>
```

Note that in the preceding markup, the div area `message` is defined to display the exception message received from the server side.

4. On page load, the form appears as shown in the following screenshot:



On the button click event, we will trigger an AJAX call to a page method. We will intentionally throw an exception in the page method and capture the corresponding error message for display on the client. Let's write the page method in the code-behind.

5. In the code-behind file `Recipe3.aspx.cs`, add the following namespaces:

```
using System.Web.Services;
using System.IO;
```

6. Write the following page method that throws an IO exception:

```
[WebMethod]
    public static string GetFileData(string FileName)
    {
        try
        {
            throw new IOException("File " + FileName + " not
found");
        }
        catch (Exception e)
        {
            return e.Message;
        }

    }
```

Next, we will see how to retrieve and display the preceding exception message on the client side.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the button control:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Retrieve the selected filename from the DropDownList:

```
var FileName = $("#ddDocumentList").val();
```

4. If the selected filename is not empty, call the `sendData()` function:

```
if ( FileName != "[-]")
           sendData();
```

5. If the filename is empty, alert the user to select a valid item from the DropDownList:

```
else
    alert("Please select the document to view");
```

6. Write the `sendData()` function:

```
function sendData() {
```

7. Get the complete base path of the current page:

```
var loc = window.location.href;
loc = (loc.substr(loc.length - 1, 1) == "/") ? loc + "Recipe3.
aspx" : loc;
```

8. Call the `.ajax()` method:

```
$.ajax({
```

9. Specify the HTTP request type as `POST`:

```
type: "POST",
```

10. Send the URL of the page method to send the AJAX request to:

```
url: loc + "/GetFileData",
```

11. Send the input parameter of the page method i.e. filename as a JSON formatted string:

```
data: '{"FileName":"' + FileName + '"}',
```

12. Specify the content type of the transfer:

```
contentType: "application/json; charset=utf-8",
```

13. Specify the data type of the response:

```
dataType: "json",
```

14. Define the callback function on the completion of the AJAX call. The callback function takes in two parameters: the XmlHttpRequest object and the response status:

```
complete: function(xhr, status) {
```

15. Parse the JSON response string retrieved from the server side in `xhr.responseText`:

```
var msg = $.parseJSON(xhr.responseText);
```

16. Clear the div area before displaying the response from the server:

```
$("#message").html("");
```

17. Display the full JSON formatted response string:

```
$("#message").append("<b>ResponseText:</b> " + xhr.responseText);
```

18. Display the actual exception message retrieved by parsing the preceding string:

```
$("#message").append("<br><br><b>Message:</b> " + msg.d);
```

19. Display the response status text:

```
$("#message").append("<br><br><b>StatusText:</b> " + xhr.
statusText);
```

20. Display the response status:

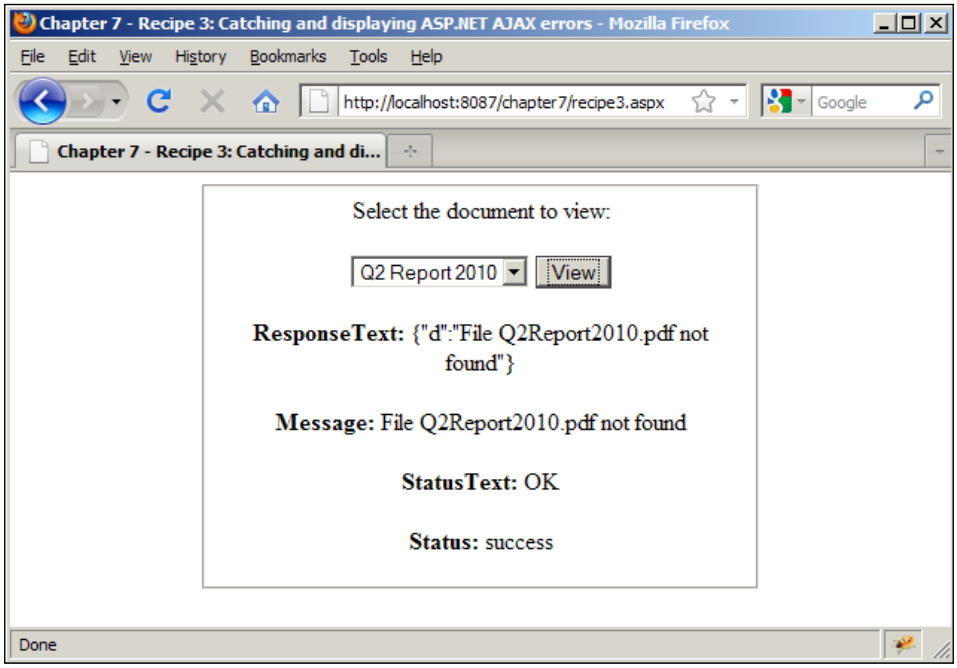
```
$("#message").append("<br><br><b>Status:</b> " + status);
                        }
                    });
                }
            });
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
                e.preventDefault();
                var FileName = $("#ddDocumentList").val();
                if ( FileName != "[-]")
                    sendData();
                else
                   alert("Please select the document to view");

                function sendData() {
                    var loc = window.location.href;
                    loc = (loc.substr(loc.length - 1, 1) == "/") ? loc
                    + "Recipe3.aspx" : loc;
                    $.ajax({
                        type: "POST",
                        url: loc + "/GetFileData",
                        data: '{"FileName":"' + FileName + '"}',
                        contentType: "application/json;
                        charset=utf-8",
                        dataType: "json",
                        complete: function(xhr, status) {
                            var msg = $.parseJSON(xhr.responseText);
                            $("#message").html("");
                            $("#message").append("<b>ResponseText:</b>
                            " + xhr.responseText);
                            $("#message").
                            append("<br><br><b>Message:</b> " +
                            msg.d);
                            $("#message").
                            append("<br><br><b>StatusText:</b> " +
                            xhr.statusText);
                            $("#message").append("<br><br><b>Status:</
                            b> " + status);
                        }
                    });
                }
            });
        });
    </script>
```

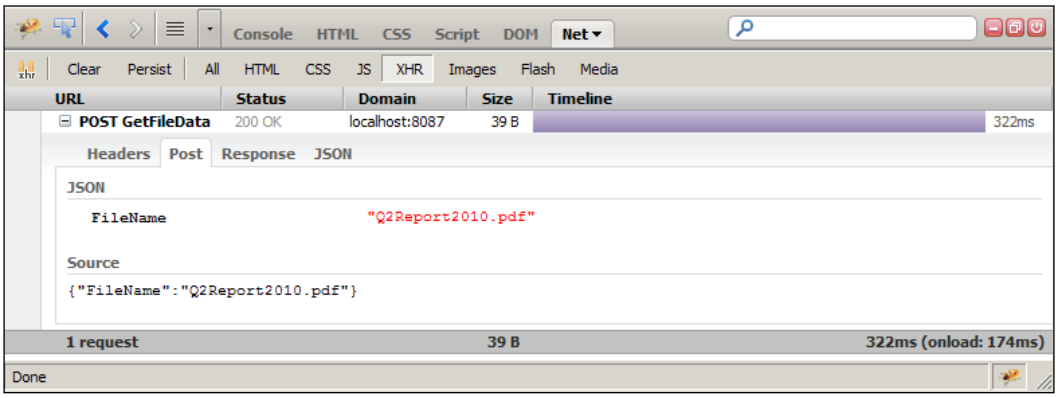
How it works...

Run the page. Select any document name from the DropDownList and click on the view button. The page displays the exception message and response statuses as shown next:

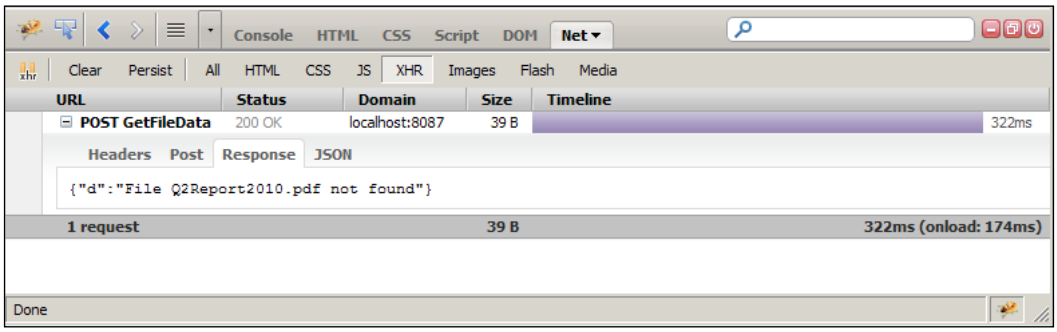


There's more...

Launch Firebug in Mozilla Firefox browser. Click the **Net** tab and then the **XHR** sub-tab. In the **XHR** tab, now select the **Post** tab to view the AJAX request as shown next:



Now, switch to the **Response** tab to view the AJAX response as displayed next:



See also

Using AJAX to load scripts in web pages

Using AJAX to load scripts in web pages

In this recipe, let's see how to work with the `script` data type in AJAX. To demonstrate, we will load the jQuery form plugin on the page using an AJAX call. We will then add the reset functionality using this plugin to the **Reset** button on the form.

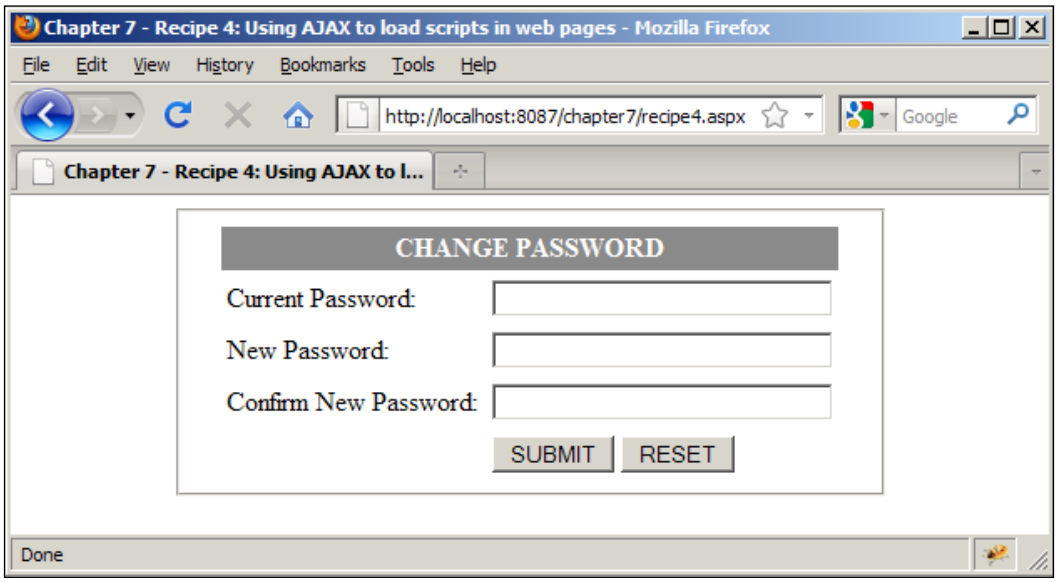
Getting ready

1. Download the jQuery form plugin from `http://plugins.jquery.com/project/form` and save it to the `js` folder in the project.
2. Add a new web form `Recipe4.aspx` to the current project.
3. Create a simple **Change Password** form with the following markup:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:400px;height:170px;">
        <table cellpadding="3" cellspacing="3" border="0">
    <tr><td colspan="2" class="header">CHANGE PASSWORD</td></tr>
    <tr><td> <asp:Label id="lblCurrentPwd" Text="Current Password:
" runat="server"/></td>
    <td> <asp:TextBox ID="txtCurrentPwd" Width="200px"
runat="server" TextMode="Password"></asp:TextBox></td></tr>
    <tr><td><asp:Label id="lblNewPwd" Text="New Password:"
runat="server"/></td>
    <td><asp:TextBox ID="txtNewPwd" Width="200px" runat="server"
TextMode="Password"></asp:TextBox></td></tr>
```

```
    <tr><td><label id="lblConfirmNewPwd" runat="server">Confirm
New Password:</label></td>
    <td><asp:TextBox ID="txtConfirmNewPwd" Width="200px"
runat="server" TextMode="Password"></asp:TextBox></td></tr>
    <tr><td></td><td><asp:Button ID="btnSubmit" runat="server"
Text="SUBMIT" />
    <asp:Button ID="btnReset" runat="server" Text="RESET" />
</td></tr>
    </table>
    </fieldset>
    </div>
    </form>
```

4. On page load, the form appears as follows:



We will now use jQuery to load the form plugin and add the reset functionality to the **Reset** button control on the form.

How to do it...

1. In the `document.ready()` function of the jQuery script block, call the `sendData()` function:

   ```
   sendData();
   ```

2. Define the `sendData()` function:

   ```
   function sendData() {
   ```

3. Call the `.ajax()` method:
   ```
   $.ajax({
   ```
4. Set the HTTP request type as `GET`:
   ```
   type: "GET",
   ```
5. Specify the URL of the script file to load:
   ```
   url: "js/jquery.form.js",
   ```
6. Specify the data type of the response. In this case, the expected response is of type `script`:
   ```
   dataType: "script",
   ```
7. Write the callback function for successful invocation:
   ```
   success: function(){
   ```
8. In the callback, define the event handler for the `click` event of the `Reset` button control:
   ```
   $("#btnReset").click(function(e){
   ```
9. Prevent default form submission:
   ```
   e.preventDefault();
   ```
10. Call the `resetForm()` method of the form plugin as follows:
    ```
    $("#form1").resetForm();
                                  });
                              }
                          });
    ```

Thus, the complete jQuery solution is as follows:

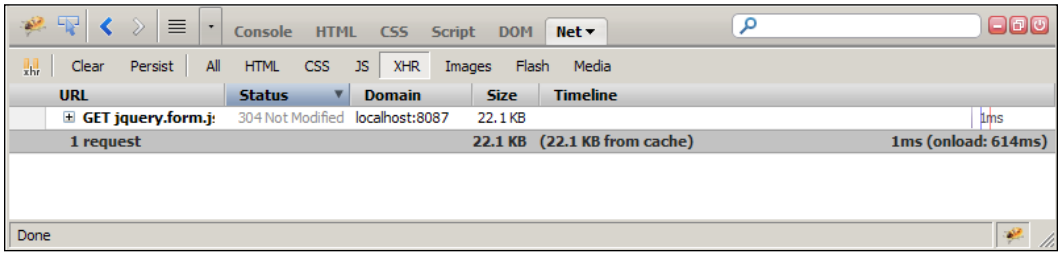
```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
        sendData();

        function sendData() {
           $.ajax({
                 type: "GET",
                 url: "js/jquery.form.js",
                 dataType: "script",
                 success: function(){
                           $("#btnReset").click(function(e){
                           e.preventDefault();
                           $("#form1").resetForm();
                           });
```

```
                    }
                });
            }
        });
</script>
```

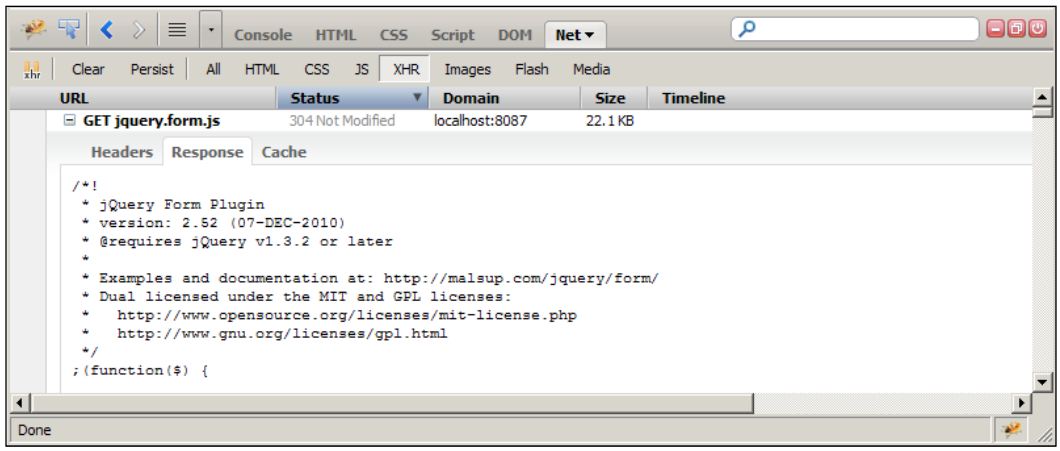
How it works...

Run the page. Fill in any random data in the change password form. On clicking the **Reset** button, you will notice that the form fields are cleared.

On inspecting the page using Firebug in Mozilla Firefox, we notice the `Get` request made to the plugin JavaScript file as follows:



Click on the **response** tab in the last screen to view the plugin JavaScript file as follows:



There's more...

We can also accomplish the loading of scripts by using an AJAX shortcut `$.getScript()` as follows:

```
$.getScript("js/jquery.form.js",function(){
                            $("#btnReset").click(function(e){
                            e.preventDefault();
                            $("#form1").resetForm();
                            });
                        });
```

`$.getScript()` function replaces the standard `$.ajax()` function and can be used to call scripts using AJAX. It also provides a callback function on successful loading of the script.

See also

Cross site AJAX querying using jQuery

Cross site AJAX querying using jQuery

Web browsers, for security reasons, enforce a 'same domain policy' and allow AJAX calls to pages only on the same domain. One method of getting around this restriction to make cross domain requests is to use JSONP (JSON with padding). In this recipe, we will demonstrate an example of cross site querying using Yahoo! Search API and jQuery AJAX.

JSONP is simply JSON wrapped in a user specified function call. The remote JSONP-enabled service receives a callback function name from the client (the calling script). It then returns the JSON response as a parameter in the specified callback function. This function is subsequently executed on the client browser.

Getting ready

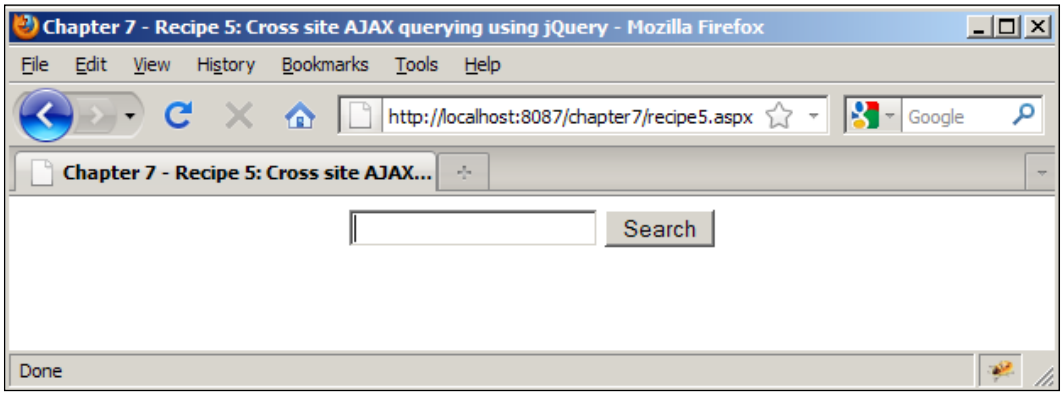
1. Add a new web form `Recipe5.aspx` to the current project.

2. Add a TextBox with a Search button control to the form as follows:

```
<form id="form1" runat="server">
    <div align="center">
    <asp:TextBox ID="txtKeyword" runat="server"></asp:TextBox>
    <asp:Button ID="btnSearch" runat="server" Text="Search" />
    <br /><br />
    <div id="result" align="left">
    </div>
    </div>
    </form>
```

   Note that in the preceding markup, we have defined a div area `result` for displaying the search result.

3. Thus, on page load, the form appears as follows:



JSON is supported as an alternative output format for many of Yahoo!'s web service APIs. In this recipe, we will use the Yahoo! Search API, which has the following syntax:

```
http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&output=json&query=search_keyword&callback=myFunction
```

where:

- `output` is the data type of output (in this case, JSON)
- `query` is the search keyword
- `callback` is the callback function executed by the client browser

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the Search button:

```
$("#btnSearch").click(function(e){
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Retrieve the search keyword from the TextBox:

```
var keyword = $("#txtKeyword").val();
```

4. If the search keyword is empty, alert the user:

```
if (keyword == '')
           alert("Please enter the search keyword");
```

5. If the search keyword is not empty, set the URL to send the AJAX request:

```
else
   {
   var url = "http://search.yahooapis.com/WebSearchService/
V1/webSearch?appid=YahooDemo&output=json&query=" + keyword +
"&callback=?";
```

When using `$.getJSON` function in jQuery, by setting the `callback` parameter in the URL to ? i.e. `callback=?`, jQuery automatically generates a function name and binds it to the default callback function in `$.getJSON`.

6. Use `.getJSON()` to load the JSON encoded data from the server using HTTP GET request. Note that the params list in the function call is empty:

```
$.getJSON(url,{},function(json){
```

7. Clear the div area for displaying the search results:

```
$("#result").html("");
```

8. Append the total count of search results returned:

```
$("#result").append("Total Search Results: " + json.ResultSet.
totalResultsAvailable + "<br/><br/>");
```

9. Loop through each item in the returned search response:

```
for(var i=0; i<json.ResultSet.Result.length; i++) {
```

10. Get the current result item:

```
var result = json.ResultSet.Result[i];
```

11. Retrieve the URL and the Title of the current result item:

```
var resultStr = '<a href="'+result['ClickUrl']+'">'+result['Tit
le']+'</a><br/>';
```

12. Retrieve the Summary of the current result item:

```
resultStr += result['Summary'] + '<br/><br/>';
```

13. Append the URL, Title, and Summary to the div area:

```
$("#result").append(resultStr);
                                          }
                      });
                      }
                  });
```

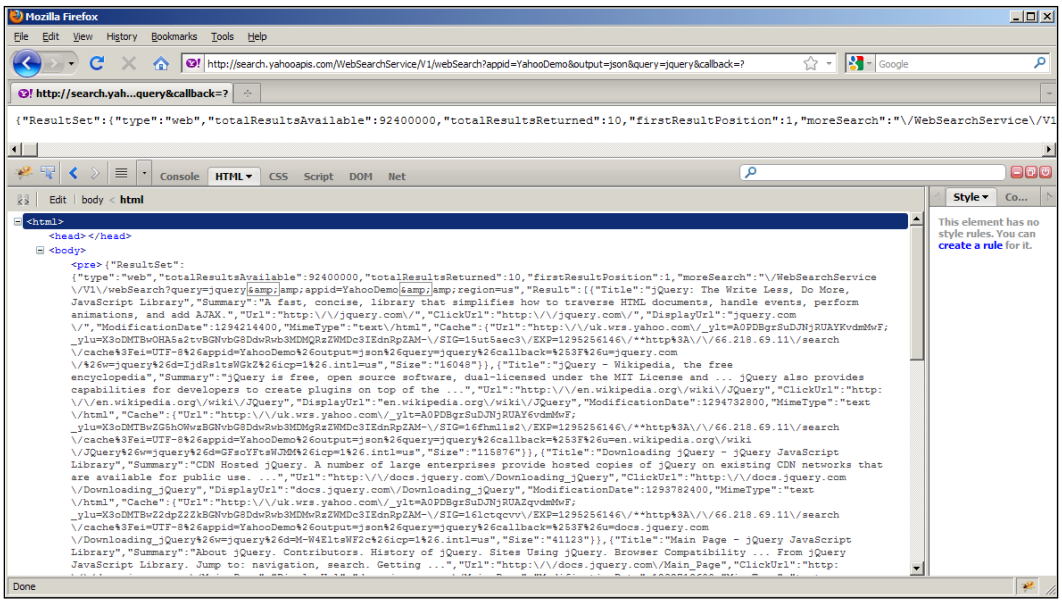
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
                $("#btnSearch").click(function(e){
                 e.preventDefault();
                 var keyword = $("#txtKeyword").val();
                 if (keyword == '')
                 alert("Please enter the search keyword");
                 else
                 {
                 var url = "http://search.yahooapis.com/
                 WebSearchService/V1/webSearch?appid=YahooDemo&output=
                 json&query=" + keyword + "&callback=?";
                 $.getJSON(url,{},function(json){
                   $("#result").html("");
                   $("#result").append("Total Search Results:
                   " + json.ResultSet.totalResultsAvailable +
                   "<br/><br/>");
      for(var i=0; i<json.ResultSet.Result.length; i++) {
                 var result = json.ResultSet.Result[i];
                 var resultStr = '<a href="'+result['ClickUrl']+'">'+res
                 ult['Title']+'</a><br/>';
resultStr += result['Summary'] + '<br/><br/>';
                                          $("#result").
                                          append(resultStr);
                 }
              });
          }
        });
        });
</script>
```

To understand the response returned by the AJAX call, access the Yahoo! Search API directly from a browser with any random search keyword. For example, we can pass the keyword **jQuery** to the API URL as follows:

```
http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&output=json&query=jQuery&callback=?
```

To view the detailed JSON output, launch Firebug in Mozilla Firefox browser as shown in the following screenshot:



Our task is to format the JSON data returned by the service in the callback function. For this purpose, we need to understand the structure of the returned response. The following is a short snippet of the response:

```
"ResultSet":{"type":"web","totalResultsAvailable":9240000
0,"totalResultsReturned":10,"firstResultPosition":1,"more
Search":"\/WebSearchService\/V1\/webSearch?query=jquery&a
mp;appid=YahooDemo&amp;region=us","Result":[{"Title":"jQue
ry: The Write Less, Do More, JavaScript Library","Summary":"A
fast, concise, library that simplifies how to traverse
HTML documents, handle events, perform animations, and add
AJAX.","Url":"http:\/\/jquery.com\/","ClickUrl":"http:\/\/jquery.
com\/","DisplayUrl":"jquery.com\/","ModificationDate":129421440
0,"MimeType":"text\/html","Cache":{"Url":"http:\/\/uk.wrs.yahoo.
com\/_ylt=A0PDBgrSuDJNjRUAYKvdmMwF;_ylu=X3oDMTBwOHA5a2tvBGNvbG8Ddw
Rwb3MDMQRzZWMDc3IEdnRpZAM-\/SIG=15ut5aec3\/EXP=1295256146\/**http%
3A\/\/66.218.69.11\/search\/cache%3Fei=UTF-8%26appid=YahooDemo%26outpu
```

```
t=json%26query=jquery%26callback=%253F%26u=jquery.com\/%26w=jquery%26d
=IjdRs1tsWGkZ%26icp=1%26.intl=us","Size":"16048"}},
```

As seen from the preceding JSON response, the total number of search results can be retrieved from `ResultSet.totalResultsAvailable`.

Each individual search item can then be parsed as follows:

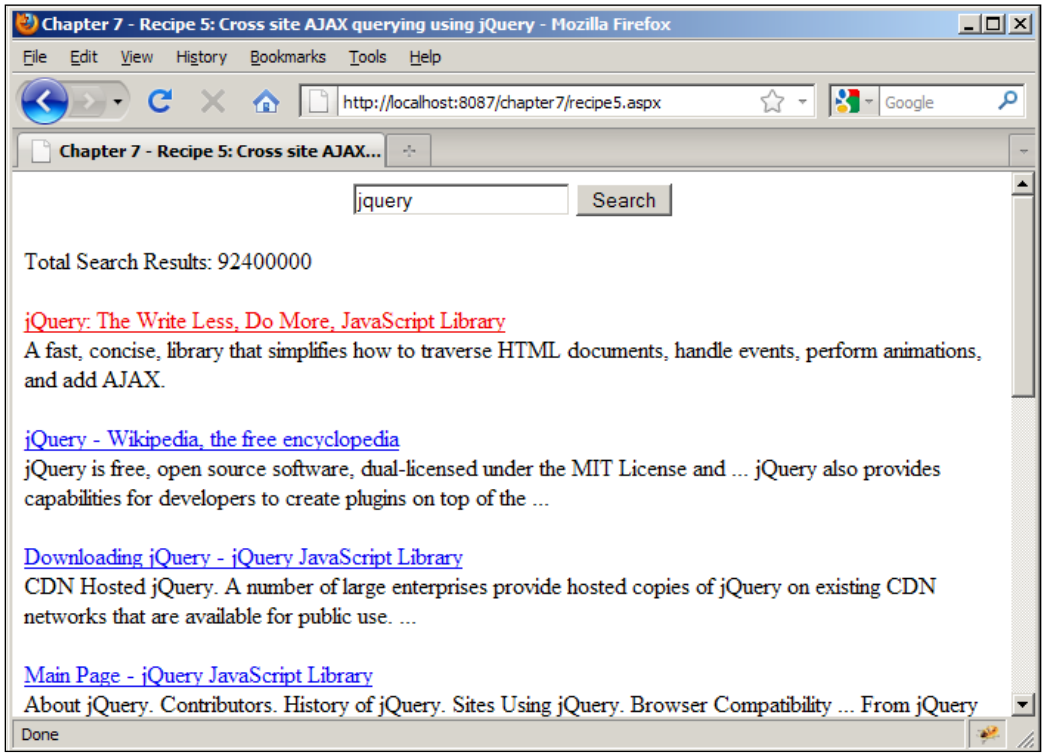
```
var result = json.ResultSet.Result[i];
```

- Retrieve `'Title'` using `result['Title']`
- Retrieve `'Summary'` using `result['Summary']`
- Retrieve `'ClickUrl'` using `result['ClickUrl']`

The response also contains additional properties such as `ModificationDate`, `DisplayUrl`, `Cache`, etc. that can be used as required.

How it works...

Run the web page. Key in any random search keyword in the `TextBox` and click the `Search` button. The page displays the search results as shown next:



Important note: When using this method of making cross domain requests, it is important to use reliable JSONP services. Since the JSON response is returned in a callback function that is executed on the client browser, the web application becomes vulnerable to a variety of attacks. For security reasons, use only JSONP services from trusted sources.

See also

Using AJAX to load scripts in web pages

Using complex data types with AJAX

So far, we have seen how AJAX calls can be made by passing simple data types as parameters. Let's focus now on a scenario where we might need to pass a complex data type such as an object instead.

Getting ready

1. Add a web form `Recipe6.aspx` to the current project.
2. Download `Json2.js` file from `http://www.json.org/js.html` and save to the `js` folder.

To convert an object into JSON format we will use the `stringify()` function available in `Json2.js` written by Douglas Crockford (`http://www.crockford.com/`).

Include the preceding file along with the jQuery library on the page as follows:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"></script>
<script src="js/json2.js" type="text/javascript"></script>
```

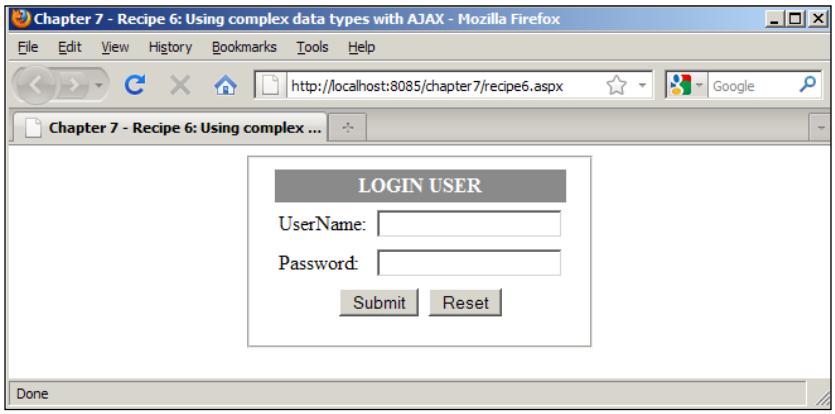
3. Create a simple user login form with the following markup:

```
<form id="form1" runat="server">
    <div align="center">
    <fieldset style="width:250px;height:130px;">
    <table border="0" cellpadding="3" cellspacing="3">
       <tr><td colspan="2" class="header">LOGIN USER</td></tr>
        <tr>
            <td align="right">
                <asp:Label ID="lblUserName" runat="server"
Text="UserName: "></asp:Label>
            </td>
            <td align="left">
```

```
                    <asp:TextBox ID="txtUserName" runat="server"
                    Width="150px"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td align="right">
                    <asp:Label ID="lblPassword" runat="server"
                    Text="Password: "></asp:Label>
                </td>
                <td align="left">
                    <asp:TextBox ID="txtPassword" runat="server"
                    TextMode="Password" Width="150px"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td align="center" colspan="2">
                    <asp:Button ID="btnSubmit" runat="server"
                    Text="Submit"/> 
                    <asp:Button ID="btnReset" runat="server"
                    Text="Reset" />
                </td>
            </tr>
        </table>
        </fieldset>
            <br />
        <div align="center" class="alertmsg">
        </div>

    </div>
        </form>
```

Note that in the markup we just saw, we have defined a div area with class `alertmsg` to display the validation result on the form.

Thus, on page load, the form appears as follows:



4. We will now write a C# class `LoginObj` containing properties `UserName` and `Password` and a method `CheckLogin()` as follows:

```
public class LoginObj
{
    public string UserName { get; set; }
    public string Password { get; set; }

    public bool CheckLogin()
    {
        //Write logic to verify UserName and Password from a
        source (DB/File/Network resource etc.)
        return true;
    }

  public LoginObj()
  {  }
}
```

For the purpose of this demo, we will return `true` as the default validation result.

5. Now, add a web service `Login.asmx` to the project.
6. Add the following to the code-behind to enable the web service to be accessed using AJAX:

```
[System.Web.Script.Services.ScriptService]
```

7. Add a web method to the web service that takes in a single parameter of type `LoginObj`:

```
[WebMethod]
    public bool CheckLogin(LoginObj userLogin)
    {
        bool bReturn = true;

        bReturn = userLogin.CheckLogin();

        return bReturn;

    }
```

We will now pass a parameter of type `LoginObj` to the web service from the jQuery AJAX call.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the event handler for the `click` event of the **Submit** button:

```
$("#btnSubmit").click(function(e) {
```

2. Prevent default form submission:

```
e.preventDefault();
```

3. Check if the input fields are empty and alert the user:

```
if (($("#txtUserName").val() == "") | ($("#txtPassword").val() ==
"""))
     alert("Please enter your login details");
```

4. If the input fields are not empty, call the `CheckLogin()` function:

```
    else
         CheckLogin();
   });
```

5. Define the `CheckLogin()` function:

```
function CheckLogin(){
```

6. Define a login object:

```
var LoginObj = new Object();
```

7. Specify the properties of the object and set the corresponding values:

```
LoginObj.UserName = $("#txtUserName").val();
LoginObj.Password = $("#txtPassword").val();
```

Note that this JavaScript object should have the same properties as the server side `LoginObj` object since we will be mapping this client side object to the corresponding server side object.

8. Define a **Data Transfer Object (DTO)** as follows:

```
var DTO = { 'userLogin': LoginObj };
```

9. Call the `.ajax()` method:

```
$.ajax({
```

10. Specify the HTTP request type as `POST`:

```
type: "POST",
```

11. Specify the URL of the web service:

```
url: "Login.asmx/CheckLogin",
```

12. State the content type of the transfer:

```
contentType: "application/json; charset=utf-8",
```

13. Specify the data type of the response:

```
dataType: "json",
```

14. Use the `.stringify()` function from `json2.js` to convert the JavaScript object to JSON string format:

```
data: JSON.stringify(DTO),
```

Note that the JSON data passed to the web method is received in the input parameter of type `LoginObj` on the server side.

15. Define the callback function for successful AJAX invocation:

```
success: function(msg) {
                        if (msg.d)
                            $(".alertmsg").html("Valid Login");
                        else
                            $(".alertmsg").html("Invalid Login");
                    },
```

16. Define the callback function in case of failure:

```
error: function(xhr, status, error) {
                        alert("An error has occurred during
                        processing: " + error);
                    }
                });
            }
```

17. Define the event handler for the click event of the `Reset` button:

```
$("#btnReset").click(function(e) {
                e.preventDefault();
                $("#txtUserName").val("");
                $("#txtPassword").val("");
                $(".alertmsg").html("");
             });
```

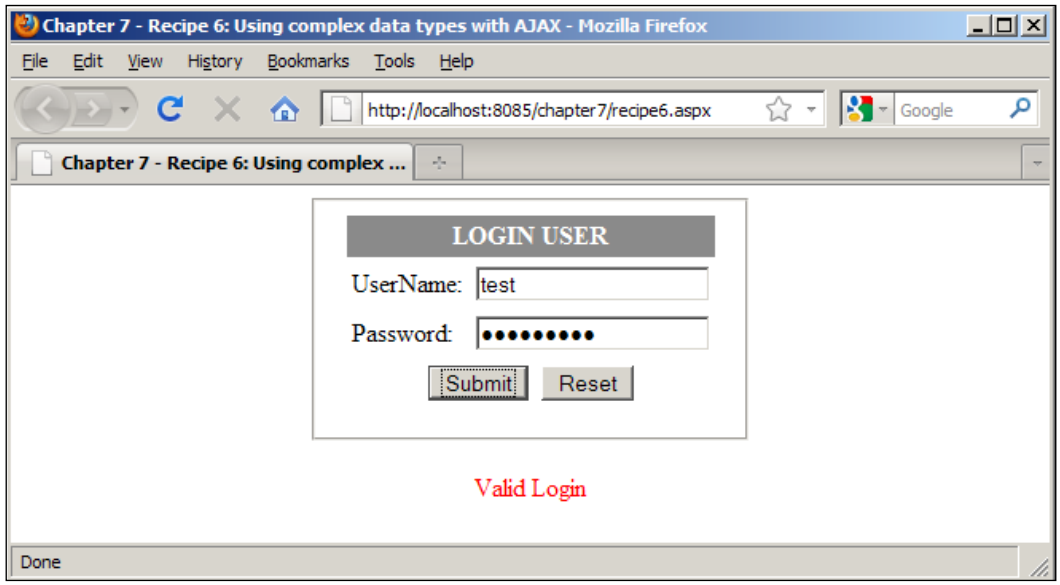
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
        $(document).ready(function() {
            $("#btnSubmit").click(function(e) {
            e.preventDefault();
```

```
            if (($("#txtUserName").val() == "") |
            ($("#txtPassword").val() == ""))
               alert("Please enter your login details");
            else
               CheckLogin();
        });

        function CheckLogin(){
            var LoginObj = new Object();
            LoginObj.UserName = $("#txtUserName").val();
            LoginObj.Password = $("#txtPassword").val();

            //Define DTO i.e. Data Transfer Object as follows
            var DTO = { 'userLogin': LoginObj };

            $.ajax({
                type: "POST",
                url: "Login.asmx/CheckLogin",
                contentType: "application/json; charset=utf-8",
                dataType: "json",
                data: JSON.stringify(DTO),
                success: function(msg) {
                    if (msg.d)
                        $(".alertmsg").html("Valid Login");
                    else
                        $(".alertmsg").html("Invalid Login");
                },
                error: function(xhr, status, error) {
                    alert("An error has occurred during
                    processing: " + error);
                }
            });
        }

        $("#btnReset").click(function(e) {
            e.preventDefault();
            $("#txtUserName").val("");
            $("#txtPassword").val("");
            $(".alertmsg").html("");
        });
    });
</script>
```

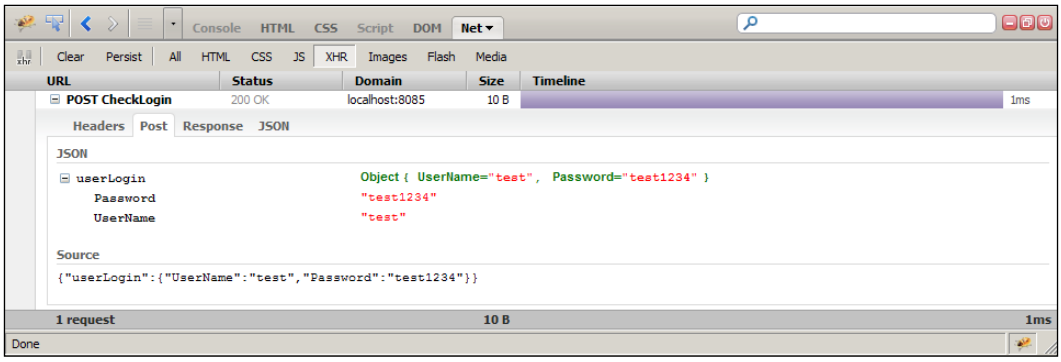
How it works...

Run the page. Key in any random data in the TextBoxes and click the **Submit** button. The page makes the AJAX call as required, passing the UserName and Password in an object and returns the response as follows:

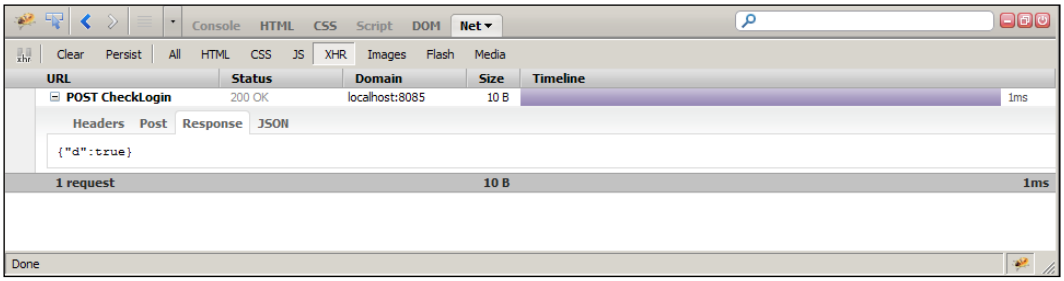


There's more...

Now, launch Firebug in Mozilla Firefox browser to view the AJAX request. The system displays the object passed as a parameter to the web method as follows:



Now, switch to the **Response** tab to see the corresponding response returned from the web method as shown in the following screen:



See also

Catching and displaying AJAX errors

8
Client Templating in jQuery

In this chapter, we will cover:

- Using jQuery Templates to display data
- Displaying arrays using nested templates
- Calling JavaScript functions and conditional statements in templates
- Creating an ASP.NET CheckBoxList in templates
- Using templates to format the data returned from web services
- Using callback functions with jQuery templates
- Calling external templates using jQuery

Introduction

There are many client templating solutions available in jQuery. Among them, jTempates is a popular plugin and has been extensively used in templating applications in the past.

However, with Microsoft's commitment to contribute to jQuery and the ease of integrating its templating solution in applications, Microsoft's jQuery templating plugin has now been accepted as an official jQuery plugin.

From jQuery 1.5 onwards, this templating solution will be part of the core engine, eliminating the need for external templating solutions.

The official repository of the plugin is located at `https://github.com/jquery/jquery-tmpl`. The beta 1 version of the plugin is also located on Microsoft CDN (Content Delivery Network) at:

- `http://ajax.microsoft.com/ajax/jquery.templates/beta1/jquery.tmpl.js`
- `http://ajax.microsoft.com/ajax/jquery.templates/beta1/jquery.tmpl.min.js`

Getting started

1. Let's start by creating a new ASP.NET website in Visual Studio and name it `Chapter8`.
2. Save the jQuery library in a script folder `js` in the project.
3. Download the jQuery template plugin from `https://github.com/jquery/jquery-tmpl`. Save it to the `js` folder.
4. For the recipes described in this chapter, in addition to including the jQuery library on ASP.NET pages, the template plugin should also be included as shown:

```
<script src="js/jquery-1.4.1.js" type="text/javascript"/>
<script src="js/jquery.tmpl.js" type="text/javascript"/>
```

There are basically two ways of defining templates:

- Inline templates defined in the ASPX page within `<script type="text/x-jquery-tmpl">` blocks
- External templates defined in `*.htm/*.html/*.txt` files

Let's take a look at some of the important methods/template tags that we will be using in this chapter:

- **.tmpl([data],[options])**: This method applies the data to the first matched element and renders the contents as a template. It takes in two parameters:
  - **data**: This is the data to be rendered. It can be either an array or an object.
  - **options**: This is an optional map of key-value pairs.

  The `.tmpl()` method can be chained with `.appendTo()`, `.prependTo()`, `.insertAfter()`, or `.insertBefore()` to add the contents to the DOM as required.
- **jQuery.tmpl(template, [data], [options])**: This method helps to render the specified data to the HTML markup. It takes in three parameters:
  - **template**: This is the HTML markup to be used as a template.
  - **data**: This is the data to be rendered. It can be either an array or an object.

- **options**: This is an optional map of key-value pairs.

- **{{tmpl([data], [options]) template}} content {{/tmpl}}**: The {{tmpl}} template tag is used with nested templates. It is used to render the child template within the content of the parent template. It takes in three parameters:
  - **data**: This is the data to be rendered. It can be either an array or an object.
  - **options**: This is an optional map of key-value pairs.
  - **template**: This is the HTML markup or text to be rendered as a template.
- **${ fieldNameOrExpression } Template tag**: This tag is used to evaluate a JavaScript field or expression when data is applied to the template for rendering.
- **{{each}} and {{/each}} Template tags**: These tags are used to iterate over a data collection. The contents between the opening and the closing tags is rendered for each data item in the collection.
- **{{if}}, {{else}}, {[/if}} Template tags**: These tags are used to evaluate conditional statements in the template block. The content between the opening and closing tags is rendered only if the condition is satisfied.

Now, let's take a look at the different applications of this templating solution in ASP.NET websites.

Using jQuery Templates to display data

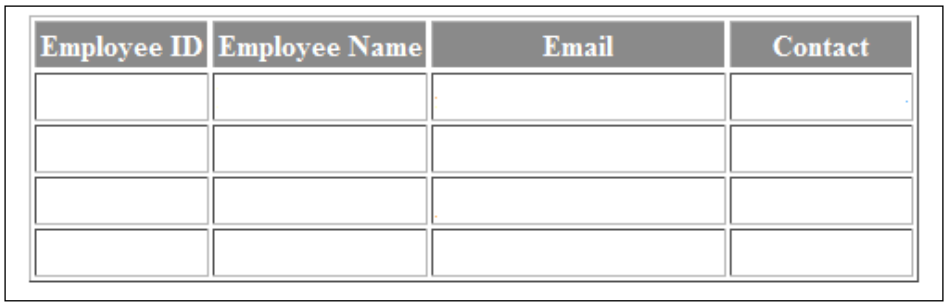
Let's begin with the basics of defining inline templates in ASP.NET pages. We will create some sample data and then apply the same to the template.

Getting ready

1. Add a new web form `Recipe1.aspx` to the current project.
2. We will define an Employee array on the client side containing data fields such as `EmployeeID`, `Name`, `Email`, and `Contact`. A snippet of the array structure is as follows:

```
var EmpData = [
{ EmpID: "I53434",
  Name: "Richard Tan",
  Email: "richard@someemail.com",
  Contact: "+65 24242444"}
];
```

3. We aim to display this data using templates in the following format:



4. Define the corresponding ASPX markup to define the previous table structure as follows:

```
<form id="form1" runat="server">
    <div align="center">
       <table border="1" cellspacing="2" cellpadding="2"
id="contentTble">
       <tr class="header">
       <td>Employee ID
       </td><td>Employee Name</td>
       <td>Email</td>
      <td>Contact</td>
      </tr>
</table>
    </div>
    </form>
```

5. We will now define the inline template to be used to render the data for each Employee record in the array. Note that each record corresponds to a table row in the rendered content:

```
<script type="text/x-jquery-tmpl" id="empTemplate">
<tr>
   <td>${EmpID}</td>
   <td>${Name}</td>
   <td>${Email} </td>
   <td>${Contact}</td>
</tr>
</script>
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the Employee array containing the sample data:

```
var EmpData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: "+65 24242444"},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.
com", Contact: "+65 8664664"},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.
com", Contact: "+65 94643646"},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@
someemail.com", Contact: "+65 4225252"}
];
```

2. Apply the array data to the template and append to the containing table on the page:

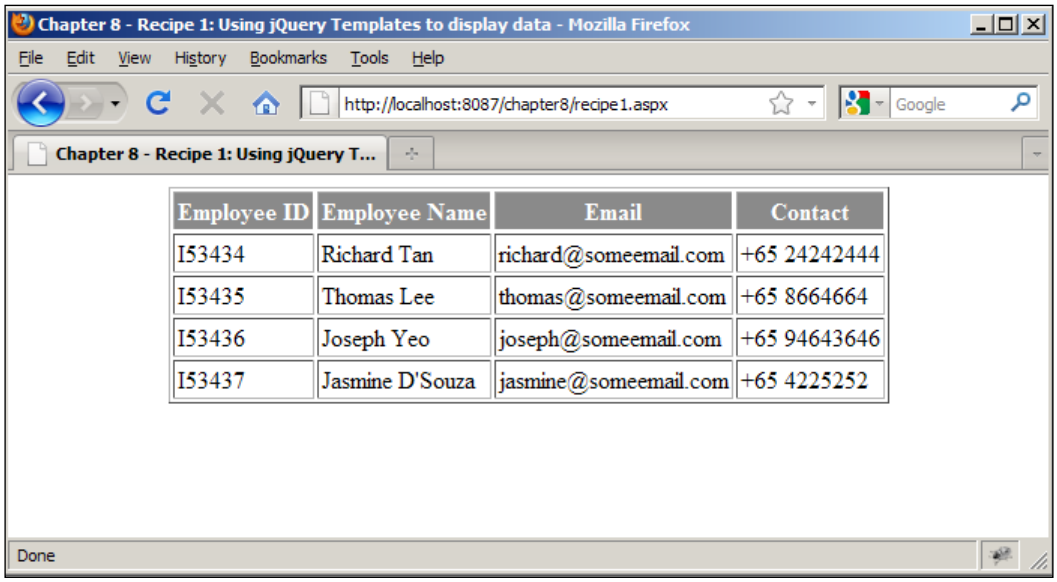
```
$("#empTemplate").tmpl(EmpData).appendTo("#contentTble");
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
var EmpData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: "+65 24242444"},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.
com", Contact: "+65 8664664"},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.
com", Contact: "+65 94643646"},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@
someemail.com", Contact: "+65 4225252"}
];
            $("#empTemplate").tmpl(EmpData).
appendTo("#contentTble");
});
</script>
```

How it works...

Run the web form. The page will display the formatted content as shown in the following screenshot:



Employee ID	Employee Name	Email	Contact
I53434	Richard Tan	richard@someemail.com	+65 24242444
I53435	Thomas Lee	thomas@someemail.com	+65 8664664
I53436	Joseph Yeo	joseph@someemail.com	+65 94643646
I53437	Jasmine D'Souza	jasmine@someemail.com	+65 4225252

See also

Displaying arrays using nested templates

Displaying arrays using nested templates

For simple markups, a single template is usually sufficient. However, in certain scenarios requiring complex markup, we might need to use nested templates. Let's see an example of calling nested inline templates on an ASP.NET page.

Getting ready

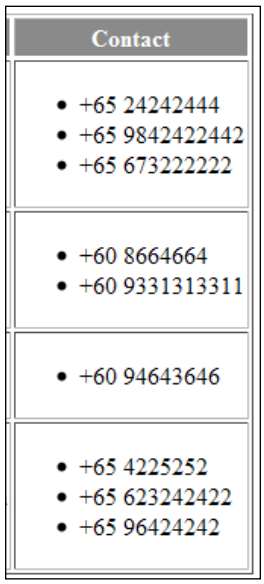
1. Add a new web form `Recipe2.aspx` to the current project.
2. We will define a client side Employee array containing data fields such as `EmployeeID`, `Employee Name`, `Email`, `Contact`, and `Country code`. In turn, we will define multiple Contact numbers for each employee so that the Contact field is also defined as an array as shown next:

```
Contact: ["24242444","9842422442","673222222"]
```

3. A short snippet of the Employee array to be used in this example is as follows:

```
var EmployeeData = [
{ EmpID: "I53434",
  Name: "Richard Tan",
  Email: "richard@someemail.com",
  Contact: ["24242444","9842422442","673222222"],
  CountryCode: "65"}
];
```

4. We aim to display the Contact column as shown next:



5. Add the following ASPX markup to the page and define the containing table:

```
<form id="form1" runat="server">
    <div align="center">
       <table border="1" cellspacing="2" cellpadding="2"
id="contentTble">
       <tr class="header">
         <td>Employee ID</td>
         <td>Employee Name</td>
         <td>Email</td>
         <td>Contact</td>
       </tr>
       </table>
</div>
</form>
```

6. Let's define a child template for formatting the Contact numbers as an unordered HTML list. Each Contact number is concatenated with the respective Country code:

```
<script type="text/x-jquery-tmpl" id="contactTemplate">
    <ul>
       {{each Contact}}
           <li>+${CountryCode} ${$value}</li>
       {{/each}}
    </ul>
</script>
```

The jQuery template tag `${$value}` retrieves the value of the current item in the `{{each}}..{//each}}` loop.

7. We will now define a parent template to display the remaining fields (`EmployeeID`, `Employee Name`, and `Email`). For each record of the parent template, the child template will be rendered using the `{{tmpl}}` tag:

```
<script type="text/x-jquery-tmpl" id="employeeTemplate">
       <tr>
          <td>${EmpID}</td>
          <td>${Name}</td>
          <td>${Email} </td>
          <td align="left">
            {{tmpl "#contactTemplate"}}
          </td>
       </tr>
</script>
```

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the client side array containing the Employee data such that the Contact number field is defined as a nested array as shown:

```
var EmployeeData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: ["24242444","9842422442","673222222"], CountryCode:
"65"},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.
com", Contact: ["8664664","9331313311"], CountryCode: "60"},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.
com", Contact: ["94643646"], CountryCode: "60"},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@
someemail.com", Contact: ["4225252","623242422", "96424242"],
CountryCode: "65"}
];
```

2. Apply the array data to the template and append the rendered contents to the containing table on the page:

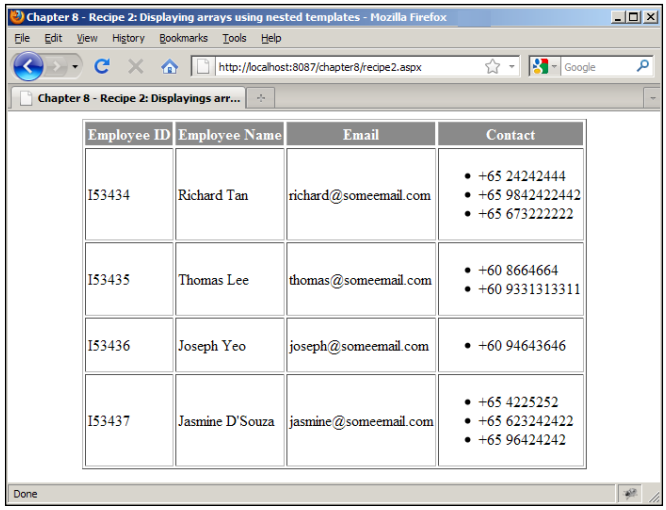
```
$("#employeeTemplate").tmpl(EmployeeData).
appendTo("#contentTble");
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
var EmployeeData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: ["24242444","9842422442","673222222"], CountryCode:
"65"},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.
com", Contact: ["8664664","9331313311"], CountryCode: "60"},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.
com", Contact: ["94643646"], CountryCode: "60"},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@
someemail.com", Contact: ["4225252","623242422", "96424242"],
CountryCode: "65"}
];
            $("#employeeTemplate").tmpl(EmployeeData).
appendTo("#contentTble");
});
</script>
```

How it works...

Run the web form. The page will display the employee data with the formatted contact numbers as shown in the following screenshot:



Employee ID	Employee Name	Email	Contact
I53434	Richard Tan	richard@someemail.com	• +65 24242444 • +65 9842422442 • +65 673222222
I53435	Thomas Lee	thomas@someemail.com	• +60 8664664 • +60 9331313311
I53436	Joseph Yeo	joseph@someemail.com	• +60 94643646
I53437	Jasmine D'Souza	jasmine@someemail.com	• +65 4225252 • +65 623242422 • +65 96424242

See also

Using jQuery Templates to display data

Calling JavaScript functions and conditional statements in templates

A useful feature of jQuery templates is the calling of JavaScript functions from within the template code. In addition, conditional statements can also be executed within the template using {{if}}...{{else}}...{{/if}} template tags.This enables us to accomplish certain data/content-specific tasks as demonstrated in this recipe.

Getting ready

1. Add a new web form `Recipe3.aspx` to the current project.
2. We will reuse the employee array containing `EmpID`, `Name`, `Email`, and `Contact` defined in the earlier recipe. A short snippet is shown next:

```
var EmployeeData = [
{ EmpID: "I53434",
  Name: "Richard Tan",
  Email: "richard@someemail.com",
  Contact: ["+65 24242444","+65 9842422442","+65 673222222"]
}
];
```

3. In addition, we will dynamically create a new column to display the total number of contact numbers available for each employee by calling a JavaScript function in the template code. Thus, we can define the ASPX markup as follows:

```
<form id="form1" runat="server">
    <div align="center">
       <table border="1" cellspacing="2" cellpadding="2"
id="contentTble">
       <tr class="header">
         <td>Employee ID</td>
         <td>Employee   Name</td>
         <td>Email</td>
         <td>Contact</td>
         <td>Available Contact Numbers</td>
       </tr>
       </table>
    </div>
    </form>
```

4. We will also use conditional statements to check the length of the `Contact` column in the template code as follows:

```
{{if Contact.length > 0}}
            ${Contact}
{{else}}-
{{/if}}
```

5. Assuming that the JavaScript function `getCount()` retrieves the total number of contact numbers for each employee, the function can be called in the template code as follows:

```
${getCount()}
```

Thus, the complete template code is shown next:

```
<script type="text/x-jquery-tmpl" id="employeeTemplate">
       <tr><td>${EmpID}</td>
       <td>${Name}</td>
       <td>${Email} </td>
       <td align="center">
       {{if Contact.length > 0}}${Contact}
       {{else}}-{{/if}}
       </td>
       <td align="center">
       ${getCount()}
       </td></tr>
</script>
```

How to do it...

1. In the JavaScript block, define the function `getCount()` to return the length of the Contact array:

```
function getCount()
           {
              var retval = this.data.Contact.length;
              return ((retval>0)? retval:"No contact numbers are
available");
           }
```

   The function returns a message **No contact numbers are available** if the length of the Contact array is 0 for any Employee record.

2. In the `document.ready()` function of the jQuery script block, define the Employee array:

```
var EmployeeData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: ["+65 24242444","+65 9842422442","+65 673222222"]},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.
com", Contact: ["+65 8664664","+65 9331313311"]},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.
com", Contact: []},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@
someemail.com", Contact: ["+65 4225252","+65 623242422", "+65
96424242"]}
];
```

3. Apply the array to the template `employeeTemplate` and append to the `contentTble` table on the form:

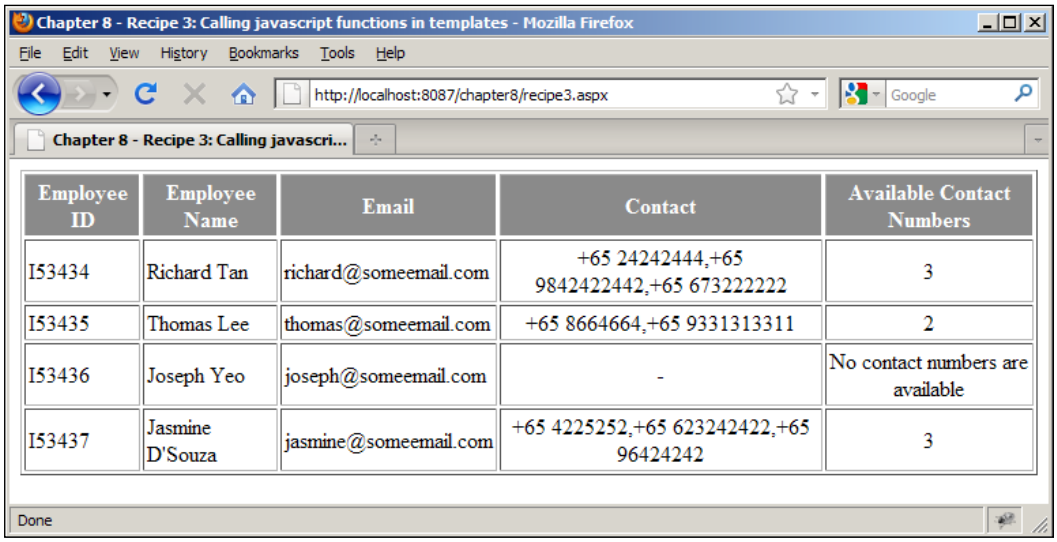
```
$("#employeeTemplate").tmpl(EmployeeData).
appendTo("#contentTble");
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
var EmployeeData = [
{ EmpID: "I53434", Name: "Richard Tan", Email: "richard@someemail.
com", Contact: ["+65 24242444","+65 9842422442","+65 673222222"]},
{ EmpID: "I53435", Name: "Thomas Lee", Email: "thomas@someemail.com",
Contact: ["+65 8664664","+65 9331313311"]},
{ EmpID: "I53436", Name: "Joseph Yeo", Email: "joseph@someemail.com",
Contact: []},
{ EmpID: "I53437", Name: "Jasmine D'Souza", Email: "jasmine@someemail.
com", Contact: ["+65 4225252","+65 623242422", "+65 96424242"]}
];
          $("#employeeTemplate").tmpl(EmployeeData).
appendTo("#contentTble");
});
function getCount()
{
    var retval = this.data.Contact.length;
    return ((retval>0)? retval:"No contact numbers are available");
}
</script>
```

How it works...

Run the web form. The page displays the formatted employee data as shown in the following screenshot:



Employee ID	Employee Name	Email	Contact	Available Contact Numbers
I53434	Richard Tan	richard@someemail.com	+65 24242444,+65 9842422442,+65 673222222	3
I53435	Thomas Lee	thomas@someemail.com	+65 8664664,+65 9331313311	2
I53436	Joseph Yeo	joseph@someemail.com	-	No contact numbers are available
I53437	Jasmine D'Souza	jasmine@someemail.com	+65 4225252,+65 623242422,+65 96424242	3

Note that the data in the `Available Contact Numbers` column is computed dynamically in the template code based on the data in the `Contact` column.

See also

Using templates to format data returned from web services

Creating an ASP.NET CheckBoxList in templates

In this recipe, we will use templates to create a CheckBoxList on an ASPX page. The CheckBoxList will be added dynamically at runtime. We will also add a `Select All` checkbox to select/deselect all items.

Getting ready

1. Add a new web form `Recipe4.aspx` to the current project.

2. We will add a client side array for a Book list consisting of the fields `BookTitle` and `Author`. A short snippet of the array is as follows:

```
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"}
];
```

3. Add a containing table to the web form. Add a **Select All** checkbox to the form. Thus, the ASPX markup of the form is as follows:

```
<form id="form1" runat="server">
<div align="center">
<table border="0" cellpadding="0" cellspacing="0"
id="contentTble">
<tr>
<td align="left"><asp:CheckBox ID="chkSelectAll" runat="server"
Text="Select All" />
</td>
</tr>
</table>
</div>
</form>
```

4. Add an inline template to the page. The HTML markup of the template consists of a CheckBox item in a table row. The text of the Checkbox item consists of the `Author`, `BookTitle` fields from the array as follows:

```
<script type="text/x-jquery-tmpl" id="bookTemplate">
    <tr>
    <td align="left">
    <input type="checkbox" ID="CheckBoxList1">${Author},${BookTit
le}</input>
    </td>
    </tr>
</script>
```

Since each CheckBox item is created with the same `CheckBoxList1` ID, they form the CheckBoxList collection on the page. Next, we will apply the client side array to the template.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the Book array as follows:

```
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"},
{Author:"Paulo Coelho", BookTitle: "Alchemist"},
```

```
{Author:"Barack Obama", BookTitle: "The Audacity of Hope"},
{Author:"Richard Bach", BookTitle: "One"}
];
```

2. Apply the array to the template. Append the rendered contents to the containing table on the page:

```
$("#bookTemplate").tmpl(bookArr).appendTo("#contentTble");
```

3. Next, we will add the **select/unselect all** property to the `Select All` CheckBox. Hence, define the callback function on the `click` event of this CheckBox:

```
$("#chkSelectAll").click(function() {
```

4. Check if the Select All CheckBox is selected using `$('#chkSelectAll').is(':checked')`.

The jQuery `:checked` selector matches all checked checkboxes/radio buttons on the form.

Accordingly, add this attribute to the CheckBoxList:

```
$("INPUT[type='checkbox']").attr('checked', $('#chkSelectAll').
is(':checked'));
                  });
```

Thus, the items in the CheckBoxList will be checked if the **Select All** CheckBox is checked. Similarly, the items will be unchecked if the **Select All** CheckBox is unchecked.

The jQuery selector `$("INPUT[type=checkbox"])` retrieves all input elements of type `checkbox` on the form.

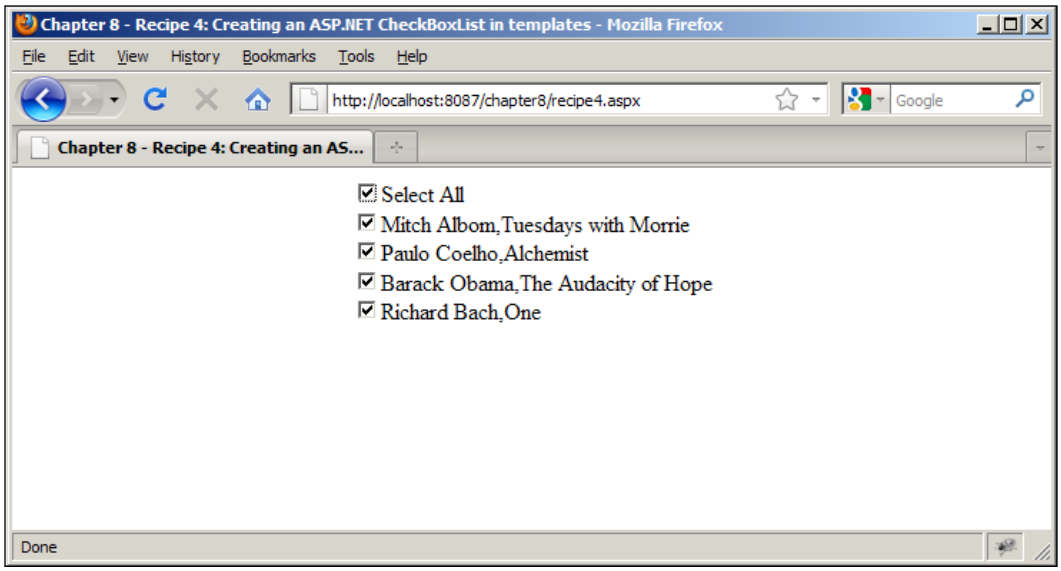
Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"},
{Author:"Paulo Coelho", BookTitle: "Alchemist"},
{Author:"Barack Obama", BookTitle: "The Audacity of Hope"},
{Author:"Richard Bach", BookTitle: "One"}
];
              $("#bookTemplate").tmpl(bookArr).
appendTo("#contentTble");
$("#chkSelectAll").click(
         function() {
```

```
                                                $("INPUT[type='checkbox']").
    attr('checked', $('#chkSelectAll').is(':checked'));
                        });
    });
    </script>
```

How it works...

Run the web form. The page displays the CheckBoxList as shown in the following screenshot:



Using the **Select All** checkbox, all items in the CheckBoxList can be checked / unchecked as required.

See also

Calling external templates using jQuery

Using templates to format data returned from web services

A useful application of jQuery templates is to format JSON data returned from web services. In the absence of a templating solution, the JSON data has to be painfully parsed and formatted in the jQuery code. This requires code maintenance in case of format changes.

jQuery templates eliminate the need to include formatting information within the jQuery code. In this recipe, let's build a web service and retrieve JSON data from the same using an AJAX call. We will then apply the data to an inline template.

Getting ready

1. Create a class `Employee` consisting of properties such as `EmpID`, `Name`, `Email`, `Contact` as shown next:

```
public class Employee
{
    public string EmpID { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string Contact { get; set; }
}
```

2. Add a new web service `EmployeeService.asmx` to the current project. Include the following in the web service code-behind to enable the service to be called using AJAX:

```
[System.Web.Script.Services.ScriptService]
```

3. Instantiate a list of type Employee in the code-behind:

```
List<Employee> empList = new List<Employee>
{
  new Employee{EmpID="I53434",Name="Richard Tan",Email="richard@
someemail.com",Contact="+65 24242444"},
  new Employee{EmpID="I53435",Name="Thomas Lee",Email="thomas@
someemail.com",Contact="+65 8664664"},
  new Employee{EmpID="I53436",Name="Joseph Yeo",Email="joseph@
someemail.com",Contact="+65 94643646"},
  new Employee{EmpID="I53437",Name="Jasmine
D'Souza",Email="jasmine@someemail.com",Contact="+65 4225252"},
};
```

4. Add a web method to the web service to return this list:

```
[WebMethod]
    public List<Employee> GetEmployeeData() {
        return empList;
    }
```

5. Add a new web form `Recipe5.aspx` to the current project.

6. Add the following css style on the form to hide elements on the page:

```
.hide
    {
      display:none;
    }
```

7. Add the following markup to display the Employee list on the form. Also add a button control to trigger the AJAX call on the page:

```
<form id="form1" runat="server">
<div align="center">
        <asp:Button ID="btnSubmit" runat="server" Text="Click to
retrieve Employee data" />
        <br /><br />
        <table border="1" cellspacing="2" cellpadding="2"
id="contentTble">
       <tr class="header">
       <td>Employee ID</td>
       <td>Employee Name</td>
       <td>Email</td>
       <td>Contact</td>
       </tr>
       </table>
</div>
</form>
```

8. Add the following inline template such that each table row will display an item from the Employee list:

```
<script type="text/x-jquery-tmpl" id="empTemplate">
       <tr>
       <td>${EmpID}</td>
       <td>${Name}</td>
       <td>${Email} </td>
       <td>${Contact}</td>
       </tr>
</script>
```

We will make an AJAX call to the web service from the ASPX page using jQuery AJAX. The JSON data returned from the web service will then be applied to the template and displayed on the form.

How to do it...

1. In the `document.ready()` function of the jQuery script block, hide the containing table on the form:
   ```
   $("#contentTble").addClass("hide");
   ```
2. Define the callback function on the `click` event of the button control:
   ```
   $("#btnSubmit").click(function(e){
   ```
3. Prevent default form submission:
   ```
   e.preventDefault();
   ```
4. Call the `sendData()` function to make the AJAX call:
   ```
   sendData();
         });
   ```
5. Define the `sendData()` function:
   ```
   function sendData() {
   ```
6. Call the `.ajax()` method:
   ```
   $.ajax({
   ```
7. Specify the type of HTTP request as `POST`:
   ```
   type: "POST",
   ```
8. Specify the URL of the web method to call:
   ```
   url: "EmployeeService.asmx/GetEmployeeData",
   ```
9. Specify the content type of the AJAX request:
   ```
   contentType: "application/json; charset=utf-8",
   ```
10. Specify the data type of the response:
    ```
    dataType: "json",
    ```
11. Define the callback function on successful invocation. The AJAX response is received in the object `msg`:
    ```
    success: function(msg) {
    ```
12. Unhide the containing table:
    ```
    $("#contentTble").removeClass("hide");
    ```
13. Apply the JSON response `msg.d` to the template. Append the rendered data to the containing table:
    ```
    $("#empTemplate").tmpl(msg.d).appendTo("#contentTble");
    ```

14. Hide the button control on the form:

```
$("#btnSubmit").addClass("hide");
                    },
```

15. Define the callback function on failure:

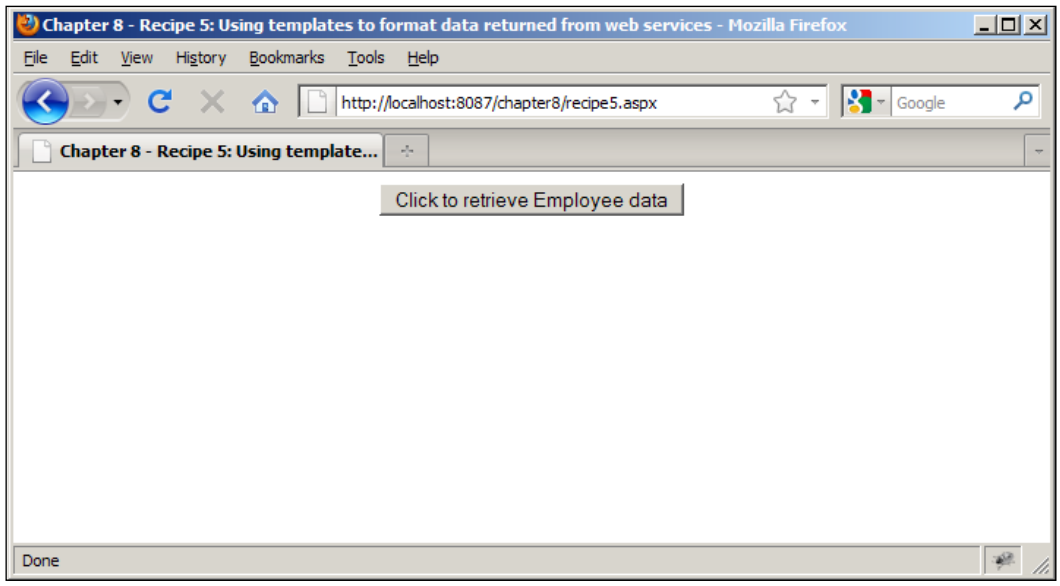
```
error: function() {
 alert("An unexpected error has occurred during processing.");
                    }
                });
            }
```

Thus, the complete jQuery solution is as follows:

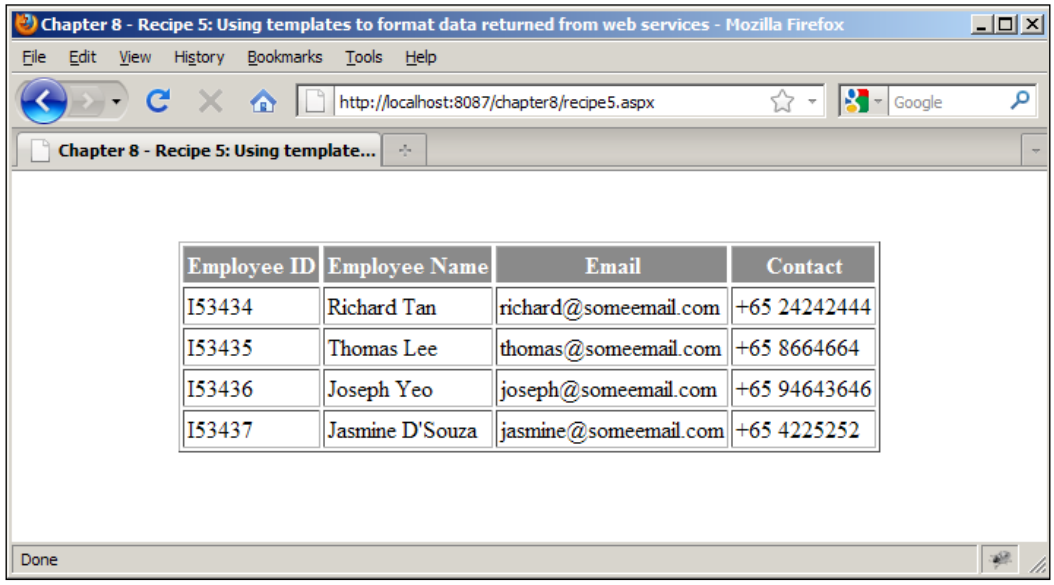
```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
      $("#contentTble").addClass("hide");
      $("#btnSubmit").click(function(e){
             e.preventDefault();
             sendData();
       });

      function sendData() {
       $.ajax({
         type: "POST",
         url: "EmployeeService.asmx/GetEmployeeData",
         contentType: "application/json; charset=utf-8",
         dataType: "json",
         success: function(msg) {
            $("#contentTble").removeClass("hide");
                     $("#empTemplate").tmpl(msg.d).
appendTo("#contentTble");
            $("#btnSubmit").addClass("hide");
          },
          error: function() {
 alert("An unexpected error has occurred during processing.");
          }
       });
     }
  });
</script>
```

How it works...

Run the web form. On page load, the form displays only the button control as follows:



Now click the button control. The page makes an AJAX call and displays the formatted employee data as shown in the following screenshot:

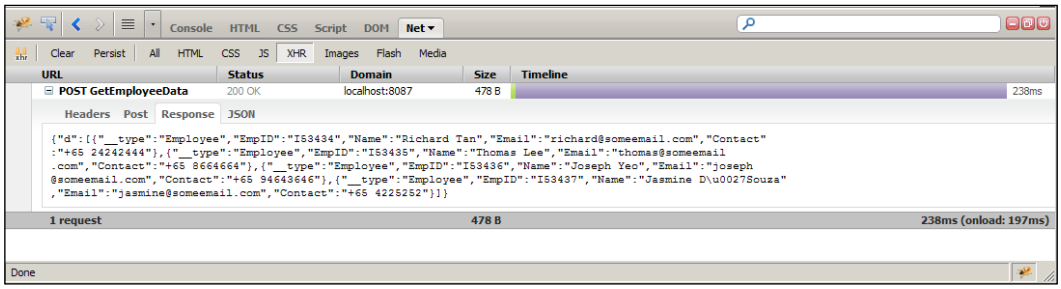


Employee ID	Employee Name	Email	Contact
I53434	Richard Tan	richard@someemail.com	+65 24242444
I53435	Thomas Lee	thomas@someemail.com	+65 8664664
I53436	Joseph Yeo	joseph@someemail.com	+65 94643646
I53437	Jasmine D'Souza	jasmine@someemail.com	+65 4225252

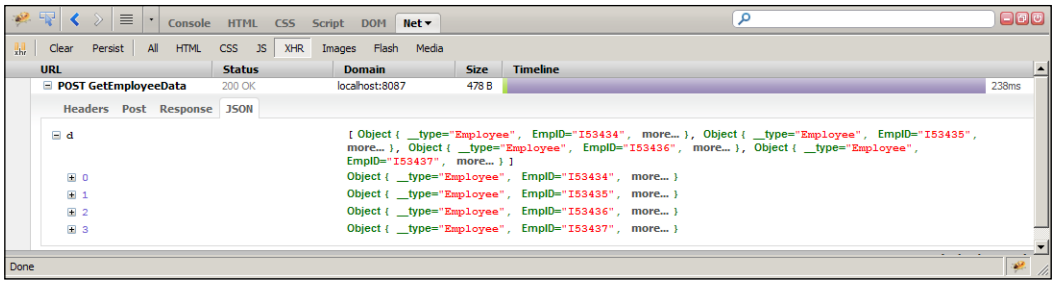
There's more...

Let's use Firebug in the Mozilla Firefox browser to view the JSON response retrieved from the web service. Load the page and click the button control to make the AJAX call. Open the Firebug window and click on the **Net | XHR** tab.

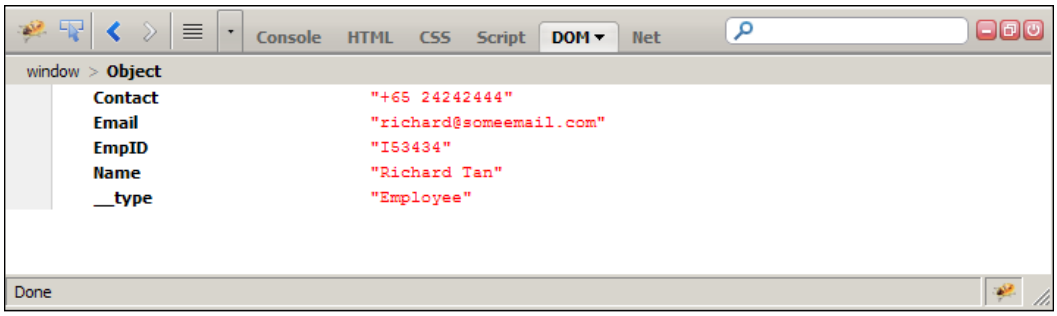
Click on the **Response** sub-tab to see the web service response as shown next:



Next, click on the **JSON** sub-tab to view the JSON formatted string as follows:



On clicking on the **more...** link, we can see the details of the individual data row as shown in the following screenshot:



See also

Using callback functions with jQuery templates

Using callback functions with jQuery templates

jQuery templates allow us to execute callback functions for further processing of the formatted contents before rendering it on the page. In this recipe, let's see how we can use the callback function to highlight alternate rows of a table. We will use the web services example described earlier and execute a callback function for each item in the returned response object.

Getting ready

1. Add a new web form `Recipe6.aspx` to the current project.
2. Add a css class to hide elements on the form:

```
.hide
    {
      display:none;
    }
```

3. Add a css class to highlight the background color of a row:

```
.highlight
    {
      background-color:Silver;
    }
```

4. Add a button control to the form to trigger the AJAX request. Also add the following markup to the form to render the data received from the web service.

```
<form id="form1" runat="server">
<div align="center">
<asp:Button ID="btnSubmit" runat="server" Text="Click to retrieve
Employee data" />
        <br /><br />
        <table border="1" cellspacing="2" cellpadding="2"
id="contentTble">
       <tr class="header">
       <td>Employee ID</td>
       <td>Employee Name</td>
       <td>Email</td>
       <td>Contact</td>
       </tr>
```

```
        </table>
</div>
</form>
```

5. Define an inline template to format the JSON data returned from the web service:

```
<script type="text/x-jquery-tmpl" id="empTemplate">
        <tr>
        <td>${EmpID}</td>
        <td>${Name}</td>
        <td>${Email} </td>
        <td>${Contact}</td>
        </tr>
</script>
```

We will make an AJAX call to the web service `EmployeeService.asmx` described in the earlier recipe and execute a callback function on each item in the response object.

How to do it...

1. In the `document.ready()` function of the jQuery script block, hide the containing table on the form:

```
$("#contentTble").addClass("hide");
```

2. Define the callback function on the `click` event of the button control:

```
$("#btnSubmit").click(function(e){
```

3. Prevent default form submission:

```
e.preventDefault();
```

4. Call the `sendData()` function to make the AJAX call:

```
sendData();
      });
```

5. Define the `sendData()` function:

```
function sendData() {
```

6. Call the `.ajax()` method:

```
$.ajax({
```

7. Specify the type of HTTP request as `POST`:

```
type: "POST",
```

8. Specify the URL of the web method to call:

```
url: "EmployeeService.asmx/GetEmployeeData",
```

9. Specify the content type of the AJAX request:

```
contentType: "application/json; charset=utf-8",
```

10. Specify the data type of the response:

```
dataType: "json",
```

11. Define the callback function on successful invocation. The AJAX response is received in the object `msg`:

```
success: function(msg) {
```

12. In the callback function, unhide the containing table:

```
$("#contentTble").removeClass("hide");
```

13. Apply the JSON response `msg.d` to the template. Call the `.each()` selector on each rendered item:

```
$("#empTemplate").tmpl(msg.d).each(
```

14. Define the callback function to be executed on the item. Pass the item and item index as parameters:

```
function(index,item){
```

15. In this callback function, check if the item index is even. If yes, then add the css class `highlight` to the item:

```
if (index % 2)
      $(this).addClass("highlight");
```

16. Append the rendered data to the containing table:

```
}).appendTo("#contentTble");
```

17. Hide the button control on the form:

```
$("#btnSubmit").addClass("hide");
                    },
```

18. Define the callback function on failure:

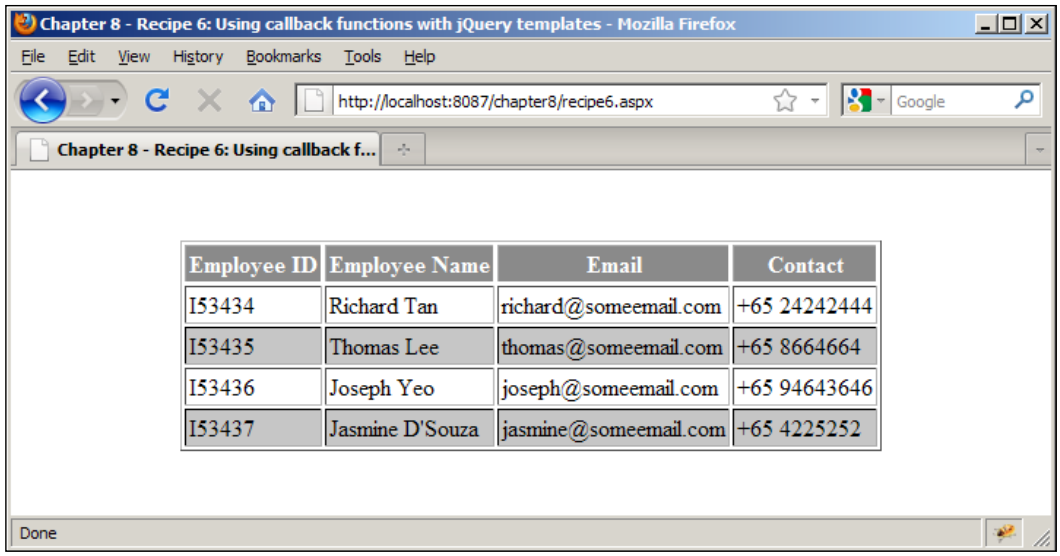
```
error: function() {
                        alert("An unexpected error has occurred
during processing.");
                    }
                });
            }
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
           $("#contentTble").addClass("hide");
           $("#btnSubmit").click(function(e){
                e.preventDefault();
                sendData();
          });

          function sendData() {
           $.ajax({
             type: "POST",
             url: "EmployeeService.asmx/GetEmployeeData",
             contentType: "application/json; charset=utf-8",
             dataType: "json",
             success: function(msg) {
                    $("#contentTble").removeClass("hide");
                    $("#empTemplate").tmpl(msg.d).each(
                       function(index,item){
                         if (index % 2)
                            $(this).addClass("highlight");
                     }).appendTo("#contentTble");
                     $("#btnSubmit").addClass("hide");
                    },
                    error: function() {
 alert("An unexpected error has occurred during processing.");
                  }
            });
         }
     });
</script>
```

How it works...

Run the web form. Click on the button control to post the AJAX request to the web service. The page will display the employee data with the even numbered rows highlighted, as shown in the following screenshot:



Employee ID	Employee Name	Email	Contact
I53434	Richard Tan	richard@someemail.com	+65 24242444
I53435	Thomas Lee	thomas@someemail.com	+65 8664664
I53436	Joseph Yeo	joseph@someemail.com	+65 94643646
I53437	Jasmine D'Souza	jasmine@someemail.com	+65 4225252

See also

Using templates to format data returned from web services

Calling external templates using jQuery

So far, we have used inline templates in our ASP.NET pages. However, in real world applications, sometimes architectural constraints might not permit formatting information in source files, thus necessitating segregation of template contents from the ASPX files. In a way, this makes maintenance of templates easier and eliminates the need to alter source files whenever there is any formatting change.

In this recipe, we will demonstrate the use of an external `*.htm` file to store the template contents. We will call the same using an AJAX request.

In addition to `*.htm`, it is also possible to maintain the template contents in any other flat file format such as `*.txt`.

Getting ready

1. Add a new web form `Recipe7.aspx` to the current project.
2. We will define a Book array consisting of data fields `Author` and `BookTitle`. A short snippet of the array is as follows:

```
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"}
];
```

3. Add the following ASPX markup to the web form:

```
<form id="form1" runat="server">
    <div align="center" >
    <ul id="contentArea">
    </ul>
    </div>
</form>
```

4. We aim to display the array contents as ListItems in the unordered list `contentArea`.
5. Define the following css style for formatting an unordered list on the form:

```
ul
    {
      text-align:left;
    }
```

6. Define an external template file `SampleTemplate.htm` with the following formatting information:

```
<li>${Author}, ${BookTitle}</li>
```

Note that the template file is defined without the `<script type="text/x-jquery-tmpl">` block since we intend to read the template contents in a string and apply the data array to the resultant markup string using the `$.tmpl(markup_string,data)` function.

We will use the jQuery function `$.get()` to make an AJAX HTTP GET request to load the contents from the template file.

How to do it...

1. In the `document.ready()` function of the jQuery script block, define the Book array as follows:

```
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"},
{Author:"Paulo Coelho", BookTitle: "Alchemist"},
{Author:"Barack Obama", BookTitle: "The Audacity of Hope"},
{Author:"Richard Bach", BookTitle: "One"}
];
```

2. Use `$.get()` and provide the template file URL. Define a callback function on receiving the response object `msg`:

```
$.get("SampleTemplate.htm",function(msg){
```

3. Use the `$.tmpl(markup_string,data)` function. Provide the response `msg.d` as the markup string and the Book array as the data to be rendered. Append the rendered contents to the `contentArea` element on the form:

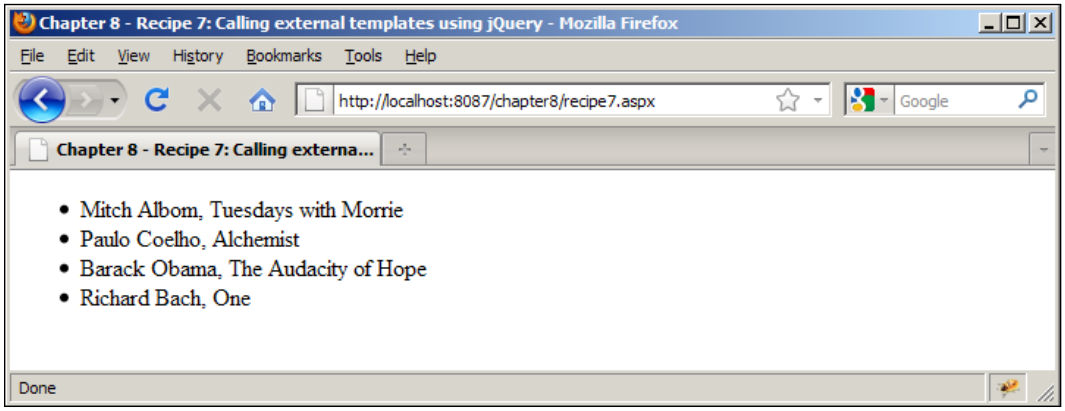
```
$.tmpl(msg,bookArr).appendTo("#contentArea");
            });
```

Thus, the complete jQuery solution is as follows:

```
<script language="javascript" type="text/javascript">
$(document).ready(function() {
var bookArr = [
{Author: "Mitch Albom", BookTitle: "Tuesdays with Morrie"},
{Author:"Paulo Coelho", BookTitle: "Alchemist"},
{Author:"Barack Obama", BookTitle: "The Audacity of Hope"},
{Author:"Richard Bach", BookTitle: "One"}
];
  $.get("SampleTemplate.htm",function(msg){
      $.tmpl(msg,bookArr).appendTo("#contentArea");
     });
});
</script>
```

How it works...

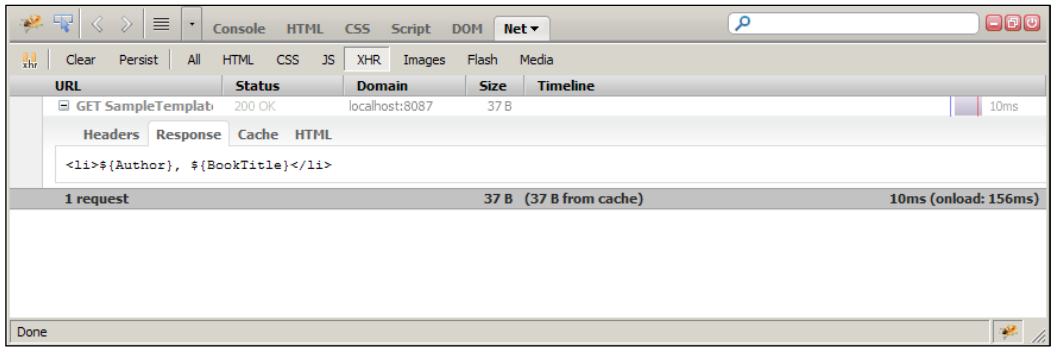
Run the web form. The page displays the Book Title/Author information formatted as per the template as shown in the following screenshot:



There's more...

Let's view the AJAX response using Firebug in the Mozilla Firefox browser. Run the page in Firefox and launch the Firebug window.

Click on the **Net | XHR** tab. Subsequently, click the **Response** sub-tab to view the HTML response as shown in the following screenshot:



See also

Creating an ASP.NET CheckBoxList in templates

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Thank you for buying
ASP.NET jQuery Cookbook

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

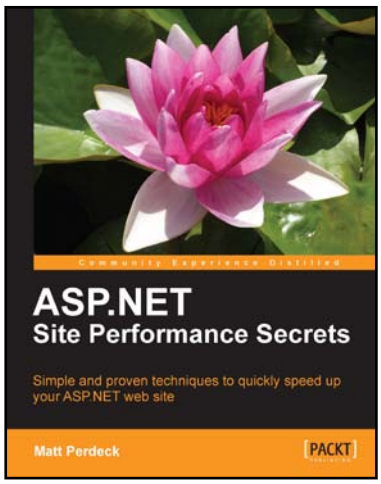
Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] PUBLISHING

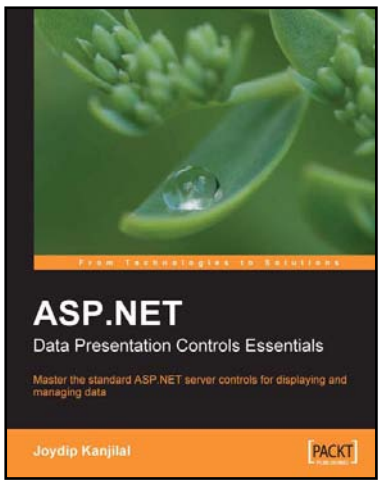


ASP.NET Site Performance Secrets

ISBN: 978-1-849690-68-3 Paperback: 456 pages

Simple and proven techniques to quickly speed up your ASP.NET website

1. Speed up your ASP.NET website by identifying performance bottlenecks that hold back your site's performance and fixing them
2. Tips and tricks for writing faster code and pinpointing those areas in the code that matter most, thus saving time and energy
3. Drastically reduce page load times



ASP.NET Data Presentation Controls Essentials

ISBN: 978-1-847193-95-7 Paperback: 256 pages

Master the standard ASP.NET server controls for displaying and managing data

1. Systematic coverage of major ASP.NET data presentation controls
2. Packed with re-usable examples of common data control tasks
3. Covers LINQ and binding data to ASP.NET 3.5 (Orcas) controls