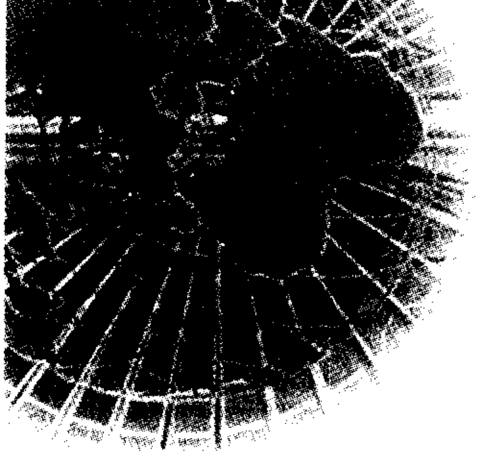


25

Application Globalization

When the Internet or the World Wide Web is making its global presence in almost all walks of life, it becomes quite imperative to move away from the traditional approach of application development to a more advanced one, where the content of a Web page is not restricted to a few international languages, which, in turn, will allow us to reach the masses from all over the world. Application globalization is another concept in application development that has greater implications from the Internet users' point of view, because it facilitates displaying the content of an application according to different cultures from all over the world. Displaying the date and time according to the local culture of a region is an example of application globalization. It provides a culture-specific Web experience to the Internet users.

ASP.NET 3.5 provides built-in mechanisms to globalize Web applications that can adapt to different languages. It is not the underlying .NET Framework that translates the language in which Web applications are developed to a different language. However, it is the built-in mechanism of ASP.NET 3.5 that reconfigures the output of a Web application, so that the content, such as date and time objects, is presented in the appropriate culture-specific format. Basically, application globalization involves two sub-processes, internationalization and localization. Internationalization is the process of internationalizing application code as per some specific standards, and localization is the process of rendering the output of an application according to a specific language and culture.

In this chapter, we take a quick look at the process of application globalization, which includes internationalization and localization. We then explore how to generate local and global resources to create globalization-enabled Web applications. We also learn to perform implicit and explicit localization and localize the script files. The chapter also describes how to perform localization of HyperText Markup Language (HTML) elements and static content.

Globalization

Globalization of Web application is the process of designing and developing Web applications that can display Web page contents in different languages and cultures. The process of globalizing an ASP.NET Web application involves separating the resources of the Web application that are culture-specific and need to be modified for different cultures. The next step is to customize the Web application for specific cultures.

To globalize Web applications in .NET Framework, you need to use the following two namespaces:

- `System.Globalization`—Contains the classes defining information related to culture, such as language, country, format, and pattern for date, currency, and numbers
- `System.Resources`—Contains the classes and interfaces that you can use to create, store, and manage culture-specific resources used in a Web application

Table 25.1 lists the classes contained in the `System.Globalization` namespace:

Table 25.1: Classes in the System.Globalization Namespace

Class	Description
`Calendar`	Represents time in different parts, such as weeks, months, and years.
`CharUnicodeInfo`	Gets information related to the Unicode character.
`ChineseLunisolarCalendar`	Represents time in different parts, such as months, days, and years. Days and months are calculated with a lunisolar calendar, whereas years are calculated using a Chinese calendar.
`CompareInfo`	Provides methods to perform culture-specific string comparisons.
`CultureAndRegionInfoBuilder`	Specifies a customized culture that is new or based on another culture and country/region.
`CultureInfo`	Represents culture-specific information, such as name, region, and language.
`DateTimeFormatInfo`	Represents the formats for the culture-specific date and time.
`DaylightTime`	Represents the daylight-saving time period.

Table 25.1: Classes in the System.Globalization Namespace

Class	Description
`EastAsianLunisolarCalendar`	Represents division of time into months, days, years, and eras. Their dates are based on the cycles of the sun and the moon.
`GregorianCalendar`	Represents the Gregorian calendar.
`HebrewCalendar`	Represents the Hebrew calendar.
`HijriCalendar`	Represents the Hijri calendar.
`JapaneseCalendar`	Represents the Japanese calendar.
`JapaneseLunisolarCalendar`	Represents time in different parts, such as months, days, and years. Days and months are calculated with a lunisolar calendar, whereas years are calculated using a Japanese calendar.
`JulianCalendar`	Represents the Julian calendar.
`KoreanCalendar`	Represents the Korean calendar.
`KoreanLunisolarCalendar`	Represents time in different parts, such as months, days, and years. Days and months are calculated with a lunisolar calendar, whereas years are calculated using a Gregorian calendar.
`NumberFormatInfo`	Defines the manner in which culture-specific numeric information is formatted and displayed.
`PersianCalendar`	Represents the `Persian` calendar.
`RegionInfo`	Represents information about a country or region.
`SortKey`	Represents the mapping results obtained after mapping a string with its sort key.
`StringInfo`	Provides different string manipulation methods using which you can split a string into separate text elements and then iterate through those elements.
`TaiwanCalendar`	Represents the Taiwan calendar.
`TaiwanLunisolarCalendar`	Represents a Taiwan lunisolar calendar in which days and months are calculated with a lunisolar calendar, whereas years are calculated using a Gregorian calendar.
`TextElementEnumerator`	Enumerates the text elements of a string.
`TextInfo`	Defines the properties and behavior of a writing system.
`ThaiBuddhistCalendar`	Represents the Thai Buddhist calendar.
`UmAlQuraCalendar`	Represents the Saudi Hijri (Um Al Qura) calendar.

Table 25.2 lists the classes contained in the `System.Resources` namespace:

Table 25.2: Classes in the System.Resources Namespace

Class	Description
`MissingManifestResourceException`	Represents the exception that is thrown when the main assembly does not contain neutral culture resources, and such resources are required as the satellite assembly is not present
`MissingSatelliteAssemblyException`	Represents the exception that is thrown when the satellite assembly for the neutral culture resources is not present
`NeutralResourcesLanguageAttribute`	Communicates to the `ResourceManager` class about the language used to write the neutral culture resources

Table 25.2: Classes in the System.Resources Namespace

[illegible]	[illegible]
`ResourceManager`	Provides dynamic access to the resources of a specific culture
`ResourceReader`	Allows you to read the resource names and the resource value pairs from the `.resources` files
`ResourceSet`	Stores the set of all localized resources corresponding to a specific culture, while neglecting others
`ResourceWriter`	Writes resources to an output stream or file in system's default format
`ResXDataNode`	Represents an element or data types in a resource file
`ResXFileRef`	Includes references of an external resource (`.resx`) file
`ResXFileRef.Converter`	Converts a `ResXFileRef` reference link to a string and vice versa
`ResXResourceReader`	Allows you to read the resource names and the resource value pairs from the `.resx` files
`ResXResourceSet`	Collects all items to create a single object of the `.resx` file
`ResXResourceWriter`	Writes resources in `.resx` file or an output stream
`SatelliteContractVersionAttribute`	Ensures that the `ResourceManager` class retrieves the required version of a satellite assembly, to simplify the updating process of the main assembly

Globalization of a Web application uses resources for user interface (UI) and static content of a Web application. As a result, different functional experts are required to build a globalized Web application. The different functional roles that are required to build a Web application are as follows:

- **Web designer**—Determines the Graphical User Interface (GUI) of a Web application. The Web designers decide how the content and controls are grouped on each Web page.
- **Developers**—Develop ASP.NET pages for the content and controls designed by Web designers. In addition, developers also perform tasks such as determining the structure for content storage, designing databases, and handling access to the content at runtime.
- **Translators**—Translate the content to be presented on the Web application to a desired language and culture. In addition, translators are also required to work with Resource editors and XML editors to modify database content.
- **Stakeholders**—Coordinates between Web designers, developers, and translators. In addition, the role requires storing the content of a Web application such that the translators can access and modify the content. The stakeholders also ensure that the Web application supports new cultures without modifying the GUI of the Web application or database structure and other components of a Web application.

Globalization of Web application includes the following two processes:

- Internationalization
- Localization

Now, let's discuss these in detail.

Internationalization

The process of Internationalization involves identifying different contents for different locales, such as date and time formats. In general, to serve the purpose, you need to write the source code for the Web application such that the code is dynamically formatted to present the content of the Web application in the desired locale. The process of Internationalization ensures that the source code of a Web application is not modified when the application needs to be customized for specific culture or regions.

The goals of the process of Internationalization are as follows:

- Presenting the content in a single UI that will present the content for any culture.
- Placing the content at a location where it can be easily translated to another language. The location should also be programmatically accessible so that the content can be added to the UI of the Web application.
- Storing the content and user input as per the required culture.

Localization

Localization refers to the process of making changes in a Web application to meet the language and cultural requirements of users from a specific geographical location. The localization process involves configuring a Web application for a given culture. This implies translating and formatting content, such as time and date, according to the culture for which the Web application is to be localized. The elements, such as numeric data, date formats, and currency, are customized for a given culture in Web applications.

Before the introduction of ASP.NET 2.0, it was difficult to localize a Web application so that it supports more than one language. Previously, the resource files and the `ResourceManager` class were used and we had to create satellite assemblies, which required considerable amount of effort and code. However, ASP.NET 2.0 simplifies the localization process by introducing following new features:

- Auto-detection of preferred culture of the client browser
- Accessing resources programmatically
- Compilation of files with the `.resx` or `.resource` extension
- Supporting the creation of resources at design-time
- Declarative resource expressions (implicit and explicit) to bind the controls or the properties of the controls to the resources

While building a multilingual Web application, it is beneficial to maintain the source code for the entire application in a single page, as it is easier to maintain a single page in comparison to maintaining separate sites for different languages. ASP.NET enables localization of a Web application into different languages without any changes to the source code through resource expressions. Resource expressions are a sub-feature of the overall expressions, such as the data-binding expressions feature of earlier versions of ASP.NET.

An amazing feature of ASP.NET 3.5 is that it supports the usage of different resource files in a Web application. These resource files follow a unique naming convention and have the `.resx` file extension. All the resource files related to a Web application are stored in a separate folder, which enables us to add more resource files to a Web application without performing the compilation process. As we go further, we will read about the samples executed for en-US (English-US), fr (French), and de (German) language. You can also use other languages in a Web application after creating resource files for that language and setting the language as a preferred language in your browser.

Now, let's discuss about cultures.

Listing the Available Cultures

ASP.NET Framework provides with the `CultureInfo` class residing in the `System.Globalization` namespace. The `CultureInfo` class contains culture-specific information, such as the language, country, and cultural conventions that are associated with a specific culture. The `CultureInfo` class also provides information required for performing culture-specific operations, such as formatting dates and numbers, casing, and comparing strings. In addition, you also need to use the `System.Resources` and `System.Threading` namespaces to set culture-specific information of a Web application.

You can use the `CultureInfo.GetCultures` method in the `CultureInfo` class to view the complete list of all the culture options according to which you can customize a Web application.

To use the `CultureInfo.GetCultures` method, follow these steps:

1. Create a console application in Visual Studio 2008 and name it `CulturesVB`. You can find the code of `CulturesVB` application in the Code\ASP.NET\Chapter 25\`CulturesVB` folder on the CD. You can find this application in the CD-ROM.
2. After creating the console application, open the class file from the Solution Explorer (`Module1.vb`), and add the code shown in Listing 25.1:

Listing 25.1: Code for the Class File

```
Imports System.Globalization
Module Module1
    Sub Main()
          Dim cultinfo As CultureInfo
                For Each cultinfo In _
                CultureInfo.GetCultures(CultureTypes.AllCultures)
                Console.WriteLine(cultinfo)
                Next cultinfo
                Console.ReadLine()
    End Sub
End Module
```

3. Now, run the `CulturesVB` application. The output of the application, a list of cultures, is shown in Figure 25.1:

Figure 25.1: List of Cultures

A Web page in a Web application consists of two culture values, `Culture` and `UICulture`. The `Culture` value determines the functions, such as date and currency. These `Culture` values are used to format data and numbers in a Web page. The `UICulture` value determines the resources that are loaded on a Web page of a Web application. Resources in a Web application represent data, such as strings or images, displayed in the UI of the Web application.

You can set the `Culture` and `UICulture` values in a Web application by using any one of the following methods:

- Through the `web.config` file
- In the code-inline page
- In the code-behind page

NOTE

To translate the content of a Web application from one language to another, you need to place the string value in a resource file.

The `Culture` value for a Web application is specified by the preceding methods so that the UI of the Web application is appropriate for the culture for which it is created. You can set the `Culture` and `UICulture` values in a Web application to override the user settings or operating system settings.

To set the `Culture` and `UICulture` values through the `web.config` file, write the following code in the Globalization section of the `web.config` file:

```
culture="en-US"
uiCulture="de"
```

In the preceding code, the combination of the language and culture values for the Web application is set to US English (en-US). The `UICulture` value is set to German language (de).

To set the `Culture` and `UICulture` values in the source code of the Web page, you need to include the following values to the Page directive:

```
<%@ Page UICulture="de" Culture="en-US"...%>
```

In the code-behind file, you can set the default culture for a Web application. ASP.NET provides the `CurrentCulture` and `CurrentUICulture` properties to set the default values for the `Culture` and `UICulture` attributes in the code-behind file. For this, you need to include the following code in the code-behind file to set the default culture to German.

The code to be added to the code-behind page:

```
Imports System.Globalization
Imports System.Threading
Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture ("de")
Thread.CurrentThread.CurrentUICulture=new CultureInfo("de")
```

ASP.NET also provides an auto-culture-handling feature that enables you to easily localize a Web application. You need to include the `Culture="auto"` and `UICulture ="auto"` attributes in the Page directive of each Web page of the Web application to enable the auto-culture handling feature.

Working with Resources

Resources, such as images and strings, are used in various elements, such as the toolbar and menu icons of a Web application. If you want to make changes in the strings or image resources used in a Web application, you need to search the entire source code of the Web application for the resource to be modified. To simplify the process of making changes to the resources used in a Web application, you can place the source code for strings and images used in the Web application in a separate file. You can then modify the code in the file to change the used resources. Such a file that contains source code for the various resources used in a Web application is called the resource file.

To create resource files, first you need to identify the sections specific to a culture in a Web application. You then need to make different resource files for the cultures for which you need to develop a Web application. Based on the culture settings made by a user in the browser, the resource files are loaded in the Web application. For example, consider that you need to display the content of a Web application in French language. For this, you need to create resource file for the French language. You can create the resource file in either the `App_LocalResources` folder or the `App_GlobalResources` folder of a Web application. The language settings of the browser that are used to view the output of a Web application are then changed to load the resource file in the Web application.

In ASP.NET Framework, you can create global resources as well as local resources. The global resources are accessible from all the Web pages of an ASP.NET Web application, while the local resources are accessible from a single Web page of the ASP.NET Web application. Now, let's discuss these resources in detail.

Working with Local Resources

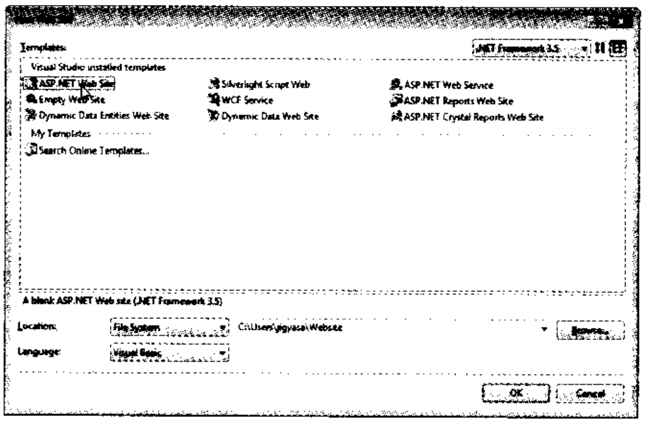
The local resources for a specific Web page in the Web application are created in the `App_LocalResources` folder. Unlike the `App_GlobalResources` folder that resides in the root directory, the `App_LocalResources` folder can exist in any directory in the ASP.NET Web application. You can make a local resource file accessible to a Web page by using the name of the resource file. The naming convention of the resource file should match the base name of the Web page that accesses the resource file. The base name should be followed by the language name and the culture name for which the resource file is created. The name of the resource file should end with the extension `.resx`. For example, to associate a resource file for German language to the `Default.aspx` page, the naming convention for the resource file is given as follows:

```
Default.aspx.de.resx
```

In the preceding name for the resource file, the base name, `Default.aspx`, matches the name of the Web page. Here, de is the International Organization for Standardization (ISO) language code of the German language. The name of the resource file ends with the `.resx` extension.

To add local resources to a website, follow these steps:

1. In Visual Studio 2008, select the `File→New→Web Site` option from the menu bar to create a new website. This opens the New Web Site dialog box, as shown in Figure 25.2:



A blank ASP.NET Web site (.NET Framework 3.5)

Location: File System C:\Users\yigyasa\Website

Language: Visual Basic

Figure 25.2: The New Web Site Dialog Box

2. In the New Web Site dialog box, select the ASP.NET Web Site template, enter the location and name of the website beside the Location field, and click the OK button.

This will create the website and add the default files in the Solution Explorer, as shown in Figure 25.3:

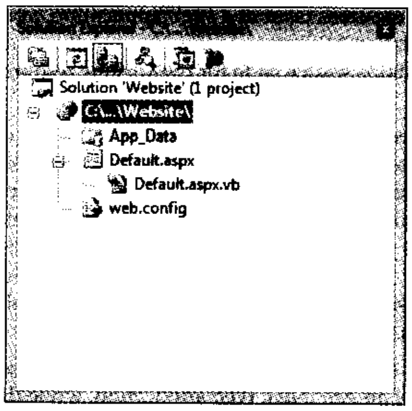


Figure 25.3: Showing the Default Files in the Solution Explorer

3. After creating the website, right-click the name of the website (`Website`) and select the Add ASP.NET `Folder→App_LocalResources` option from the context menu to add the `App_LocalResources` folder to the website.

Working with Global Resources

Unlike local resources that are generated automatically, you need to add the global resources to a Web application. To add global resources, you can create a resource file with the `.resx` extension in the `App_GlobalResources` folder. The `App_GlobalResources` folder resides in the root directory of an ASP.NET Web application and contains the global resources for the Web application. In addition, the

App_GlobalResources folder also provides a simple way to programmatically access global resources in a Web application.

The process of adding the global resources in a website is similar to the process of adding the local resources. The only difference is that you need to select the Add ASP.NET Folder→App_GlobalResources option from the context menu to add the App_GlobalResources folder.

Using Localization Expression

The localization expressions are added for controls, such as labels and buttons, on the Web page of a Web application. The localization expressions enable you to access the local and global resources of the control programmatically to generate the localized content. When the localization expressions are added to a Web application, it results in modification of the control declaration on the Web page. The declarative statement meta:resourcekey, which is added to the control declaration, indicates that the ASP.NET parser should generate code to retrieve property values from either the local or the global resources that are created. The localization expressions generated for resources are either implicit localization expressions or explicit localization expressions.

Implicit Expression

Implicit localization expressions are automatically generated when resources are generated using implicit localization. While using implicit localization, the properties of the controls created on a Web page are read from a resource file. You create a resource file that contains localized values for properties, such as text, of the controls. At runtime, ASP.NET Framework examines the controls created on the Web page to verify if the controls are using implicit localization. The values of the controls that are using implicit localization are then substituted in the resource files.

To perform implicit localization, you need to create controls on a Web page and then generate a resource file for customizing the application for different languages. You can find the code of ImplicitLocalizationVB application in the
Code\ASP.NET\Chapter 25\ImplicitLocalizationVB folder on the CD. To create controls on a Web page, create a new ASP.NET Web application, named ImplicitLocalizationVB, in Visual Studio 2008, and follow these steps:

1. Double-click the Default.aspx page in the Solution Explorer to open the page.
2. Switch to the Design view of the Default.aspx page and drop three Label controls (Label1, Label2 and Label3) and a Calendar control (Calendar1) from the Toolbox onto the Design view of the file to create a UI of the Web application. You can also write the code, as shown in Listing 25.2, in the Defualt.aspx page to add controls:

Listing 25.2: Code for the Default.aspx Page

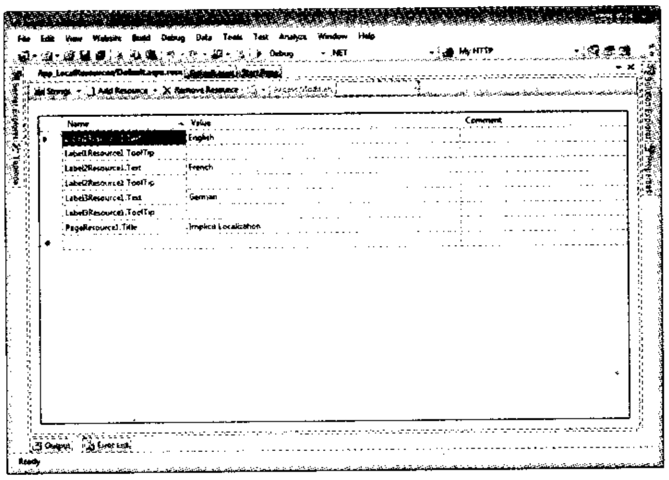
```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default"
  Culture="auto" meta:resourcekey="PageResource1" UICulture="auto" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Implicit Localization</title>
  <link href="StyleSheet.css" rel="Stylesheet" type="text/css" />
</head>
<body>
  <form id="form1" runat="server">
        <div>
              <div id="header">

              </div>
        <div id="sidebar">
              <div id="nav">
                    &nbsp; &nbsp;&nbsp;<br />
```

```
                        <strong><span style="text-decoration: underline"><em>
                        <br />
                        </em></span></strong>
                        <br />
                        <br />
                        <asp:Calendar ID="Calendar1" runat="server" BackColor="White"
                          BorderColor="White"
                        BorderWidth="1px" Font-Names="Verdana" Font-Size="9pt"
                          ForeColor="Black" Height="190px"
                        NextPrevFormat="FullMonth" Width="254px">
                        <SelectedDayStyle BackColor="#333399" ForeColor="White" />
                        <TodayDayStyle BackColor="#CCCCCC" />
                        <OtherMonthDayStyle ForeColor="#999999" />
                        <NextPrevStyle Font-Bold="True" Font-Size="8pt"
                          ForeColor="#333333" VerticalAlign="Bottom" />
                        <DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
                        <TitleStyle BackColor="White" BorderColor="Black"
                          BorderWidth="4px" Font-Bold="True"
                        Font-Size="12pt" ForeColor="#333399" />
                        </asp:Calendar>
                </div>
        </div>
        <div id="content">
                <div class="itemContent">
                <br />
                <br />
                &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                &nbsp; &nbsp; &nbsp; &nbsp;&nbsp;
                <table style="width: 383px; height: 193px; background-color: #99ffff">
                <tr>
                        <td colspan="2" style="width: 100px; background-color: White">
                        &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                        &nbsp; &nbsp; &nbsp;
                        &nbsp; &nbsp;<span style="text-decoration:
                          underline"><strong>I<span>mplicitLocalization</span>
                        </strong></span>
                        </td>
                </tr>
                <tr>
                        <td style="width: 75px; height: 18px; background-color: White">
                        &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; <strong>English</strong>
                        </td>
                        <td style="width: 100px; height: 18px; background-color: White">
                        &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                        <asp:Label ID="Label1" runat="server" Font-Bold="True"
                          meta:resourcekey="Label1Resource1"
                        Text="English"></asp:Label>
                        </td>
                </tr>
                <tr>
                        <td style="width: 75px; background-color: White">
                        <span style="background-color:
                          White"><strong>French</strong>&nbsp; </span>
                        </td>
                        <td style="width: 100px; background-color: White">
                        <strong>&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                          &nbsp;&nbsp; &nbsp; &nbsp;</strong><asp:Label
                        ID="Label2" runat="server" Font-Bold="True"
                          meta:resourcekey="Label2Resource1"
                        Text="French"></asp:Label>
                        </td>
                </tr>
                <tr>
                        <td style="width: 75px; background-color: White">
                        &nbsp; &nbsp; &nbsp; &nbsp;&nbsp; <strong>German</strong>
                        </td>
                        <td style="width: 100px; background-color: white">
                        &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                        <asp:Label ID="Label3" runat="server" Font-Bold="True"
                          meta:resourcekey="Label3Resource1"
                        Text="German"></asp:Label>
                        </td>
                    </tr>
                    </table>
                    </div>
                    <div id="footer">
                        <p class="left">
                        All content copyright &copy; Kogent Solutions Inc.</p>
                    </div>
                    </div>
            </div>
        </form>
    </body>
    </html>
```

3. After adding the controls on the `Default.aspx` page, generate a local resource file for the `Default.aspx` page. To do so, click on the Design view of the `Default.aspx` page and click the `Tools`→`Generate Local Resource` option from the menu bar.

A new folder `App_LocalResources` is created in the Solution Explorer. Within the `App_LocalResources` folder, a file named `Default.aspx.resx` is created.

4. After creating the resource file, you need to use the Resource editor to place the localized text within the resource file. Therefore, double-click the `Default.aspx.resx` resource file to open the Resource editor for the file. The Resource editor consists of the Name, Value, and comment columns.
5. Change the property in the Value column as English, French, and German for the `Label1Resource1.Text`, `Label2Resource1.Text`, and `Label3Resource1.Text` fields, respectively, as shown in Figure 25.4:



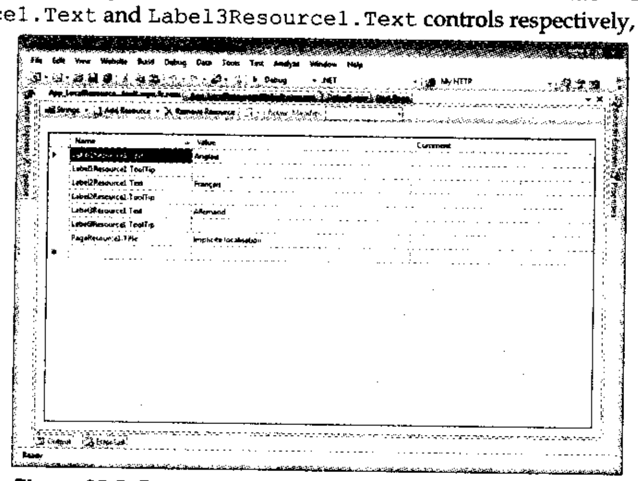
Name	Value	Comment
Label1Resource1.Text	English	
Label1Resource1.ToolTip		
Label2Resource1.Text	French	
Label2Resource1.ToolTip		
Label3Resource1.Text	German	
Label3Resource1.ToolTip		
PageResource1.Title	Implicit Localization	

Figure 25.4: Resource Editor of the Default.aspx.resx File

6. Now, press `F5` to run the Web application. The names of the labels in the Web application are displayed as edited in the `Default.aspx.resx` file.

You need to create a unique resource file for each combination of culture and language that you need to include in a Web application.

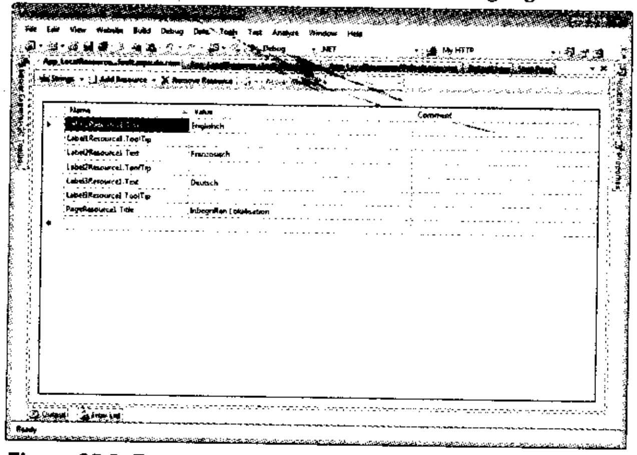
7. To create a French language resource file for the Web application, right-click the `Default.aspx.resx` resource file in the Solution Explorer and select the Copy option from the context menu to copy the resource file.
8. Right-click the `App_LocalResources` folder and select the Paste option from the context menu to paste the file in the `App_LocalResources` folder. A resource file named `copy of Default.aspx.resx` file is created in the `App_LocalResources` folder.
9. Now, right-click the `copy of Default.aspx.resx` resource file and select the Rename option to rename the resource file as `Default.aspx.fr.resx`.
10. Double-click the `Default.aspx.fr.resx` resource file to open it, and then set the text property in the Value column as Anglais, Français, and Allemand for the `Label1Resource1.Text`, `Label2Resource1.Text` and `Label3Resource1.Text` controls respectively, as shown in Figure 25.5:



Name	Value	Comment
[illegible]	Anglais	
Label1Resource1.ToolTip		
Label2Resource1.Text	Français	
Label2Resource1.ToolTip		
Label3Resource1.Text	Allemand	
Label3Resource1.ToolTip		
PageResource1.Title	Implicite localisation	

Figure 25.5: Resource Editor for the Default.aspx.fr.resx File

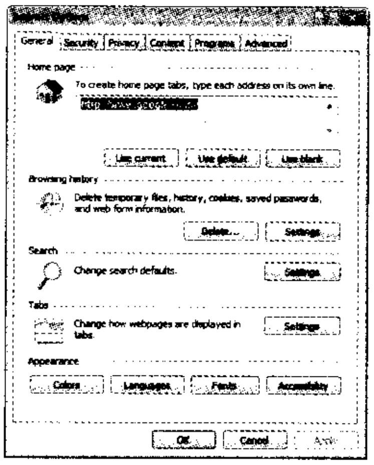
11. Similarly, create and add a German language resource file (`Default.aspx.de.resx`) to the Web application. Figure 25.6 shows the Resource editor for German language:



Name	Value	Comment
[illegible]	Englisch	
Label1Resource1.ToolTip		
Label2Resource1.Text	Französisch	
Label2Resource1.ToolTip		
Label3Resource1.Text	Deutsch	
Label3Resource1.ToolTip		
PageResource1.Title	Inbegriffen Lokalisation	

Figure 25.6: Resource Editor for the Default.aspx.de.resx File

12. Now, alter the browser settings so that ASP.NET uses the resource file and includes the language and culture specified in the browser settings in the Web application. To change the language settings of the Microsoft Internet Explorer (IE) browser, select `Tools→Internet Options` on the menu bar of IE to open the Internet Options dialog box, as shown in Figure 25.7:



To create home page tabs, type each address on its own line.

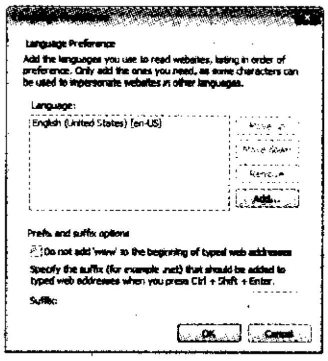
Delete temporary files, history, cookies, saved passwords, and web form information.

Change search defaults.

Change how webpages are displayed in tabs.

Figure 25.7: The Internet Options Dialog Box

13. Click the Languages button on the General tab page to open the Language Preference dialog box, as shown in Figure 25.8:



Language Preference

Add the languages you use to read websites, listing in order of preference. Only add the ones you need, as some characters can be used to impersonate websites in other languages.

Language:

English (United States) [en-US]

Prefix and suffix options

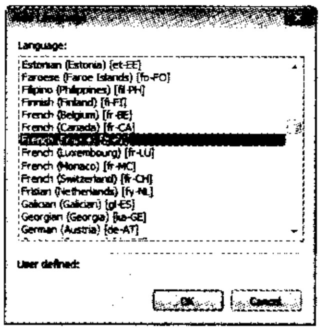
Do not add 'www' to the beginning of typed web addresses

Specify the suffix (for example .net) that should be added to typed web addresses when you press Ctrl + Shift + Enter.

Suffix:

Figure 25.8: The Language Preference Dialog Box

14. Click the Add button on the dialog box to open the Add Language dialog box. Select the `French (France)[ fr-FR]` option in the Language list box, as shown in Figure 25.9:



Language:

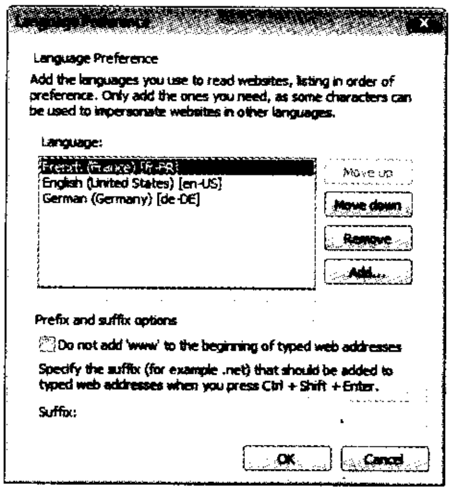
Estonian (Estonia) [et-EE]
Faroese (Faroe Islands) [fo-FO]
Filipino (Philippines) [fil-PH]
Finnish (Finland) [fi-FI]
French (Belgium) [fr-BE]
French (Canada) [fr-CA]
French (France) [fr-FR]
French (Luxembourg) [fr-LU]
French (Monaco) [fr-MC]
French (Switzerland) [fr-CH]
Frisian (Netherlands) [fy-NL]
Galician (Galician) [gl-ES]
Georgian (Georgia) [ka-GE]
German (Austria) [de-AT]

User defined:

Figure 25.9: The Add Language Dialog Box

15. Click the OK button to close the Add Language dialog box. Similarly, you can select `German (Germany)[ de-DE]` language from the Add Language dialog box to add the German language. Select the `French [ France][ fr-FR]` option from the Language box of the Language Preference dialog box.

16. After selecting the language, click the Move up button to move the selected language up in the order. The `French [ France] [ fr-FR]` option is moved up in the Language box, as shown in Figure 25.10:



Language Preference

Add the languages you use to read websites, listing in order of preference. Only add the ones you need, as some characters can be used to impersonate websites in other languages.

Language:

French (France) [fr-FR]
English (United States) [en-US]
German (Germany) [de-DE]

Prefix and suffix options

☐ Do not add 'www' to the beginning of typed web addresses

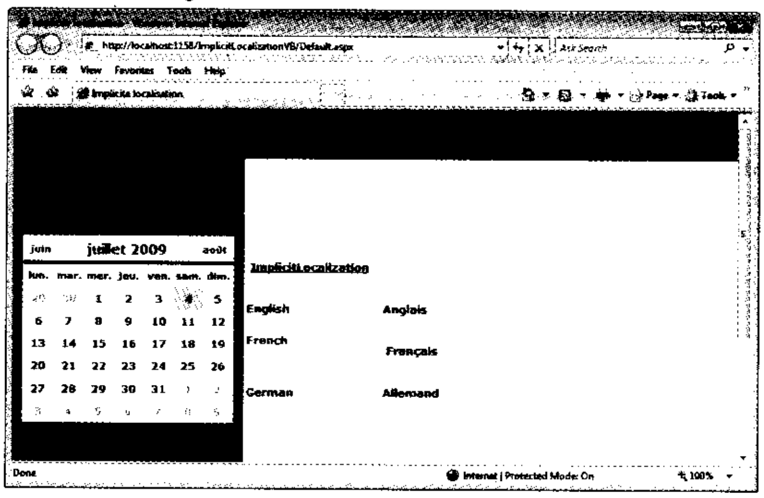
Specify the suffix (for example .net) that should be added to typed web addresses when you press Ctrl + Shift + Enter.

Suffix:

Figure 25.10: The Language Preference Dialog Box

17. Click the OK button to close the Language Preference dialog box, and then close and apply the changes made in the Internet Options dialog box.
18. After making the changes in the settings of the browser, run the Web application by pressing `F5` to display the culture settings in the application by using implicit localization.

The output of the preceding application, when the Language settings in the browser indicate the preferred language as French, is shown in Figure 25.11:



http://localhost:1158/ImplicitLocalizationVB/Default.aspx

juin			juillet 2009			août
lun.	mar.	mer.	jeu.	ven.	sam.	dim.
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

ImplicitLocalization

English Anglais
French Français
German Allemand

Figure 25.11: Output of the Web Application with French as the Preferred Language

The output of the preceding application, when the Language settings in the browser indicate the preferred language as German, is shown in Figure 25.12:

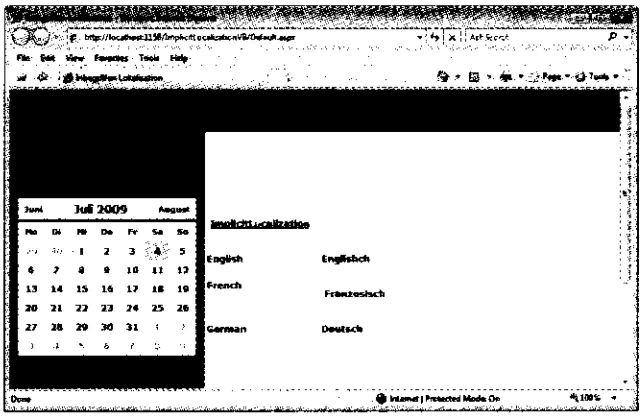


Figure 25.12: Output of the Web Application with German as the Preferred Language

The output of the preceding application, when the Language settings in the browser indicate the preferred language as English, is shown in Figure 25.13:

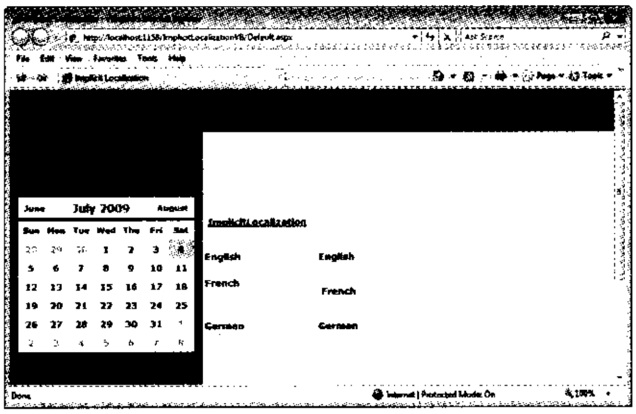


Figure 25.13: Output of the Web Application with English as the Preferred Language

You can view the code of the resource files XML Editor, by right-clicking the name of the file in the Solution Explorer, selecting `Open With` option. The complete code of the resource file (`Default.aspx.resx`) is shown in Listing 25.3:

Listing 25.3: Complete Code for the `Default.aspx.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
        Microsoft ResX Schema
        Version 2.0
        The primary goals of this format is to allow a simple XML format that is mostly
        human readable. The generation and parsing of the various data types are done
        through the TypeConverter classes associated with the data types.
    Example:
        ... ado.net/XML headers & schema ...
        <resheader name="resmimetype">text/microsoft-resx</resheader>
        <resheader name="version">2.0</resheader>
        <resheader name="reader">System.Resources.ResXResourceReader,
          System.Windows.Forms, ...</resheader>
        <resheader name="writer">System.Resources.ResXResourceWriter,
          System.Windows.Forms, ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
      comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
    <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing"
      mimetype="application/x-microsoft.net.object.bytearray.base64">
    <value>[base64 mime encoded string representing a byte array form of the .NET
      Framework object]</value>
    <comment>This is a comment</comment>
    </data>
    There are any number of "resheader" rows that contain simple
    name/value pairs.
    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.
    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:
    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.
    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
-->
<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
    <xsd:element name="root" msdata:IsDataSet="true">
            <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
            <xsd:element name="metadata">
            <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="name" use="required" type="xsd:string" />
            <xsd:attribute name="type" type="xsd:string" />
            <xsd:attribute name="mimetype" type="xsd:string" />
            <xsd:attribute ref="xml:space" />
            </xsd:complexType>
            </xsd:element>
            <xsd:element name="assembly">
            <xsd:complexType>
```

```
        <xsd:attribute name="alias" type="xsd:string" />
        <xsd:attribute name="name" type="xsd:string" />
        </xsd:complexType>
        </xsd:element>
        <xsd:element name="data">
        <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="value" type="xsd:string" minOccurs="0"
          msdata:Ordinal="1" />
        <xsd:element name="comment" type="xsd:string" minOccurs="0"
          msdata:Ordinal="2" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required"
          msdata:Ordinal="1" />
        <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
        <xsd:attribute name="mimetype" type="xsd:string"
          msdata:Ordinal="4" />
        <xsd:attribute ref="xml:space" />
        </xsd:complexType>
        </xsd:element>
        <xsd:element name="resheader">
            <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0"
              msdata:Ordinal="1" />
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
        </xsd:element>
        </xsd:choice>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
<resheader name="resmimetype">
    <value>text/microsoft-resx</value>
</resheader>
<resheader name="version">
    <value>2.0</value>
</resheader>
<resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
      Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
<resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
      Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
<data name="Label1Resource1.Text" xml:space="preserve">
    <value>Englishch</value>
</data>
<data name="Label1Resource1.ToolTip" xml:space="preserve">
    <value />
</data>
<data name="Label2Resource1.Text" xml:space="preserve">
    <value>Franzosisch</value>
</data>
<data name="Label2Resource1.ToolTip" xml:space="preserve">
    <value />
</data>
<data name="Label3Resource1.Text" xml:space="preserve">
```

```
        <value>Deutsch</value>
    </data>
    <data name="Label3Resource1.ToolTip" xml:space="preserve">
        <value />
    </data>
    <data name="PageResource1.Title" xml:space="preserve">
        <value>Inbegriffen Lokalisation</value>
    </data>
</root>
```

The complete code of another resource file (`Default.aspx.fr.resx`) is shown in Listing 25.4:

Listing 25.4: Complete Code for the `Default.aspx.fr.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
  Microsoft ResX Schema

  Version 2.0

  The primary goals of this format is to allow a simple XML format
  that is mostly human readable. The generation and parsing of the
  various data types are done through the TypeConverter classes
  associated with the data types.

  Example:

  ... ado.net/XML headers & schema ...
  <resheader name="resmimetype">text/microsoft-resx</resheader>
  <resheader name="version">2.0</resheader>
  <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
    ...</resheader>
  <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
    ...</resheader>
  <data name="Name1"><value>this is my long string</value><comment>this is a
    comment</comment></data>
  <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
  <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
      <value>[base64 mime encoded serialized .NET Framework object]</value>
  </data>
  <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
    microsoft.net.object.bytearray.base64">
      <value>[base64 mime encoded string representing a byte array form of the .NET
        Framework object]</value>
      <comment>This is a comment</comment>
  </data>

  There are any number of "resheader" rows that contain simple
  name/value pairs.

  Each data row contains a name, and value. The row also contains a
  type or mimetype. Type corresponds to a .NET class that support
  text/value conversion through the TypeConverter architecture.
  Classes that don't support this are serialized and stored with the
  mimetype set.

  The mimetype is used for serialized objects, and tells the
  ResXResourceReader how to depersist the object. This is currently not
  extensible. For a given mimetype the value must be set accordingly:

  Note - application/x-microsoft.net.object.binary.base64 is the format
```

```
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
    -->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
        <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
        <xsd:element name="root" msdata:IsDataSet="true">
            <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
            <xsd:element name="metadata">
            <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="name" use="required" type="xsd:string" />
            <xsd:attribute name="type" type="xsd:string" />
            <xsd:attribute name="mimetype" type="xsd:string" />
            <xsd:attribute ref="xml:space" />
            </xsd:complexType>
            </xsd:element>
            <xsd:element name="assembly">
            <xsd:complexType>
            <xsd:attribute name="alias" type="xsd:string" />
            <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
            </xsd:element>
            <xsd:element name="data">
            <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0"
              msdata:Ordinal="1" />
            <xsd:element name="comment" type="xsd:string" minOccurs="0"
              msdata:Ordinal="2" />
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"
              msdata:Ordinal="1" />
            <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
            <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
            <xsd:attribute ref="xml:space" />
            </xsd:complexType>
            </xsd:element>
            <xsd:element name="resheader">
            <xsd:complexType>
            <xsd:sequence>
```

```
                    <xsd:element name="value" type="xsd:string" minOccurs="0"
                      msdata:Ordinal="1" />
                    </xsd:sequence>
                    <xsd:attribute name="name" type="xsd:string" use="required" />
                    </xsd:complexType>
                    </xsd:element>
                    </xsd:choice>
                    </xsd:complexType>
            </xsd:element>
    </xsd:schema>
    <resheader name="resmimetype">
            <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="version">
            <value>2.0</value>
    </resheader>
    <resheader name="reader">
            <value>System.Resources.ResXResourceReader, System.Windows.Forms,
              Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="writer">
            <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
              Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <data name="Label1Resource1.Text" xml:space="preserve">
            <value>Anglais</value>
    </data>
    <data name="Label1Resource1.ToolTip" xml:space="preserve">
            <value />
    </data>
    <data name="Label2Resource1.Text" xml:space="preserve">
            <value>Français</value>
    </data>
    <data name="Label2Resource1.ToolTip" xml:space="preserve">
            <value />
    </data>
    <data name="Label3Resource1.Text" xml:space="preserve">
            <value>Allemand</value>
    </data>
    <data name="Label3Resource1.ToolTip" xml:space="preserve">
            <value />
    </data>
    <data name="PageResource1.Title" xml:space="preserve">
            <value>Implicite localisation</value>
    </data>
</root>
```

The complete code of another resource file (`Default.aspx.de.resx`) is shown in Listing 25.5:

Listing 25.5: Complete Code for the `Default.aspx.de.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
  Microsoft ResX Schema
  Version 2.0
  The primary goals of this format is to allow a simple XML format
  that is mostly human readable. The generation and parsing of the
  various data types are done through the TypeConverter classes
  associated with the data types.
  Example:
    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
      ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
      ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
      comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
        <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
      microsoft.net.object.bytearray.base64">
        <value>[base64 mime encoded string representing a byte array form of the .NET
          Framework object]</value>
        <comment>This is a comment</comment>
    </data>
    There are any number of "resheader" rows that contain simple
    name/value pairs.
    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.
    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:
    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.
    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.
    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
  -->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
    <xsd:element name="root" msdata:IsDataSet="true">
        <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
                <xsd:element name="metadata">
                <xsd:complexType>
                        <xsd:sequence>
                        <xsd:element name="value" type="xsd:string" minOccurs="0" />
                        </xsd:sequence>
                        <xsd:attribute name="name" use="required" type="xsd:string" />
                        <xsd:attribute name="type" type="xsd:string" />
```

```
                    <xsd:attribute name="mimetype" type="xsd:string" />
                    <xsd:attribute ref="xml:space" />
                </xsd:complexType>
                </xsd:element>
                <xsd:element name="assembly">
                    <xsd:complexType>
                        <xsd:attribute name="alias" type="xsd:string" />
                        <xsd:attribute name="name" type="xsd:string" />
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="data">
                    <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="value" type="xsd:string" minOccurs="0"
                          msdata:Ordinal="1" />
                        <xsd:element name="comment" type="xsd:string"
                          minOccurs="0" msdata:Ordinal="2" />
                    </xsd:sequence>
                    <xsd:attribute name="name" type="xsd:string" use="required"
                      msdata:Ordinal="1" />
                    <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                    <xsd:attribute name="mimetype" type="xsd:string"
                      msdata:Ordinal="4" />
                    <xsd:attribute ref="xml:space" />
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="resheader">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="value" type="xsd:string"
                              minOccurs="0" msdata:Ordinal="1" />
                        </xsd:sequence>
                        <xsd:attribute name="name" type="xsd:string"
                          use="required" />
                    </xsd:complexType>
                </xsd:element>
            </xsd:choice>
            </xsd:complexType>
    </xsd:element>
    </xsd:schema>
    <resheader name="resmimetype">
        <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="version">
        <value>2.0</value>
    </resheader>
    <resheader name="reader">
        <value>System.Resources.ResXResourceReader, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="writer">
        <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <data name="Label1Resource1.Text" xml:space="preserve">
        <value>Englishch</value>
    </data>
    <data name="Label1Resource1.ToolTip" xml:space="preserve">
        <value />
    </data>
    <data name="Label2Resource1.Text" xml:space="preserve">
        <value>Französisch</value>
    </data>
    <data name="Label2Resource1.ToolTip" xml:space="preserve">
        <value />
    </data>
    <data name="Label3Resource1.Text" xml:space="preserve">
        <value>Deutsch</value>
    </data>
    <data name="Label3Resource1.ToolTip" xml:space="preserve">
        <value />
    </data>
    <data name="PageResource1.Title" xml:space="preserve">
        <value>Inbegriffen Lokalisation</value>
    </data>
</root>
```

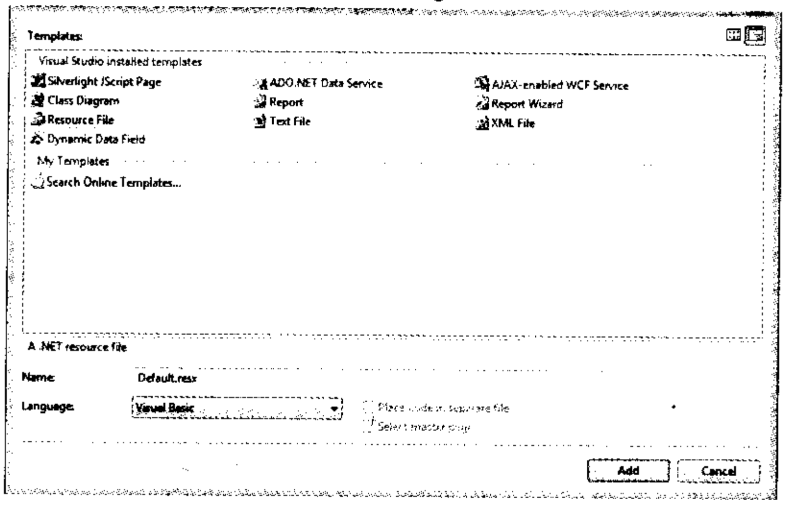
Listings 25.3, 25.4, and 25.5 contain the complete code of all the resource files of the `ImplicitLocalizationVB` application.

Explicit Expression

Explicit localization expressions are automatically generated when resources are created using explicit localization. You need to use explicit localization in the ASP.NET Web pages to localize large text messages, in addition to the controls and labels that are localized.

Let's create a Web application for explicit localization, named `ExplicitLocalizationVB`. `You can find the code of ExplicitLocalizationVB application in the Code\ASP.NET\Chapter 25\ExplicitLocalizationVB folder on the CD`. To do so, follow these steps:

1. Right-click the root directory of the Web application in the Solution Explorer and select the `Add ASP.NET Folder→App_GlobalResources` option from the context menu to add the `App_GlobalResources` folder in the Solution Explorer.
2. Right-click the `App_GlobalResources` folder and select the Add New Item option from the context menu to open the Add New Item dialog box, as shown in Figure 25.14:



A .NET resource file

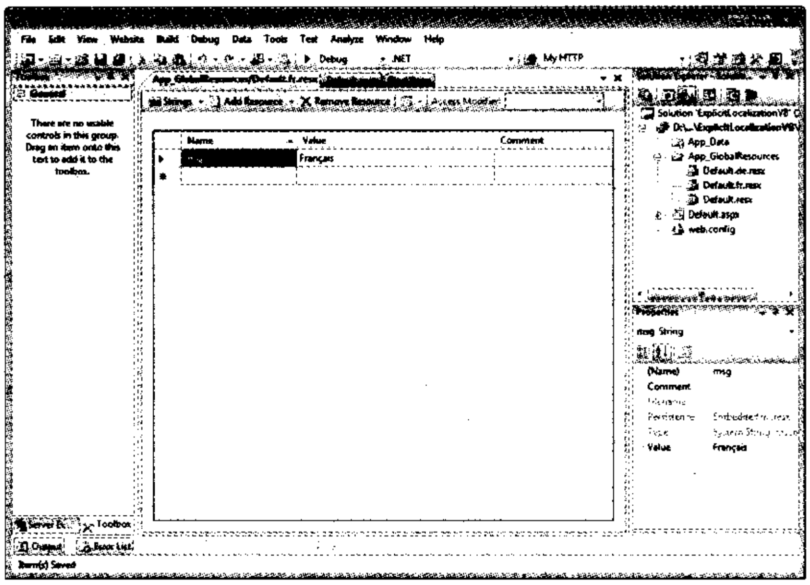
Name: Default.resx

Language: Visual Basic

Figure 25.14: The Add New Item Dialog Box

3. Select Resource File in the Templates pane and type `Default.resx` in the Name field. Then, click the Add button to add the resource file in the `App_GlobalResources` folder.

4. Similarly, create two other resource files in the `App_GlobalResources` folder to include the resource files for the French and German languages. Name the resource files as `Default.fr.resx` and `Default.de.resx`.
5. Double-click the `Default.fr.resx` file in the Solution Explorer to open it. Type `msg` in the Name column and `Français` in the Value column, as shown in Figure 25.15:



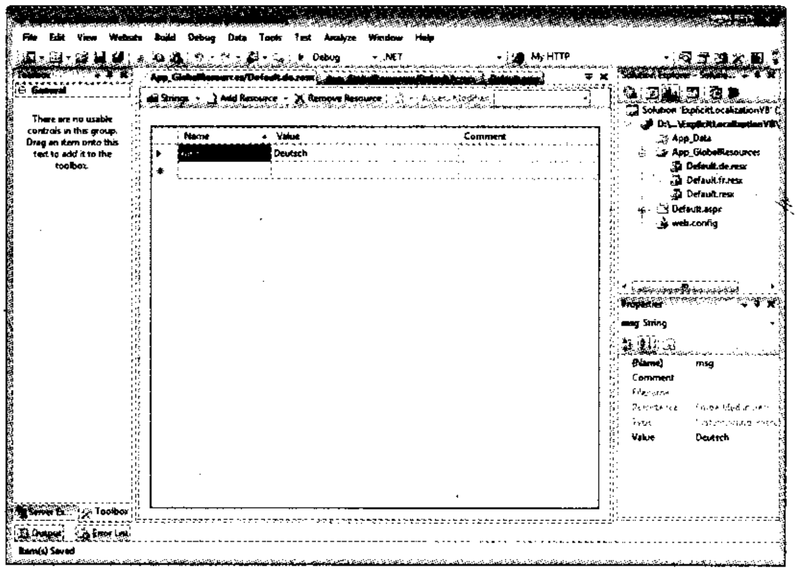
Name	Value	Comment
	Français	

msg String

(Name)	msg
Comment	
Value	Français

Figure 25.15: Resource Editor for the Default.fr.resx File

6. Double-click the `Default.de.resx` file in the Solution Explorer to open the file. Type `msg` in the first row of the Name column and `Deutsch` in the first row of the Value column, as shown in Figure 25.16:



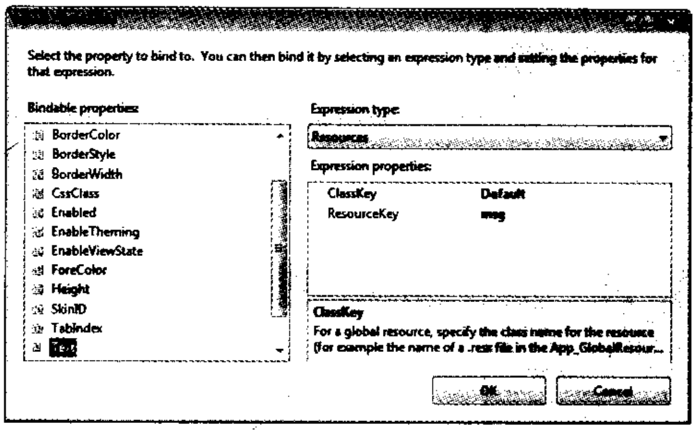
Name	Value	Comment
	Deutsch	

msg String

(Name)	msg
Comment	
Value	Deutsch

Figure 25.16: Resource Editor for the Default.de.resx File

7. Similarly, double-click the `Default.resx` file in the Solution Explorer to open the file. Type `msg` in the first row of the Name column and `English` in the first row of the Value column.
8. Now, design UI for the Web application. To do so, add a `Label` control on the `Default.aspx` page to display the text of the language, whose resource file is loaded in the application.
9. After adding the Label control, open the properties of the Label control and click the ellipsis button in the Expressions property in the Properties window, which displays the Label1 Expressions dialog box (the name of this dialog box may vary on the basis of the name control). The Text option is selected by default in the Bindable Properties section of the Label1 Expressions dialog box.
10. Select Resources from the drop-down list in the Expression type section. Set `ClassKey` to Default and `ResourceKey` to `msg` in the `ClassKey` and `ResourceKey` Expression properties, respectively, as shown in Figure 25.17:


Select the property to bind to. You can then bind it by selecting an expression type and setting the properties for that expression.

Bindable properties:
BorderColor
BorderStyle
BorderWidth
CssClass
Enabled
EnableTheming
EnableViewState
ForeColor
Height
SkinID
TabIndex
Text

Expression type:
Resources

Expression properties:

ClassKey	Default
ResourceKey	msg

ClassKey
For a global resource, specify the class name for the resource (for example the name of a .resx file in the App_GlobalResour...

Figure 25.17: Values specified in the Label1 Expressions Dialog Box

11. Click the OK button to close the Label1 Expressions dialog box.

After creating the UI for a Web application using explicit localization, we need to change the browser settings. The browser settings are changed so that the desired language is displayed when the Web application is executed. The output of the Web application, when the browser settings are changed to English, is shown in Figure 25.18:

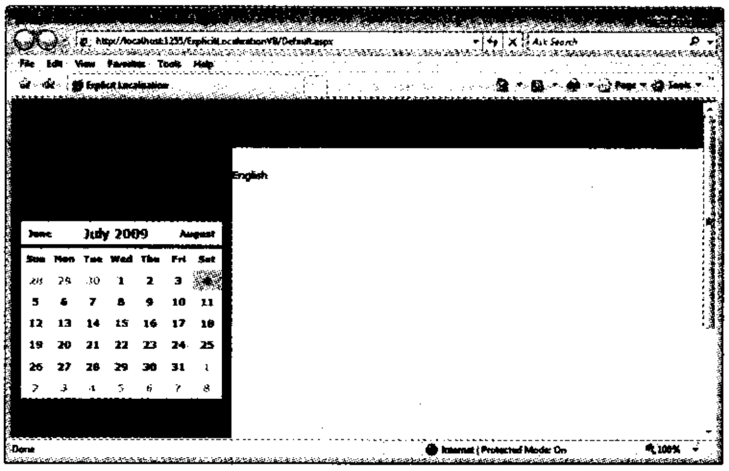


Figure 25.18: Output of the Web Application with English as the Preferred Language

The output of the Web application, when the browser settings are changed to French, is shown in Figure 25.19:

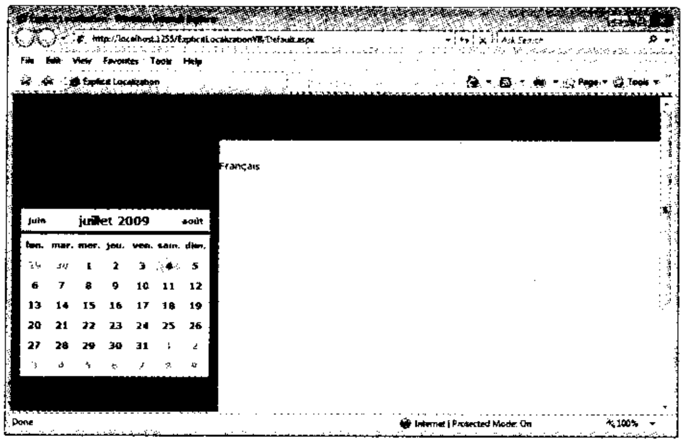

Explicit Localization
Français
juin
juillet 2009
août
lun. mar. mer. jeu. ven. sam. dim.
Done
Internet | Protected Mode: On
100%

Figure 25.19: Output of the Web Application with French as the Preferred Language

The output of the Web application, when the browser settings are changed to German, is shown in Figure 25.20:

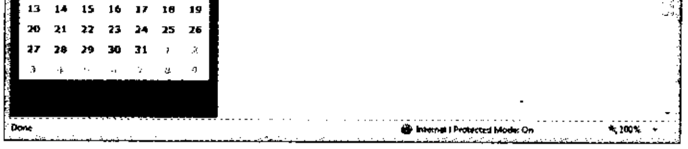


Figure 25.20: Output of the Web Application with German as the Preferred Language

The complete code for the `Default.aspx` page of the `ExplicitLocalizationVB` application is shown in Listing 25.6:

Listing 25.6: Complete Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Culture="auto"
  UICulture="auto" Inherits="_Default" %>
<!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>Explicit Localization</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form1" runat="server">
        <div>
        <div id="header">
                                        </div>
        <div id="sidebar">
```

```
            <div id="nav">
                    &nbsp; &nbsp;&nbsp;<br />
                    <strong><span style="text-decoration: underline"><em>
                    <br />
                    </em></span></strong>
                    <br />
                    <br />
                    <asp:Calendar ID="Calendar1" runat="server" BackColor="White"
                      BorderColor="White"
                    BorderWidth="1px" Font-Names="Verdana" Font-Size="9pt" ForeColor="Black"
                      Height="190px"
                    NextPrevFormat="FullMonth" Width="254px">
                    <SelectedDayStyle BackColor="#333399" ForeColor="White" />
                    <TodayDayStyle BackColor="#CCCCCC" />
                    <OtherMonthDayStyle ForeColor="#999999" />
                    <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333"
                      VerticalAlign="Bottom" />
                    <DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
                    <TitleStyle BackColor="White" BorderColor="Black" BorderWidth="4px" Font-
                    Bold="True"
                    Font-Size="12pt" ForeColor="#333399" />
                    </asp:Calendar>
            </div>
            </div>
            <div id="content">
                    <div class="itemContent">
                    <br />
                    <asp:Label ID="Label1" runat="server" Text="<%$ Resources:Default, msg
                      %>"></asp:Label>&nbsp;
                    </div>
            </div>
            <div id="footer">
                    <p class="left">
                    All content copyright &copy; Kogent Solutions Inc.</p>
            </div>
            </div>
    </form>
  </body>
  </html>
```

The complete code of the resource file (`Default.de.resx`) is shown in Listing 25.7:

Listing 25.7: Complete Code for the `Default.de.resx` File

```
  <?xml version="1.0" encoding="utf-8"?>
  <root>
    <!--
    Microsoft ResX Schema

    Version 2.0

    The primary goals of this format is to allow a simple XML format
    that is mostly human readable. The generation and parsing of the
    various data types are done through the TypeConverter classes
    associated with the data types.

    Example:

    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
```

```
<resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
  ...</resheader>
<resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
  ...</resheader>
<data name="Name1"><value>this is my long string</value><comment>this is a
  comment</comment></data>
<data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
<data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
      <value>[base64 mime encoded serialized .NET Framework object]</value>
</data>
<data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
  microsoft.net.object.bytearray.base64">
      <value>[base64 mime encoded string representing a byte array form of the .NET
        Framework object]</value>
      <comment>This is a comment</comment>
</data>

There are any number of "resheader" rows that contain simple
name/value pairs.

Each data row contains a name, and value. The row also contains a
type or mimetype. Type corresponds to a .NET class that support
text/value conversion through the TypeConverter architecture.
Classes that don't support this are serialized and stored with the
mimetype set.

The mimetype is used for serialized objects, and tells the
ResXResourceReader how to depersist the object. This is currently not
extensible. For a given mimetype the value must be set accordingly:

Note - application/x-microsoft.net.object.binary.base64 is the format
that the ResXResourceWriter will generate, however the reader can
read any of the formats listed below.

mimetype: application/x-microsoft.net.object.binary.base64
value   : The object must be serialized with
        : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
        : and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.soap.base64
value   : The object must be serialized with
        : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
        : and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.bytearray.base64
value   : The object must be serialized into a byte array
        : using a System.ComponentModel.TypeConverter
        : and then encoded with base64 encoding.
-->
<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
<xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
      <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
            <xsd:element name="metadata">
                  <xsd:complexType>
                  <xsd:sequence>
                  <xsd:element name="value" type="xsd:string" minOccurs="0" />
                  </xsd:sequence>
```

```
                        <xsd:attribute name="name" use="required" type="xsd:string" />
                        <xsd:attribute name="type" type="xsd:string" />
                        <xsd:attribute name="mimetype" type="xsd:string" />
                        <xsd:attribute ref="xml:space" />
                        </xsd:complexType>
                <xsd:element>
                <xsd:element name="assembly">
                        <xsd:complexType>
                                <xsd:attribute name="alias" type="xsd:string" />
                                <xsd:attribute name="name" type="xsd:string" />
                        </xsd:complexType>
                </xsd:element>
                <xsd:element name="data">
                <xsd:complexType>
                        <xsd:sequence>
                        <xsd:element name="value" type="xsd:string" minOccurs="0"
                          msdata:Ordinal="1" />
                        <xsd:element name="comment" type="xsd:string" minOccurs="0"
                          msdata:Ordinal="2" />
                        </xsd:sequence>
                        <xsd:attribute name="name" type="xsd:string" use="required"
                          msdata:Ordinal="1" />
                        <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                        <xsd:attribute name="mimetype" type="xsd:string"
                          msdata:Ordinal="4" />
                        <xsd:attribute ref="xml:space" />
                </xsd:complexType>
                </xsd:element>
                <xsd:element name="resheader">
                <xsd:complexType>
                        <xsd:sequence>
                        <xsd:element name="value" type="xsd:string" minOccurs="0"
                          msdata:Ordinal="1" />
                        </xsd:sequence>
                        <xsd:attribute name="name" type="xsd:string" use="required" />
                </xsd:complexType>
                </xsd:element>
                </xsd:choice>
        </xsd:complexType>
        </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
        <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
        <value>2.0</value>
  </resheader>
  <resheader name="reader">
        <value>System.Resources.ResXResourceReader, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
        <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <data name="msg" xml:space="preserve">
        <value>Deutsch</value>
  </data>
</root>
```

The complete code of another resource file (`Default.fr.resx`) is shown in Listing 25.8:

Listing 25.8: Complete Code for the `Default.fr.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
  Microsoft ResX Schema

  Version 2.0

  The primary goals of this format is to allow a simple XML format
  that is mostly human readable. The generation and parsing of the
  various data types are done through the TypeConverter classes
  associated with the data types.

  Example:

  ... ado.net/XML headers & schema ...
  <resheader name="resmimetype">text/microsoft-resx</resheader>
  <resheader name="version">2.0</resheader>
  <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
    ...</resheader>
  <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
    ...</resheader>
  <data name="Name1"><value>this is my long string</value><comment>this is a
    comment</comment></data>
  <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
  <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
         <value>[base64 mime encoded serialized .NET Framework object]</value>
  </data>
  <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
    microsoft.net.object.bytearray.base64">
         <value>[base64 mime encoded string representing a byte array form of the .NET
           Framework object]</value>
         <comment>This is a comment</comment>
  </data>

  There are any number of "resheader" rows that contain simple
  name/value pairs.

  Each data row contains a name, and value. The row also contains a
  type or mimetype. Type corresponds to a .NET class that support
  text/value conversion through the TypeConverter architecture.
  Classes that don't support this are serialized and stored with the
  mimetype set.

  The mimetype is used for serialized objects, and tells the
  ResXResourceReader how to depersist the object. This is currently not
  extensible. For a given mimetype the value must be set accordingly:

  Note - application/x-microsoft.net.object.binary.base64 is the format
  that the ResXResourceWriter will generate, however the reader can
  read any of the formats listed below.

  mimetype: application/x-microsoft.net.object.binary.base64
  value   : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
          : and then encoded with base64 encoding.

  mimetype: application/x-microsoft.net.object.soap.base64
  value   : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
          : and then encoded with base64 encoding.
    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
-->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
        <xsd:element name="root" msdata:IsDataSet="true">
        <xsd:complexType>
            <xsd:choice maxOccurs="unbounded">
            <xsd:element name="metadata">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
                </xsd:sequence>
                <xsd:attribute name="name" use="required" type="xsd:string" />
                <xsd:attribute name="type" type="xsd:string" />
                <xsd:attribute name="mimetype" type="xsd:string" />
                <xsd:attribute ref="xml:space" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="assembly">
                <xsd:complexType>
                    <xsd:attribute name="alias" type="xsd:string" />
                    <xsd:attribute name="name" type="xsd:string" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="data">
                <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="value" type="xsd:string" minOccurs="0"
                     msdata:Ordinal="1" />
                    <xsd:element name="comment" type="xsd:string"
                     minOccurs="0" msdata:Ordinal="2" />
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string" use="required"
                 msdata:Ordinal="1" />
                <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                <xsd:attribute name="mimetype" type="xsd:string"
                 msdata:Ordinal="4" />
                <xsd:attribute ref="xml:space" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="resheader">
                <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="value" type="xsd:string" minOccurs="0"
                     msdata:Ordinal="1" />
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string" use="required" />
                </xsd:complexType>
            </xsd:element>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    </xsd:schema>
    <resheader name="resmimetype">
        <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="version">
        <value>2.0</value>
    </resheader>
    <resheader name="reader">
        <value>System.Resources.ResXResourceReader, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="writer">
        <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <data name="msg" xml:space="preserve">
        <value>Français</value>
    </data>
</root>
```

The complete code of another resource file (`Default.resx`) is shown in Listing 25.9:

Listing 25.9: Complete Code for the `Default.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
  Microsoft ResX Schema

  Version 2.0

  The primary goals of this format is to allow a simple XML format
  that is mostly human readable. The generation and parsing of the
  various data types are done through the TypeConverter classes
  associated with the data types.

  Example:

  ... ado.net/XML headers & schema ...
  <resheader name="resmimetype">text/microsoft-resx</resheader>
  <resheader name="version">2.0</resheader>
  <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
    ...</resheader>
  <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
    ...</resheader>
  <data name="Name1"><value>this is my long string</value><comment>this is a
    comment</comment></data>
  <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
  <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
        <value>[base64 mime encoded serialized .NET Framework object]</value>
  </data>
  <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
    microsoft.net.object.bytearray.base64">
        <value>[base64 mime encoded string representing a byte array form of the .NET
          Framework object]</value>
        <comment>This is a comment</comment>
  </data>

  There are any number of "resheader" rows that contain simple
  name/value pairs.

  Each data row contains a name, and value. The row also contains a type or mimetype.
  Type corresponds to a .NET class that support text/value conversion through the
  TypeConverter architecture.
  Classes that don't support this are serialized and stored with the mimetype set.

  The mimetype is used for serialized objects, and tells the
  ResXResourceReader how to depersist the object. This is currently not
  extensible. For a given mimetype the value must be set accordingly:
    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
    -->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
    <xsd:element name="root" msdata:IsDataSet="true">
        <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
              <xsd:element name="metadata">
              <xsd:complexType>
                    <xsd:sequence>
                    <xsd:element name="value" type="xsd:string" minOccurs="0" />
                    </xsd:sequence>
                    <xsd:attribute name="name" use="required" type="xsd:string" />
                    <xsd:attribute name="type" type="xsd:string" />
                    <xsd:attribute name="mimetype" type="xsd:string" />
                    <xsd:attribute ref="xml:space" />
              </xsd:complexType>
              </xsd:element>
              <xsd:element name="assembly">
              <xsd:complexType>
                    <xsd:attribute name="alias" type="xsd:string" />
                    <xsd:attribute name="name" type="xsd:string" />
              </xsd:complexType>
              </xsd:element>
              <xsd:element name="data">
              <xsd:complexType>
                    <xsd:sequence>
                          <xsd:element name="value" type="xsd:string" minOccurs="0"
                            msdata:Ordinal="1" />
                          <xsd:element name="comment" type="xsd:string"
                            minOccurs="0" msdata:Ordinal="2" />
                    </xsd:sequence>
                    <xsd:attribute name="name" type="xsd:string" use="required"
                      msdata:Ordinal="1" />
                    <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                    <xsd:attribute name="mimetype" type="xsd:string"
                      msdata:Ordinal="4" />
                    <xsd:attribute ref="xml:space" />
              </xsd:complexType>
              </xsd:element>
              <xsd:element name="resheader">
              <xsd:complexType>
```

```
                        <xsd:sequence>
                              <xsd:element name="value" type="xsd:string" minOccurs="0"
                                msdata:Ordinal="1" />
                        </xsd:sequence>
                        <xsd:attribute name="name" type="xsd:string" use="required" />
                  </xsd:complexType>
                  </xsd:element>
            </xsd:choice>
            </xsd:complexType>
      </xsd:element>
      </xsd:schema>
      <resheader name="resmimetype">
            <value>text/microsoft-resx</value>
      </resheader>
      <resheader name="version">
            <value>2.0</value>
      </resheader>
      <resheader name="reader">
            <value>System.Resources.ResXResourceReader, System.Windows.Forms,
              Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
      </resheader>
      <resheader name="writer">
            <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
              Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
      </resheader>
      <data name="msg" xml:space="preserve">
            <value>English</value>
      </data>
    </root>
```

Listings 25.7, 25.8, and 25.9 contain the complete code of all the resource files Default.de.resx, Default.fr.resx and Default.resx respectively, of the `ExplicitLocalizationVB` application.

Performing Localization

You can generate resources for Server control properties by using implicit and explicit localization expressions. You can apply localization expressions to the `@ Page` directive and other sections of HTML to identify the areas to be localized before generating the resources. However, for localizing a Web application, other contents, such as the HTML page title, and static content in the Web application should also be localized.

Elements of HTML

HTML Controls cannot be localized using implicit or explicit expressions, unless the controls run on the server. To specify that the HTML controls run on the server, you need to include the `runat="server"` attribute in the `@ Page` directive. After the HTML controls are marked as Server controls, local resources are automatically generated for the localizable properties, such as the Text and ToolTip properties of the controls. HTML Server controls can also be bound to implicit or explicit expressions; the latter is generated by using the Expressions dialog box as described earlier in the explicit localization section. You can also bind the HTML elements, such as page titles and style sheet links of the page directive, to the resources.

Static Content

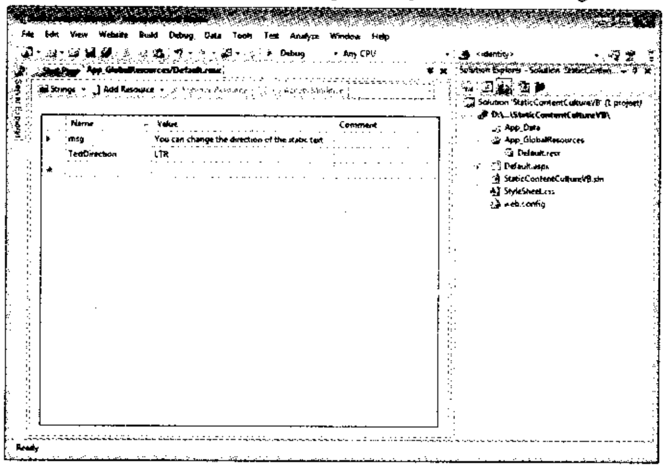
You can use the localization expressions to localize the properties of controls used on a Web page. However, sometimes the Web application also includes static content, such as copyright information or disclaimer information. ASP.NET Framework allows you to localize the static content. To localize a static content in a Web application, the `<Localize runat="server">` tag is used.

For example, consider that you need to add the disclaimer information in a Web application. The code to be added in the `Default.aspx` page of the Web application to localize static content is given here:

```
<asp:Localize runat="server" meta:resourcekey="Information"> Disclaimer: This
   sample application is not intended to provide accurate or
   up-to-date information.</asp:Localize>
```

In the preceding code, a local resource value is generated for the `Text` property of the `Localize` control. You can also set the text direction for the static content used in a Web application for a specific local culture of a region, where the content is read from right to left or vice versa. For this, you can define a property for text direction in either local resources or global resources and use explicit localization expressions to set the direction property of the text. To do so, create a new Web application in Visual Studio 2008 named `StaticContentCultureVB`. You can find the code of `StaticContentCultureVB` application in the `Code\ASP.NET\Chapter 25\StaticContentCultureVB` folder on the CD, and follow these steps:

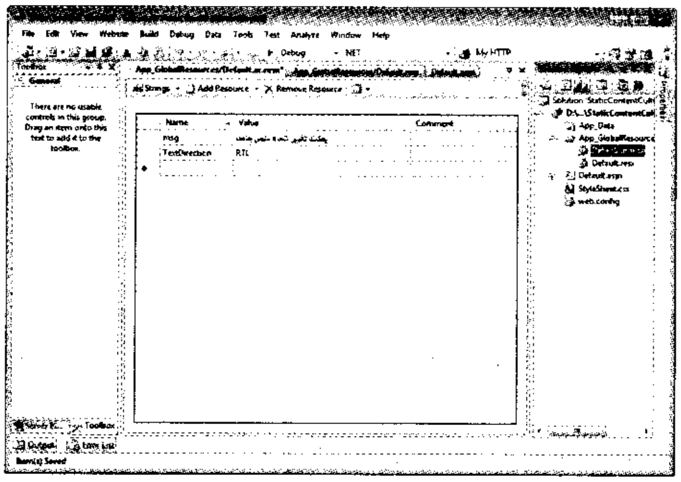
1. Add the `App_GlobalResources` folder in the `StaticContentCultureVB` Web application.
2. Now, add a new item `Default.resx` in the `App_GlobalResources` folder in the Solution Explorer and set the Name field as msg and TextDirection and their corresponding value in the Value field as You can change the direction of the static text and LTR respectively, as shown in Figure 25.21:



Name	Value	Comment
msg	You can change the direction of the static text	
TextDirection	LTR	

Figure 25.21: Resource Editor for the Default.resx File

3. Add another new item `Default.ar.resx` for the Arabic language and set the values for the Name and Value fields in the Resource editor, as shown in Figure 25.22:



Name	Value	Comment
msg	[illegible]	
TextDirection	RTL	

Figure 25.22: Resource Editor for the Default.ar.resx File

4. Now, add a Label control in the Design view of the `Default.aspx` file and add the code, as shown in Listing 25.10, in the `Default.aspx` page of the Web application to set the direction property of the static content:

Listing 25.10: Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" Culture="auto" UICulture="auto"
   CodeFile="Default.aspx.vb"
   Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html id="Html1" runat="server" dir="<%$ Resources:default,TextDirection %>"
   xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <title>StaticContentCulture Example</title>
   <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body id="Body1" runat="server" dir="<%$ Resources:default,TextDirection %>">
   <form id="form1" runat="server">
         <div>
         <div id="header">

         </div>
         <div id="sidebar" style="height: 1041px">
               <div id="nav">
                     &nbsp;
               </div>
         </div>
         <div id="content">
         <div class="itemContent">
               <br />
               <br />
               &nbsp;<asp:Label ID="Label1" runat="server" Text="<%$
                  Resources:default,msg %>"></asp:Label>
               <div id="footer">
                     &nbsp;<p class="left">
                     All content copyright &copy; Kogent Solution Inc.</p>
               </div>
         </div>
         </div>
         </div>
   </form>
</body>
</html>
```

The complete code for the resource file, `Default.ar.resx`, is shown in Listing 25.11:

Listing 25.11: Code for the `Default.ar.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>
   <!--
   Microsoft ResX Schema

   Version 2.0

   The primary goals of this format is to allow a simple XML format
   that is mostly human readable. The generation and parsing of the
   various data types are done through the TypeConverter classes
   associated with the data types.

   Example:

   ... ado.net/XML headers & schema ...
   <resheader name="resmimetype">text/microsoft-resx</resheader>
```

```
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
      ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
      ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
      comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
        <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
     microsoft.net.object.bytearray.base64">
        <value>[base64 mime encoded string representing a byte array form of the .NET
          Framework object]</value>
        <comment>This is a comment</comment>
    </data>

    There are any number of "resheader" rows that contain simple
    name/value pairs.

    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.

    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:

    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
    -->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
          <xsd:element name="root" msdata:IsDataSet="true">
          <xsd:complexType>
                <xsd:choice maxOccurs="unbounded">
                <xsd:element name="metadata">
                      <xsd:complexType>
                      <xsd:sequence>
                      <xsd:element name="value" type="xsd:string" minOccurs="0" />
                      </xsd:sequence>
                      <xsd:attribute name="name" use="required" type="xsd:string" />
                      <xsd:attribute name="type" type="xsd:string" />
                <xsd:attribute name="mimetype" type="xsd:string" />
                <xsd:attribute ref="xml:space" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="assembly">
                <xsd:complexType>
                    <xsd:attribute name="alias" type="xsd:string" />
                    <xsd:attribute name="name" type="xsd:string" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="data">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
                  msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string"
                  minOccurs="0" msdata:Ordinal="2" />
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string"
                  use="required" msdata:Ordinal="1" />
                <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                <xsd:attribute name="mimetype" type="xsd:string"
                  msdata:Ordinal="4" />
                <xsd:attribute ref="xml:space" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="resheader">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
                  msdata:Ordinal="1" />
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string" use="required" />
                </xsd:complexType>
            </xsd:element>
        </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    </xsd:schema>
    <resheader name="resmimetype">
        <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="version">
        <value>2.0</value>
    </resheader>
    <resheader name="reader">
        <value>System.Resources.ResXResourceReader, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="writer">
        <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <data name="msg" xml:space="preserve">
    <value> يمكنك تغير اتجاه النص هنا </value>

    </data>
    <data name="TextDirection" xml:space="preserve">
        <value>RTL</value>
    </data>
</root>
```

The complete code for the resource file, `Default.resx`, is shown in Listing 25.12:

Listing 25.12: Code for the `Default.resx` File

```
<?xml version="1.0" encoding="utf-8"?>
<root>

  <!--
  Microsoft ResX Schema
  Version 2.0

  The primary goals of this format is to allow a simple XML format
  that is mostly human readable. The generation and parsing of the
  various data types are done through the TypeConverter classes
  associated with the data types.

  Example:

  ... ado.net/XML headers & schema ...
  <resheader name="resmimetype">text/microsoft-resx</resheader>
  <resheader name="version">2.0</resheader>
  <resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms,
    ...</resheader>
  <resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms,
    ...</resheader>
  <data name="Name1"><value>this is my long string</value><comment>this is a
    comment</comment></data>
  <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
  <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
        <value>[base64 mime encoded serialized .NET Framework object]</value>
  </data>
  <data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-
    microsoft.net.object.bytearray.base64">
        <value>[base64 mime encoded string representing a byte array form of the .NET
          Framework object]</value>
        <comment>This is a comment</comment>
  </data>
  There are any number of "resheader" rows that contain simple name/value pairs.

  Each data row contains a name, and value. The row also contains a
  type or mimetype. Type corresponds to a .NET class that support
  text/value conversion through the TypeConverter architecture.
  Classes that don't support this are serialized and stored with the
  mimetype set.
  The mimetype is used for serialized objects, and tells the
  ResXResourceReader how to depersist the object. This is currently not
  extensible. For a given mimetype the value must be set accordingly:

  Note - application/x-microsoft.net.object.binary.base64 is the format
  that the ResXResourceWriter will generate, however the reader can
  read any of the formats listed below.

  mimetype: application/x-microsoft.net.object.binary.base64
  value   : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Binary.BinaryFormatter
          : and then encoded with base64 encoding.

  mimetype: application/x-microsoft.net.object.soap.base64
  value   : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
          : and then encoded with base64 encoding.

  mimetype: application/x-microsoft.net.object.bytearray.base64
  value   : The object must be serialized into a byte array
          : using a System.ComponentModel.TypeConverter
```

```
            : and then encoded with base64 encoding.
    -->
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
          <xsd:element name="root" msdata:IsDataSet="true">
          <xsd:complexType>
                  <xsd:choice maxOccurs="unbounded">
                  <xsd:element name="metadata">
                          <xsd:complexType>
                          <xsd:sequence>
                          <xsd:element name="value" type="xsd:string" minOccurs="0" />
                          </xsd:sequence>
                          <xsd:attribute name="name" use="required" type="xsd:string" />
                          <xsd:attribute name="type" type="xsd:string" />
                          <xsd:attribute name="mimetype" type="xsd:string" />
                          <xsd:attribute ref="xml:space" />
                          </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="assembly">
                  <xsd:complexType>
                          <xsd:attribute name="alias" type="xsd:string" />
                          <xsd:attribute name="name" type="xsd:string" />
                  </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="data">
                  <xsd:complexType>
                          <xsd:sequence>
                          <xsd:element name="value" type="xsd:string" minOccurs="0"
                            msdata:Ordinal="1" />
                          <xsd:element name="comment" type="xsd:string" minOccurs="0"
                            msdata:Ordinal="2" />
                          </xsd:sequence>
                          <xsd:attribute name="name" type="xsd:string" use="required"
                            msdata:Ordinal="1" />
                          <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
                          <xsd:attribute name="mimetype" type="xsd:string"
                            msdata:Ordinal="4" />
                          <xsd:attribute ref="xml:space" />
                  </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="resheader">
                  <xsd:complexType>
                  <xsd:sequence>
                          <xsd:element name="value" type="xsd:string" minOccurs="0"
                          msdata:Ordinal="1" />
                          </xsd:sequence>
                          <xsd:attribute name="name" type="xsd:string" use="required" />
                  </xsd:complexType>
                  </xsd:element>
          </xsd:choice>
          </xsd:complexType>
    </xsd:element>
    </xsd:schema>
    <resheader name="resmimetype">
          <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="version">
          <value>2.0</value>
    </resheader>
    <resheader name="reader">
          <value>System.Resources.ResXResourceReader, System.Windows.Forms,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="writer">
        <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
          Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <data name="msg" xml:space="preserve">
        <value>You can chagne the direction of the static text</value>
    </data>
    <data name="TextDirection" xml:space="preserve">
        <value>LTR</value>
    </data>
</root>
```

5. Now, run the Web application. The output of the `StaticContentCultureVB` Web application, when the preferred language in the browser settings is changed to English, is shown in Figure 25.23:

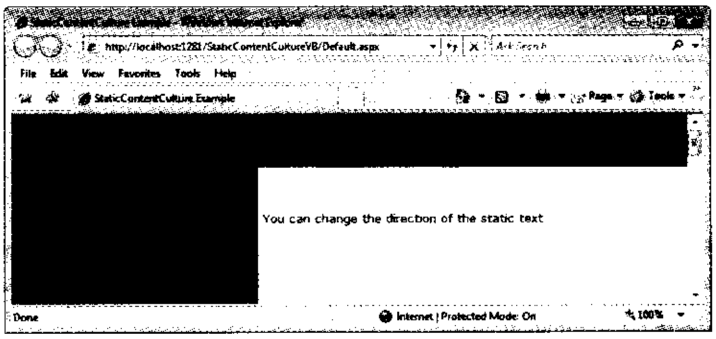

You can change the direction of the static text

Figure 25.23: Output Shown in English Language

The output of the Web application, when the preferred language in browser settings is changed to Arabic, is shown in Figure 25.24:

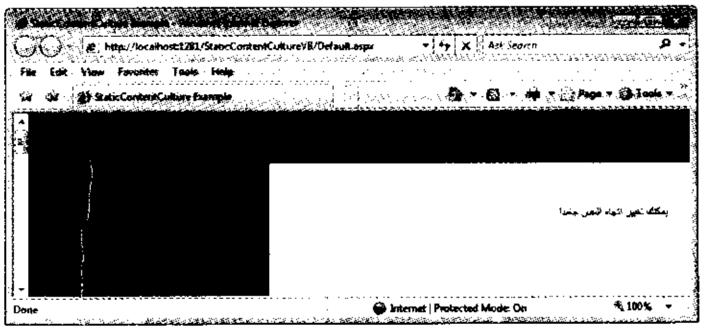


Figure 25.24: Output Shown in Arabic Language

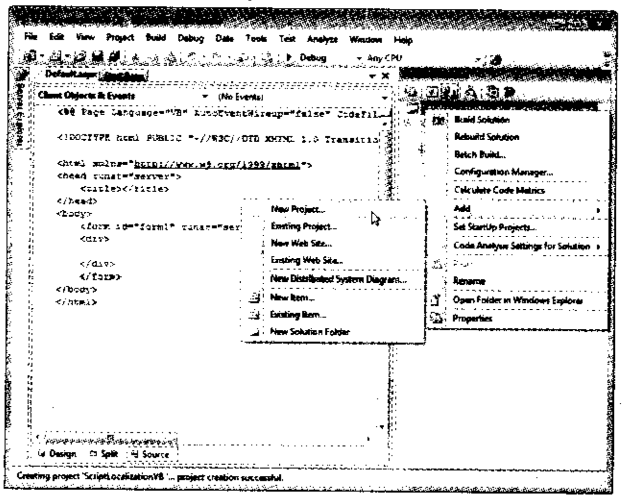
Localizing Script Files

You can localize your script files with the help of the `ScriptManager` control. The `ScriptManager` control enables certain automatic behaviors for localized applications. These include the following:

- Automatically locating script files based on settings and naming conventions. For instance, it loads debug-enabled scripts when in debugging mode, and loads localized scripts based on the browser's UI selection.
- Enabling the definition of cultures, including custom cultures.
- Caching scripts to efficiently manage many requests.

Now, let's create a Web application named `ScriptLocalizationVB`. `You can find the code of ScriptLocalizationVB application in the Code\ASP.NET\Chapter 25\ScriptLocalizationVB folder on the CD`, and follow these steps to localize your script files:

1. In the `ScriptLocalizationVB` Web application, add a Class Library project named `LocalizingResources` by right-clicking the name of the Web application in the Solution Explorer, and select Add→`New Project`, as shown in Figure 25.25:

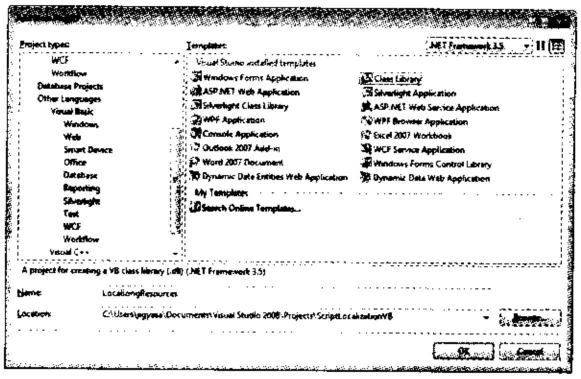


```
<%@ Page Language="VB" AutoEventWireup="false" CodeFil
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitio
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="ser
    <div>

    </div>
    </form>
</body>
</html>
```

Creating project 'ScriptLocalizationVB'... project creation successful.

Figure 25.25: Adding New Project in the ScriptLocalization Application

2. The New Project option opens the Add New Project dialog box (Figure 25.26). Select the Class Library template and change the name of the Class Library project as `LocalizingResources`, as shown in Figure 25.26:



A project for creating a VB class library (.dll) (.NET Framework 3.5)

Name: LocalizingResources

Location: C:\Users\[illegible]\Documents\Visual Studio 2008\Projects\ScriptLocalizationVB

Figure 25.26: Add New Project Dialog Box

3. Now, click the OK button. It will add the `LocalizingResources` application to the Solution Explorer.
4. In the `LocalizingResources` application, add the Jscript file named `CreateScript.js` and change the value of the `Build Action` property of the script file to `Embedded Resource`.
5. Add the following code in the `CreateScript.js` file:

```
function CreateScript(fileName)
{
    alert(Message.FileCreated.replace(/FILENAME/, fileName));
}
```

6. Now add two resource files in the `LocalizingResources` application as `CreateResource.resx` for English language and `CreateResource.ru.resx` for Russian language and add the following resource string to the `CreateResource.resx` file:

- **FileCreated** – The file FILENAME has been created

Also add the following resource string to the `CreateResource.ru.resx` file:

- **FileCreated** – FILENAME файл был создан

7. After adding the resource strings in the resource files, add the following lines of code in the `AssemblyInfo` file of the `LocalizingResources` application:

```
<Assembly: System.Web.UI.WebResource("LocalizingResources.CreateScript.js",
  "text/javascript")>
<Assembly: System.Web.UI.ScriptResource("LocalizingResources.CreateScript.js",
  "LocalizingResources.CreateResource", "Message")>
```

8. Now, add references of the `System.Web` and `System.Web.Extensions` assemblies in the `LocalizingResources` project. You also need to add the reference of the `LocalizingResources` project in the `ScriptLocalizationVB` project. You can see the `LocalizingResources` project in the Projects tab of the Add Reference dialog box, as shown in Figure 25.27:

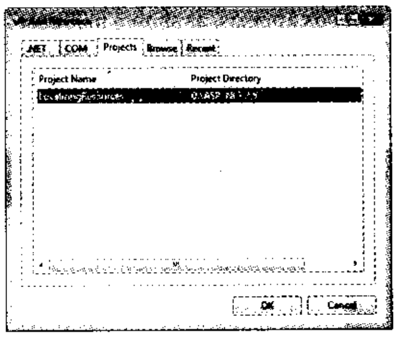


Figure 25.27: Add Reference Dialog Box

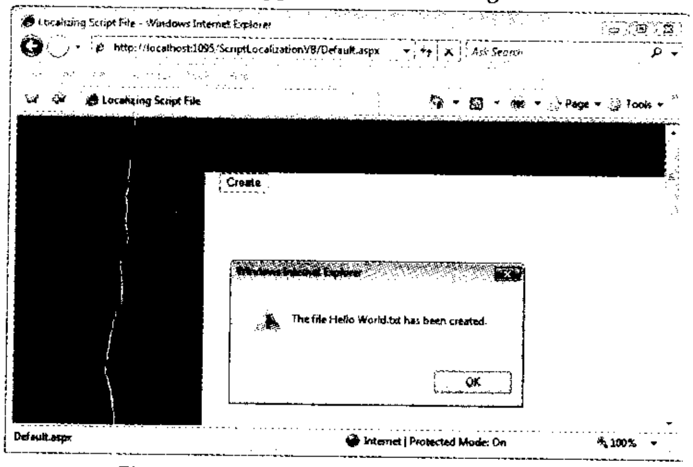
9. After adding the references, open the `Default.aspx` page of the `ScriptLocalizationVB` application and add the code shown in Listing 25.13:

Listing 25.13: Code for the `Default.aspx` Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Culture="auto"
    UICulture="auto" Inherits="_Default" %>
<!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>Localizing Script File</title>
  <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form id="form1" runat="server">
        <div>
        <div id="header">

        </div>
        <div id="sidebar" style="height: 1041px">
              <div id="nav">
                    &nbsp;
              </div>
        </div>
        <div id="content">
              <div class="itemContent">
                    <asp:ScriptManager ID="ScriptManager1" runat="server"
                      EnableScriptLocalization="true">
                    <Scripts>
                          <asp:ScriptReference Assembly="LocalizingResources"
                            Name="LocalizingResources.CreateScript.js" />
                    </Scripts>
                    </asp:ScriptManager>
```

```
                         <asp:Button ID="btnCreate" runat="Server"
                           OnClientClick="CreateScript('Hello World.txt');"
                         Text="Create" />
                         <div id="footer">
                              &nbsp;<p class="left">
                              All content copyright &copy; Kogent Solution Inc.</p>
                         </div>
                    </div>
               </div>
               </div>
     </form>
   </body>
   </html>
```

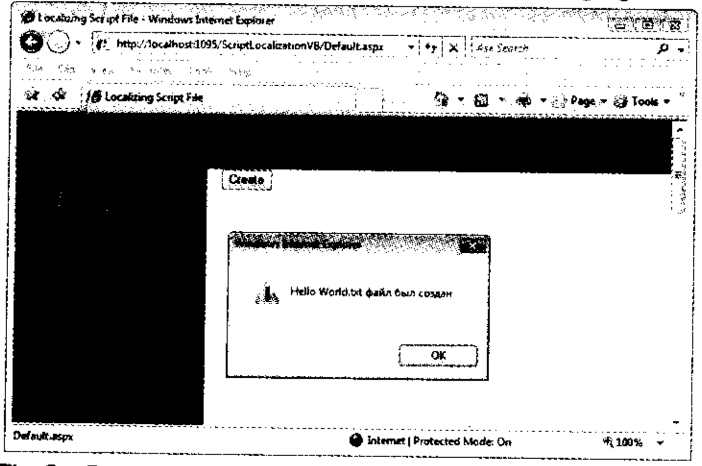
10. Now press F5 to run the Web application. Notice that the Create button is displayed in the browser (Figure 25.28).
11. Click the Create button. A message box appears, as shown in Figure 25.28:


Localizing Script File - Windows Internet Explorer
http://localhost:1095/ScriptLocalizationVB/Default.aspx
Ask Search
Localizing Script File
Page
Tools
Create
OK
Default.aspx
Internet | Protected Mode: On
100%

The file Hello World.txt has been created.

Figure 25.28: The Confirmation Message Box

12. In Figure 25.28, the preferred language is English. Change the preferred language to Russian. The output of the `ScriptLocalization` application after changing the preferred language is shown in Figure 25.29:


Localizing Script File - Windows Internet Explorer
http://localhost:1095/ScriptLocalizationVB/Default.aspx
Ask Search
Localizing Script File
Page
Tools
Create
OK
Default.aspx
Internet | Protected Mode: On
100%

Hello World.txt файл был создан

Figure 25.29: The Confirmation Message Box after Selecting Russian as the Preferred Language

Figure 25.29 displays a message box with a message in Russian language, which you have selected in your browser.

Now, let's discuss deploying the resources.

Deploying Resources

In the earlier versions of ASP.NET, the resources were deployed using a hub and spoke model. The hub refers to the main assembly that contains the non-localizable executable code and the resources for a default culture. Each spoke connects to a satellite assembly that contains the resources for a single culture, but does not contain any code.

However, ASP.NET 2.0 supports site pre-compilation. This implies that the Web pages and the local resources are compiled into deployable assemblies. ASP.NET also supports dynamic runtime compilation. In dynamic runtime compilation, the source code, local and global resources are deployed in raw source format. The runtime compiler then parses the Web pages and generates assemblies for deployment.

A hybrid of site pre-compilation and dynamic runtime compilation is also possible in ASP.NET 3.5. This implies that ASP.NET 3.5 provides with the ability to deploy Web pages and resources as source and compile all these source files into binary assemblies. Compiling files into binary assemblies enables you to edit the Web page layout and resource content. You do not need to dynamically recompile the assemblies of unaffected applications.

Summary

In this chapter, we have introduced the concept of Globalization of Web applications. We have learned how to create local and global resources in a Web application, and perform implicit and explicit localization. In addition, we have learned how to localize the script files in a Web applicatiou.

In the next chapter, you learn how to develop rich-interactive applications with Silverlight.

Quick Revise

Q1. **What is globalization?**

Ans: Globalization of Web application is the process of designing and developing Web applications that can display Web page contents in different languages and cultures.

Q2. **Which namespaces are used to globalize Web applications in .NET Framework?**

Ans: To globalize Web applications in .NET Framework, the following two namespaces are used:

- **System.Globalization**—Contains the classes defining information related to culture, such as language, country, format, and pattern for date, currency, and numbers
- **System.Resources**—Contains the classes and interfaces that you can use to create, store, and manage culture-specific resources used in a Web application

Q3. **What is internationalization and localization of a Web application?**

Ans: The process of Internationalization involves identifying different contents for different locales, such as date and time formats. In general, to serve the purpose, you need to write the source code for the Web application such that the code is dynamically formatted to present the content of the Web application in the desired locale. The process of Internationalization ensures that the source code of a Web application is not modified when the application needs to be customized for specific culture or regions whereas localization refers to the process of making changes in a Web application to meet the language and cultural requirements of users from a specific geographical location. The localization process involves configuring a Web application for a given culture. This implies translating and formatting content, such as time and date, according to the culture for which the Web application is to be localized. The elements, such as numeric data, date formats, and currency, are customized for a given culture in Web applications.

Q4. What do you understand by Culture and UICulture values?

Ans: The Culture value determines the functions, such as date and currency. These Culture values are used to format data and numbers in a Web page whereas; the UICulture value determines the resources that are loaded on a Web page of a Web application. Resources in a Web application represent data, such as strings or images, displayed in the UI of the Web application.

Q5. What is resource file?

Ans: A file that contains source code for the various resources used in a Web application is called the resource file.

Q6. Which method is used in the CultureInfo class to view the complete list of all the culture options according to which you can customize a Web application?

Ans: CultureInfo.GetCultures method

Q7. To translate the content of a Web application from one language to another, you need to place the string value in a resource file. (True/False)

Ans: True

Q8. What is the difference between global resources and local resources?

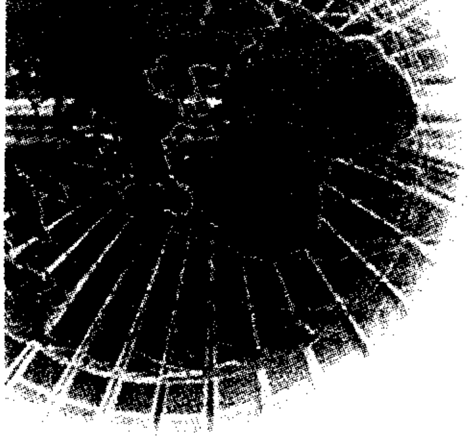
Ans: The global resources are accessible from all the Web pages of an ASP.NET Web application, while the local resources are accessible from a single Web page of the ASP.NET Web application.

Q9. The App_GlobalResources folder can exist in any directory in the ASP.NET Web application. (True/False)

Ans: False

Q10. What should be the naming convention of the resource file?

Ans: The naming convention of the resource file should match the base name of the Web page that accesses the resource file. The base name should be followed by the language name and the culture name for which the resource file is created. The name of the resource file should end with the extension .resx.



26

Developing Rich-Interactive Applications with Silverlight

In the initial years of the Internet and Web development, websites were developed using only Hypertext Markup Language (HTML). The content built using only HTML for the Web pages is mostly static. Gradually, with the advent of other new technologies and languages, such as Cascading Style Sheets (CSS), XHTML, and JavaScript, it is now possible to add user interactivity and a little bit of dynamism to Web pages. With the passing years, some other technologies, such as Adobe Flash and JavaFX, gave a new dimension of interactivity to Web pages as these technologies facilitated the use of animations and multimedia in Web pages. Websites made using these technologies gave a huge boost to various fields of business, such as e-learning and online advertising. However, some of these technologies are expensive to deploy, varied in performance, and also have inadequate compatibility with different programming technologies.

To overcome the limitations of earlier technologies, such as Adobe Flash and the ever increasing demand of Rich Internet Applications (RIA), Microsoft brought out Silverlight. Formerly known as Windows Presentation Foundation/Everywhere (WPF/E), Silverlight is an inexpensive small plug-in that can work in a cross-platform, cross-browser, and cross-device environment. Silverlight is supported by both Microsoft Windows and Apple Macintosh family of operating systems as well as popular Web browsers, such as Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari.

Silverlight is a Web-based technology that allows Web designers and developers to stretch the boundaries of Web application development. It is an integration of the rich user interface (UI) of desktop applications and the transparency of other Web languages, such as HTML and JavaScript. Silverlight allows you to develop Web applications that contain high-fidelity multimedia content and eye-catching visual effects. Web designers and developers have already embraced Silverlight for building high-quality interactive Web applications.

In this chapter, you learn about the main features and architecture of Silverlight. You also learn how to install Silverlight and to incorporate Silverlight functionality in Visual Studio 2008 projects. This chapter also describes the association between Silverlight and ASP.NET. Now, let's begin with the main features of Silverlight.

Main Features of Silverlight

As stated earlier, Silverlight was previously known as WPF/E because it is a subset of WPF, inheriting many functionalities of WPF. For example, Silverlight supports XAML, 2-D vector graphics, animations, and multimedia. However, certain features, such as 3-D graphics and hardware rendering of WPF, are not available to Silverlight. You may wonder about the difference between XAML Browser Applications (XBAPs) in WPF and Silverlight. One of the notable differences between the two is that XBAPs can run only in Internet Explorer and Firefox, whereas Silverlight applications can run in multiple Web browsers. Secondly, XBAPs require .NET Framework to be available on the users' computers, while Silverlight provides the necessary components on its own and does not need .NET Framework. Thirdly, an XBAP has access to the complete WPF functionality, while a Silverlight application has only limited WPF functionality.

The first version of Silverlight unveiled by Microsoft was Silverlight 1.0. This version was released in 2007 and was based on the JavaScript application model. You could incorporate the functionality of Silverlight 1.0 in .NET Framework 2.0 and Visual Studio 2005 applications. The latest version of Silverlight is Silverlight 2.0, which is fully supported by Visual Studio 2008 SP1.

Silverlight 2.0 offers a variety of innovative features that benefit the Web developers and designers to build intriguing Web sites. As compared to the previous versions, Silverlight 2.0 has many improvements that facilitate an easier and quicker way of developing Web sites. For example, Silverlight 2.0 has an improved programming model, comprehensive UI framework, and support for Deep Zoom technology, 2-D graphics, animations, multimedia, and networking. Now, let's go through these main features of Silverlight 2.0.

Improved Programming Model

Silverlight technology supports the .NET Framework programming model to develop Silverlight applications easily. Silverlight leverages many functionalities and features of .NET Framework, such as type safety and exception handling. It also includes many classes from the base class library, and offers you the provision of working with IO, collections and generics, and threading. You can use any of the .NET languages, such as VB and C# to develop Silverlight applications. You can also use other dynamic language such as IronPython and IronRuby. Furthermore, you can integrate Silverlight functionalities with the existing ASP.NET Web applications. You can combine ASP.NET AJAX and Silverlight to get the best of both the technologies. Silverlight also supports Language Integrated Query (LINQ) to Objects, which facilitates you to work with various types of data in the Web applications.

Note that Silverlight seamlessly integrates with the HTML, JavaScript Document Object Model (DOM) as well as with XAML parser providing you with the facility to work with HTML, JavaScript, and XAML simultaneously. In this way, Silverlight extends a cohesive platform for developing interactive and superior quality Web applications.

Comprehensive UI Framework

Silverlight has a comprehensive UI framework to allow you to design the UI of Web applications by using the predefined controls, styles, themes, and templates. Silverlight offers a wide range of controls and layout features to build exceptionally attractive Web applications. Most of the controls and layout features are the same as those available in WPF. For example, the Grid, StackPanel, Calendar, Button, TextBox, and RadioButton controls of WPF are also available in Silverlight. You can also use styles and control templates in the Silverlight applications. Moreover, there are additional controls, such as MultiScaleImage that are specific to Silverlight. You can also use the data binding and data manipulation controls, such as DataGrid. These new controls were designed from scratch to give an extra edge to the Web applications.

These predefined Silverlight controls and layout features help you to quickly and easily develop some high-end Web applications. You can set the properties of these controls to get the desired effect. Note that these properties can also be set using XAML. XAML is the facilitator for separating the design and the code of Web applications.

NOTE

In Silverlight 2.0, a new feature called the Visual State Manager (for Microsoft Expression Blend software, which is a tool that allows you to design the UI of Web application using Silverlight) is available. The Visual State Manager allows you to easily modify existing control templates and create new control templates.

Support for Deep Zoom Technology

Silverlight 2.0 includes a new feature called Deep Zoom, which allows you to zoom high-resolution images in the applications. You can use the Deep Zoom technology to deliver unbelievably creative and uniquely interactive Web applications. With the Deep Zoom technology, you can easily, quickly, and smoothly zoom in or out multiple photographs simultaneously. You can see a highly detailed and fine view of the photographs by using the MultiScaleImage control available in Silverlight 2.0.

Support for 2-D Graphics, Animations, and Multimedia

Silverlight has built-in support for using 2-D vector graphics and animations in Web applications. You can create 2-D shapes, such as rectangles and ellipses, geometries, and brushes. You can also transform and animate the shapes and geometries. In addition, you can make your Web applications dynamic by using storyboards, controlling the animation timeline and speed, and much more.

There is an extensive support in Silverlight for captivating multimedia experiences on the Web. You can include .jpeg or .png images in the applications to give them an attractive UI. In addition, Silverlight supports various multimedia formats, such as .wma, .wmv, and .mp3 and has some of the essential codecs for playing back multimedia content. You can also use High-Definition (HD) quality video (up to 720p) in Silverlight applications. Silverlight also offers protection through authentication and authorization of users and Digital Rights Management (DRM) for the multimedia content that you add to your Web applications.

Support for Networking

Silverlight 2.0 contains classes to support networking and predefined sockets. It allows you to use various HTTP, XML, RSS, and REST services. Furthermore, you can use an XML format to use the data and resources on other Web sites. Silverlight extends the cross-domain support for networking. You can also use Silverlight 2.0 for communication with sockets over standard network protocols, such as IPv4 and IPv6.

Now, let's now move ahead to learn about the Silverlight architecture.

Architecture of Silverlight

As you know, the primary components of Silverlight include HTML, JavaScript, XAML, and .NET Framework. While developing Silverlight applications, you will be using a combination of varied features of these technologies. These individual technologies are incorporated in Silverlight as different components and services. Figure 26.1 depicts a pictorial representation of the Silverlight architecture:

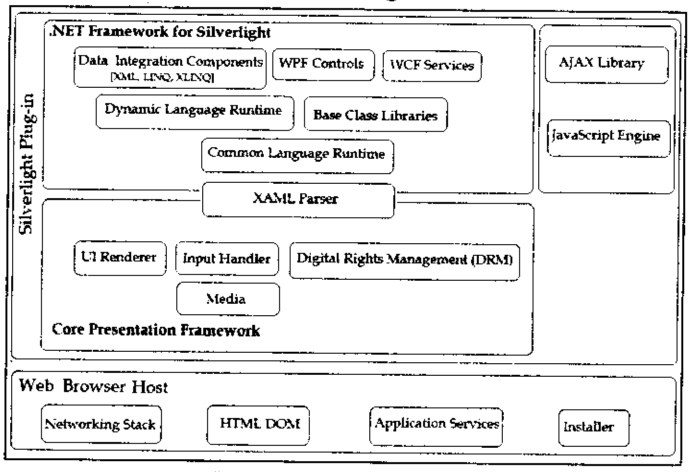


Figure 26.1: Architecture of Silverlight 2.0

As shown in Figure 26.1, Silverlight 2.0 is a blend of multifarious technologies and features. It has two main components, .NET Framework for Silverlight and the core presentation framework. Now, let's learn about these two components in detail.

The .NET Framework for Silverlight

Silverlight comes with a smaller version of.NET Framework, which has a toned down version of the Common Language Runtime (CLR) that offers some of the basic functionalities, such as common type system, type checking, debugging, and garbage collection.The .NET Framework for Silverlight component also offers certain class libraries that allow you to work with strings, collections, generics, reflection, and input/output. Some of the class libraries of Silverlight also provide a wide range of WPF controls such as Button, TextBox, and RadioButton. The .NET Framework for Silverlight component also allows you to integrate data in Silverlight applications through XML, LINQ, and XLINQ and manipulate the data through serialization.

The .NET Framework in Silverlight component also enables you to easily access data and services at remote locations through Windows Communication Foundation (WCF). In addition, you can use HTTP objects, cross-domain HTTP requests, RSS feeds, JSON, and SOAP services in Silverlight applications.

NOTE

The class libraries in the .NET Framework of Silverlight component are packaged and deployed with the Silverlight application on the users' computers.

The Core Presentation Framework

The core presentation framework is a component in Silverlight that offers the features and services for designing the UI of Silverlight applications through controls, styles, templates, 2-D vector graphics, animations, and multimedia. This component also allows you to incorporate user input and interactivity through various input devices, such as mouse and keyboard. It also allows you to bind data with various controls, and specify the UI layout of the Silverlight applications. Moreover, you can also manage the digital rights of the multimedia content used in your Silverlight applications through this component. The XAML parser for processing the XAML content in Silverlight applications is also provided by the core presentation framework.

Now, you are aware of what Silverlight is and how it is a robust and cohesive platform for developing interactive and rich Internet applications. Now, let's learn how to install Silverlight on your computer.

Installation of Silverlight

Silverlight does not come with any Web browser or Visual Studio 2008 but needs to be installed separately. Note that to run Silverlight-enabled Web applications, the users need not install .NET Framework 3.5 or Visual Studio 2008 on their computers. Instead, the users just need to install the Silverlight plug-in that becomes a part of the Web browser itself. The Silverlight plug-in provides a runtime environment in which the Silverlight applications run.

The Silverlight plug-in is freely available and downloadable from the Microsoft website that is located at http://www.microsoft.com/silverlight. However, prior to installing the Silverlight plug-in on your computer, you need to ensure that all the system requirements are met for a smooth installation. Though, the Silverlight plug-in can run on any operating system and Web browser, it is advisable to ensure whether the configuration of your computer supports Silverlight. Table 26.1 lists the operating systems and Web browsers that support Silverlight 2.0:

Table 26.1: Supporting Operating Systems and Web Browsers for the Silverlight 2.0 Plug-in

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Windows Vista	Supported		Supported	
Windows XP SP2	Supported	Supported	Supported	
Windows 2000		Supported (only Silverlight 2)		
Windows Server 2003 (except IA-64)	Supported	Supported	Supported	
Mac OS 10.4.8+ (PowerPC based)			Supported (only Silverlight 1.0)	Supported (only Silverlight 1.0)
Mac OS 10.4.8+ (Intel based)			Supported	Supported

Based on the different operating systems that support Silverlight, the minimal system requirements for installing Silverlight 2.0 are listed in Table 26.2:

Table 26.2: Minimum System Requirements for Silverlight 2.0

[illegible]	[illegible]	[illegible]	[illegible]
Processor Type	X86 or X64	PowerPC G4	Intel Core Duo
Processor Speed	500 MHz	800 MHz	1.83 GHz
RAM	128 MB	128 MB	128 MB

After you ensure that your computer has the compatible and necessary operating systems, Web browsers, and hardware configuration, you can download the Silverlight plug-in from the Microsoft website (`http://www.microsoft.com`). The actual size of Silverlight plug-in when installed on the computer is approximately 4 MB. Note that the Silverlight plug-in only provides the runtime environment to view the content of Silverlight applications. In order to develop Silverlight-based Web applications, you need to install the

Silverlight Software Development Kit (SDK) tools, which is about 72.7 MB (for Silverlight 2.0). You can install the Silverlight SDK separately from the Silverlight plug-in or install them together.

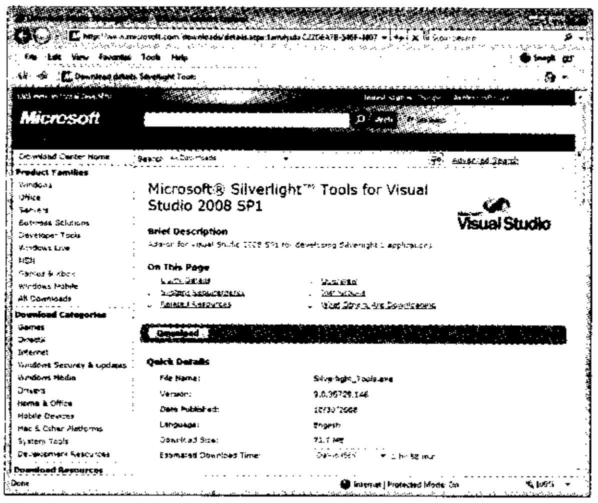
Now, let's install the Silverlight 2.0 plug-in and SDK together using the Silverlight 2.0 Tools add-in. For this follow these steps:

1. Open the Web browser (Internet Explorer) and go to `http://silverlight.net/GetStarted/`, as shown in Figure 26.2:

Figure 26.2: Opening the Microsoft website for Downloading Silverlight 2.0

2. Below Get Started Building Silverlight 2 Applications label, click the Install Silverlight Tools for Visual Studio 2008 SP1 link (shown in Figure 26.2).

As soon as click the link the Microsoft Silverlight Tools for Visual Studio 2008 SP1 Web page opens, as shown in Figure 26.3:



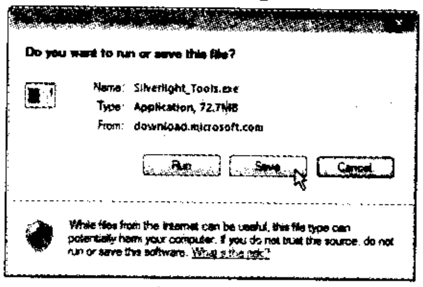
Silverlight_Tools.exe
9.0.30729.146
10/30/2008
English
71.7 MB

Figure 26.3: Preparing to Install Silverlight 2.0 Tools

On this Web page, you can see links to verify the system requirements and get a step-by-step explanation of Silverlight installation. If you scroll down the Web page, then you can also view the components that are installed with this add-in.

3. Now, to start downloading, you need to click the `Download` button. When you click that button, the File Download dialog box (Figure 26.4) appears prompting you to immediately run the Silverlight 2.0 Tools add-in or install and then run the add-in; to install the add-in on your computer.

4. Click the Save button on the dialog box as shown in Figure 26.4:



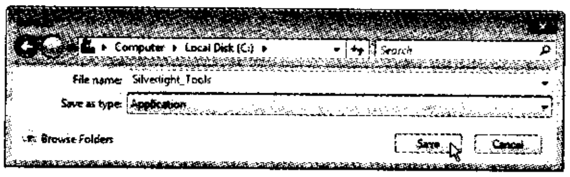
Do you want to run or save this file?

Name: Silverlight_Tools.exe
Type: Application, 72.7MB
From: download.microsoft.com

While files from the internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not run or save this software. What's the risk?

Figure 26.4: Prompting to Download and Install the Silverlight 2.0 Tools

As soon as you click the `Save` button in the `File Download` dialog box, the `Save As` dialog box appears, as shown in Figure 26.5:

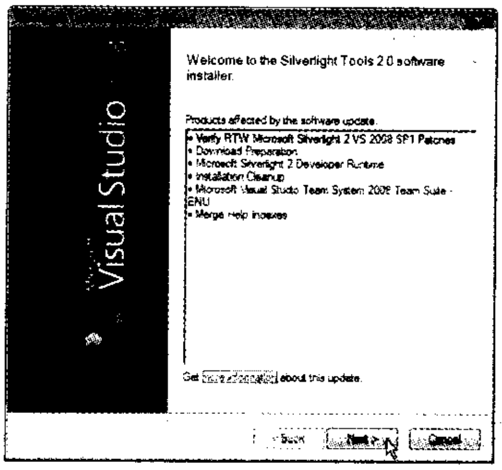


File name: Silverlight_Tools
Save as type: Application

Figure 26.5: Saving the Silverlight 2.0 Tools

You can specify the name and desired location where you want to save the Silverlight 2.0 Tools add-in in the `File name` and `Save as type` text box. In this case, you can see the add-in is saved in the C drive as Silverlight2.

5. Now, click the `Save` button in the `Save As` dialog box. The add-in (plug-in and the SDK) starts to download, the size of the add-in is around 72.7 MB.
6. After the add-in is downloaded on the computer, go to its location and double-click it. The files are extracted and the Silverlight 2.0 Tools add-in runs by displaying the Silverlight Tools Installation Wizard, as shown in Figure 26.6:



Welcome to the Silverlight Tools 2.0 software installer.

Products affected by the software update.

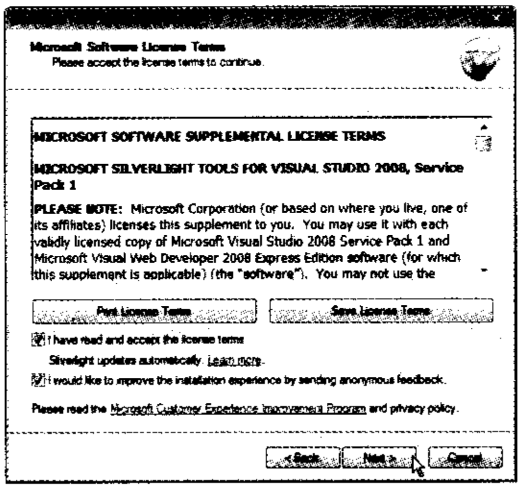
- Verify RTW Microsoft Silverlight 2 VS 2008 SP1 Patches
- Download Preparation
- Microsoft Silverlight 2 Developer Runtime
- Installation Cleanup
- Microsoft Visual Studio Team System 2008 Team Suite - ENU
- Merge Help Indexes

Get more information about this update.

Figure 26.6: Displaying the Silverlight Tools 2.0 Installation Wizard

On the first page of Silverlight Tools Installation Wizard, you can see a list of the products that are affected when the Silverlight 2.0. Tools add-in is installed on the computer.

7. Now, click the Next button to view the next page (Microsoft Software License Terms) of the Silverlight Tools Installation Wizard, as shown in Figure 26.7:


Microsoft Software License Terms
Please accept the license terms to continue.

MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS

MICROSOFT SILVERLIGHT TOOLS FOR VISUAL STUDIO 2008, Service Pack 1

PLEASE NOTE: Microsoft Corporation (or based on where you live, one of its affiliates) licenses this supplement to you. You may use it with each validly licensed copy of Microsoft Visual Studio 2008 Service Pack 1 and Microsoft Visual Web Developer 2008 Express Edition software (for which this supplement is applicable) (the "software"). You may not use the

I have read and accept the license terms

Silverlight updates automatically. Learn more.

I would like to improve the installation experience by sending anonymous feedback.

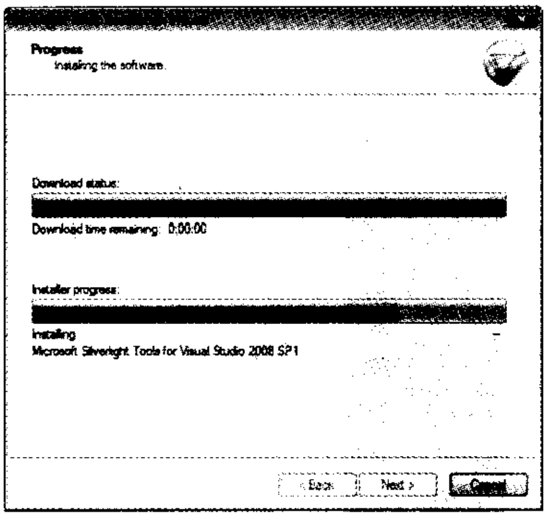
Please read the Microsoft Customer Experience Improvement Program and privacy policy.

Figure 26.7: Displaying the License Terms for Installing the Silverlight Tools

8. Read the license terms and conditions and select the 'I have read and accept the license terms.' check box.

The 'I would like to improve the installation experience by sending anonymous feedback.' check box is selected by default to allow you contribute towards the improvement of add-in installation.

9. Now, click the `Next` button, which displays the `Progress` page of the Silverlight Tools Installation Wizard. Figure 26.8 shows that the installation of the Silverlight Tools is in progress:


Progress
Installing the software.

Download status:

Download time remaining: 0:00:00

Installer progress:

Installing
Microsoft Silverlight Tools for Visual Studio 2008 SP1

Figure 26.8: Installing the Silverlight Tools 2.0

NOTE

If the system requirements of your computer do not meet the minimum requirements for Silverlight 2.0 or if there is any incompatible process or service running (for example, an instance of Visual Studio 2008 or Web browser (Internet Explorer) is open), then the system requirement page is displayed after the Microsoft Software Licence Terms page. In such a case, you need to resolve the incompatibilities and then click the Refresh button on the System Requirements page to proceed with the installation.

After the add-in is installed, the final page of the Silverlight Tools Installation Wizard appears, as shown in Figure 26.9:

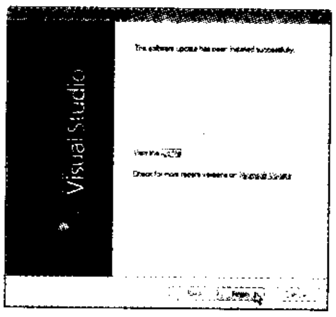

Visual Studio
The software update has been installed successfully.
Check for more recent versions on Microsoft Update
Finish

Figure 26.9: Notifying the Successful Installation of the Silverlight Tools 2.0

10. Click the Finish button to close the Silverlight Tools Installation Wizard.

Finally, both the Silverlight 2.0 plug-in (runtime) and the SDK are installed on the computer. You can now use Silverlight 2.0 to develop rich internet applications.

NOTE

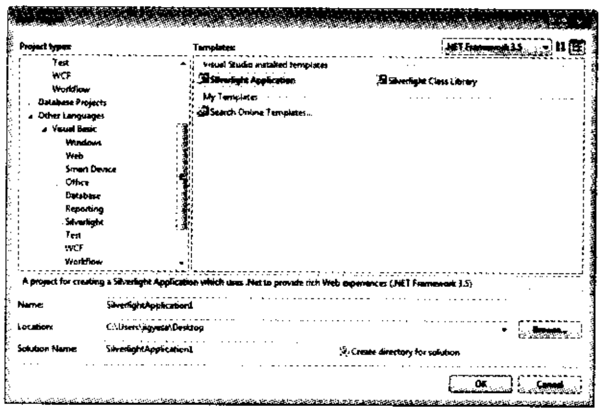
To install these Silverlight Tools the Visual Studio 2008 SP1 must be installed on your computer.

Now, let's move ahead to learn about developing Silverlight applications with Visual Studio 2008.

Silverlight Applications in Visual Studio 2008

You can create Silverlight applications by using a text editor such as Notepad. However, various tools such as Visual Studio 2008, Expression Studio and ComponentOne Studio, provide an efficient and quick means of creating Silverlight applications as these tools include the essential files and functionality by default. Among these tools, Visual Studio 2008 is an ideal choice for both designers and developers of Silverlight based applications. Note that Silverlight facilitates designers and developers to effortlessly coordinate with each other to build strikingly attractive and compelling Web applications.

Now, open Visual Studio and select `File→New→Project` from the menu bar. The `New Project` dialog box will appear. Select `Silverlight` from `Project types` list, you will find that two new project templates available, as shown in Figure 26.10:



A project for creating a Silverlight Application which uses .Net to provide rich Web experiences (.NET Framework 3.5)

Name: SilverlightApplication1

Location: C:\Users\jigyasa\Desktop

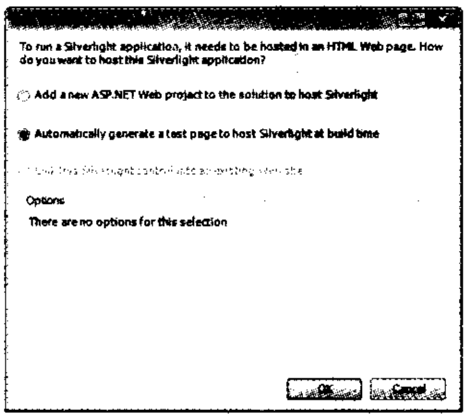
Solution Name: SilverlightApplication1

Create directory for solution

Figure 26.10: Displaying the Silverlight Project Templates in Visual Studio 2008

You can see that the Project types pane has a Silverlight option and when you click this option, it offers you the two templates, Silverlight Application and Silverlight Class Library. The Silverlight Application project template allows you to create Web applications that incorporate Silverlight, while the Silverlight Class Library project template allows you to create reusable libraries (.dll) of classes.

You can create Silverlight applications by specifying a name and location for the application in the New Project dialog box that appears when you select File→New→Project in the Visual Studio 2008 IDE. When you click the OK button on the New Project dialog box, the Add Silverlight Application dialog box appears, as shown in Figure 26.11:


To run a Silverlight application, it needs to be hosted in an HTML Web page. How do you want to host this Silverlight application?

Add a new ASP.NET Web project to the solution to host Silverlight

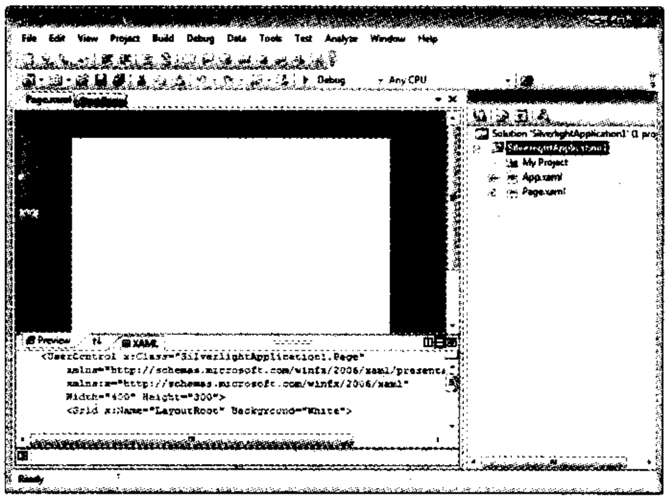
[✓ Automatically generate a test page to host Silverlight at build time]

Options

There are no options for this selection

Figure 26.11: Specifying the Host for the Silverlight 2.0 Application

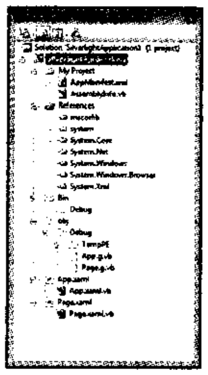
In the Add Silverlight Application dialog box, you need to specify the host for the Silverlight application. Note that Silverlight applications are hosted in HTML Web pages. You can select the `Add a new ASP.NET Web project to the solution to host Silverlight` option to automatically add and host the new Silverlight application in a Web application. With this option, most of the files and references are included automatically in the solution. However, if you want to include all the files and references on your own, you can select the `Automatically generate a test page to host Silverlight at build time` option. With this option, an HTML page is automatically added to the solution for hosting the Silverlight application. In our case, select the `Automatically generate a test page to host Silverlight at build time` option and click the OK button. A new Silverlight 2.0 application, along with a simple HTML page, is created, as shown in Figure 26.12:



```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
```

Figure 26.12: Silverlight 2.0 Application that is Hosted in a HTML Page

You can create Silverlight applications by using HTML, JavaScript, and XAML. While using Visual Studio 2008 to create Silverlight 2.0 applications, some of these files are automatically added to the Silverlight application project. You can view the references to some of those files in the Solution Explorer, as shown in Figure 26.13:



```
Solution 'SilverlightApplication1' (1 project)
  [illegible]
    My Project
      AppManifest.xml
      AssemblyInfo.vb
    References
      mscorlib
      system
      System.Core
      System.Net
      System.Windows
      System.Windows.Browser
      System.Xml
    Bin
      Debug
    obj
      Debug
        TempPE
          App.g.vb
          Page.g.vb
    App.xaml
      App.xaml.vb
    Page.xaml
      Page.xaml.vb
```

Figure 26.13: Showing the Files that Make Up a HTML Page-hosted Silverlight 2.0 Application

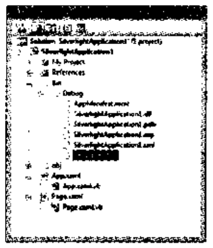
As shown in Figure 26.13, there are several files in the Silverlight application. Broadly, a Silverlight 2.0 application has three essential files:

- An HTML file (a test page) that hosts the Silverlight application (This file is available only after the compilation of the Silverlight application)
- A JavaScript file that provides methods to create and initialize the Silverlight plug-in (This file comes with the Silverlight SDK and is not visible in Solution Explorer)
- An XAML file that defines the UI of the Silverlight application

Depending on the host application that you select, there may be other files that are automatically added to the project. Normally, you use the predefined project templates in Visual Studio 2008. Now, let's discuss each of the files in detail beginning with the HTML file that hosts the Silverlight applications.

The HTML Host Page (Test Page)

As you know, Silverlight applications are hosted in Web pages. One Silverlight application can take up the entire HTML page or just a portion of the page. Moreover, there can be one or more Silverlight applications in a single HTML page. While creating a Silverlight application using Visual Studio 2008, you can opt to host the application in a simple HTML Web page or a full-fledged Web application. In case, the Silverlight application is inside a single HTML page, then when you compile the application, a test page, TestPage.html, is automatically created in the Bin→Debug folder inside the Silverlight application project folder. This test page is the Web page that is the host for the Silverlight application. Figure 26.14 shows the TestPage.html in the Solution Explorer:



```
Solution 'SilverlightApplication1' (1 project)
  SilverlightApplication1
    My Project
    References
    Bin
      Debug
        AppManifest.xaml
        SilverlightApplication1.dll
        SilverlightApplication1.pdb
        SilverlightApplication1.xap
        SilverlightApplication1.xml
        [illegible]
    obj
    App.xaml
      App.xaml.vb
    Page.xaml
      Page.xaml.vb
```

Figure 26.14: Showing the TestPage.html File in the Solution Explorer

If you cannot view the TestPage.html in the Solution Explorer, click the Show All Files button in the Solution Explorer.

NOTE

You can also host a Silverlight 2.0 application in an ASP.NET Web application. In such a case, the host Web page can be an .aspx file. You will learn more about hosting Silverlight 2.0 applications in ASP.NET application later in the chapter.

The JavaScript Helper File

You have learned earlier that a Silverlight application runs inside a plug-in that resides on the computer as a part of the Web browser. The Silverlight plug-in provides the visual region where content of the Silverlight application appears. Therefore, before creating a Silverlight application, you need to include the Silverlight plug-in within the HTML page. Although, you have already downloaded and installed the Silverlight plug-in on your computer, it needs to be created and initialized in the Web page. Unlike the previous versions of Silverlight, you do not need to explicitly initialize the plug-in in Silverlight 2.0; the initialization is done automatically. Initialization of the Silverlight plug-in is discussed in detail later in this chapter.

The Silverlight plug-in is generally held by a block-level HTML element such as the <div> element. The plug-in is then initialized by methods defined in two JavaScript helper files – CreateSilverlight.js and Silverlight.js. These JavaScript files provide programmatic access and logic for the UI elements and objects. The CreateSilverlight.js file has the `createSilverlight()` method that is called in the HTML host page (discussed earlier). The `createSilverlight()` method contains calls to the `createObject()` or `createObjectEx()` method that exists in the Silverlight.js file.

The Silverlight.js comes with the Silverlight SDK and is the main file that provides the functionality to initialize the Silverlight plug-in, verify the existence of the Silverlight plug-in on the users' computers, and check the version of the Silverlight plug-in installed on the users' computers. This JavaScript helper file defines a new namespace `Sys.Silverlight` that contains the `createObject()` and `createObjectEx()` methods. The `createObject()` and `createObjectEx()` methods create and initialize the Silverlight plug-in in the HTML page that hosts the Silverlight application. Both the methods have the same functionality and have several parameters, the values of which are fetched from the <object> or <embed> tag. Note that the <object> and <embed> tags are also generated by these two methods. The difference between the `createObject()` and the `createObjectEx()` method is that only the createObjectEx() method packs all the parameters in a JavaScript Object Notation (JSON) dictionary.

NOTE

You can find the Silverlight.js file at the location C:\Program Files\Microsoft SDKs\Silverlight\v2.0\Tools. You can also add your own methods in the file. However, it is recommended that you modify the existing contents of this file, that is, avoid changing the definitions of the createObject() and createObjectEx() methods.

The XAML Files

The design and UI of Silverlight application are defined through XAML Every Silverlight application has a few XAML files, such as `App.xaml`, `App.xaml.vb`, `Page.xaml`, and or `Page.xaml.vb`, which define the UI and basic behavior of a Silverlight application. The App.xaml and the App.xaml.vb files represent the application and the Page.xaml and Page.xaml.vb files represent the UI of the application. The App.xaml file is an XAML file that contains application-level information and settings:

Here is the default content of the `App.xaml` file:

```
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SilverlightApplication1.App">
    <Application.Resources>
    </Application.Resources>
</Application>
```

In the preceeding code, you can see that there is only one element named Application. The Application element has the xmlns, xmlns:x, and x:Class attributes. The xmlns attribute refers to the default namespace for working with the Silverlight 2.0 applications. The default namespace for Silverlight 2.0 is `http://schemas.microsoft.com/winfx/2006/xaml/presentation`, which provides the facility to work with Silverlight controls and types. The xmlns:x attribute refers to the namespace that provide the types and classes for XAML as a whole. Note that whenever you use an element that is not defined in the default Silverlight namespace but in the XAML namespace, you need to use `x:` as a prefix for the element. The x:Class attribute binds the Application element in the App.xaml file to the partial class (in the App.xaml.vb file) that inherits from the Application class. Note that the x:Class attribute is bound to the App partial class in the SilverlightApplication1 namespace.

You can access these application-level settings in the App.xaml.vb file, which is the code-behind file for App.xaml. You can also specify the event handlers for various application-level events, as follows:

The App class has some properties that you can use to configure the application. Table 26.3 lists the noteworthy properties of the App class:

Table 26.3: Noteworthy Properties of the App Class

[illegible]	[illegible]
`Host`	Allows you to retrieve information about the Web application that hosts the Silverlight application.
`Resources`	Retrieves a collection of application-level resources, such as styles and templates.
`RootVisual`	Allows you to set or retrieve the main UI object for the application. By default, it is a Page object.

Now, let's discuss the `Page.xaml` and `Page.xaml.vb` files that correspond to the UI of the Silverlight application. The Page.xaml file is yet another XAML file that contains several XAML elements for designing the UI of the Silverlight application. However `Page.xaml.vb` are the code-behind files, that contains code to respond user interaction with the page.

Here is the default content of the Page.xaml file :

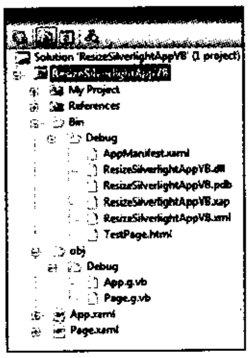
```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
    </Grid>
</UserControl>
```

In the preceeding code, you can see that the topmost element is a `UserControl` element that has a Grid element as its child. You may also notice that the `UserControl` element has five attributes—`x:Class`, `xmlns`, `xmlns:x`, `Width`, and `Height`. The `x:Class` attribute binds this element to the partial class named Page class in the `SilverlightApplication1` namespace that is present in the code-behind (`Page.xaml.vb`) file. The `xmlns` and `xmlns:x` attributes refer to the default `Silverlight` and `XAML` namespaces, respectively. The `Width` and `Height` attributes of the `UserControl` element refer to the width and height of the element.

Now, let's move on to learn about the application package (`.xap` file).

The Application Package

When you compile Silverlight 2.0 applications, they are compiled into application packages. The application packages are files with the .xap extension. A `.xap` (pronounced as *zap*) file consists of all the necessary files and resources for running a Silverlight application, as shown in Figure 26.15:



```
Solution 'ResizeSilverlightAppVB' (1 project)
  ResizeSilverlightAppVB
    My Project
    References
    Bin
      Debug
        AppManifest.xaml
        ResizeSilverlightAppVB.dll
        ResizeSilverlightAppVB.pdb
        ResizeSilverlightAppVB.xap
        ResizeSilverlightAppVB.xml
        TestPage.html
    obj
      Debug
        App.g.vb
        Page.g.vb
    App.xaml
    Page.xaml
```

Figure 26.15: Showing the .xap File for a Silverlight 2.0 Application

Note that you can see the `.xap` file (Figure 26.15) only when the Silverlight application is compiled. Moreover, the file appears in the `Bin→Debug` folder inside the Silverlight application project folder. Note that in the Solution Explorer, you may not be able to view the .xap file within the Silverlight application project folder. In such a case, click the `Show All Files` button in the toolbar of the Solution Explorer.

A .xap file is the unit of deployment and it contains the application manifest and the assemblies of a Silverlight application. The application manifest, stored as the `AppManifest.xaml`, is an XAML file that has a list of the assemblies used in the Silverlight application. The application manifest contains the `Deployment` and `AssemblyPart` elements for including the list of the required assemblies as well as the application assembly. The application assembly exists in the `Bin→Debug` folder after you compile the application and has the same name as the Silverlight application, for instance, the application assembly of a Silverlight application named `MySilverlightApp1` is `MySilverlightApp1.dll`.

NOTE

You can change the extension of a .xap file to .zip and extract the containing files. Moreover, the .xap file is specified as one of the arguments in the createObject() or createObjectEx() method while initializing the Silverlight plug-in in the HTML host page, as discussed in the next topic.

Changing the Size of the Silverlight Plug-in

At the time of instantiating the Silverlight plug-in, the height and width of Silverlight plug-in are set to occupy the whole windows of the Web browser (You can see the settings of the height and width attributes of HTML DOM object that represents the Silverlight plug-in in the test page). You can change the size of the Silverlight plug-in if you want to display the Silverlight content only in a part of the browser window. You can also display the Silverlight in a full-screen mode. Perform the following steps to change the size of the Silverlight plug-in:

1. Create a Silverlight application named `ResizeSilverlightVB`. You can find the code of `ResizeSilverlightVB` application in the Code\ASP.NET\Chapter 26\`ResizeSilverlightVB` folder on the CD.
2. Open the `Page.xaml` file of the application. Listing 26.1 shows the code for the `Page.xaml` file `ResizeSilverlightVB` application:

Listing 26.1: Showing the Code for the `Page.xaml` File

```
<UserControl x:Class="ResizeSilverlightAppVB.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="200" Height="400">
  <UserControl.Resources>
      <Storyboard x:Name="storyboard1">
          <DoubleAnimation Duration="0:0:9" AutoReverse="True" To="200"
          Storyboard.TargetName="ellipse1" Storyboard.TargetProperty="Height"/>
          <DoubleAnimation Duration="0:0:9" AutoReverse="True" To="100"
          Storyboard.TargetName="ellipse1" Storyboard.TargetProperty="Width"/>
          <ColorAnimation Duration="0:0:9" AutoReverse="True" To="#33FFFF"
          Storyboard.TargetName="ellipse1"
          Storyboard.TargetProperty="(Shape.Fill).(Brush.Color)"/>
      </Storyboard>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="Black" >
      <Ellipse x:Name="ellipse1" Fill="Red" Height="50" Width="25" Margin="0,0,0,100"
      Loaded="ellipseLoaded"/>
      <Button x:Name="Clip" Content="Clip" Height="25" Width="75" Margin="0,225,100,0"
      Click="clipPlugin"/>
      <Button x:Name="FullScreen" Content="FullScreen" Height="25" Width="75"
      Margin="100,225,0,0" Click="fullScreenPlugin"/>
  </Grid>
</UserControl>
```

In Listing 26.1, the `Height` and `Width` attributes of the root `UserControl` element in the XAML file are set to 400 and 200 pixels, respectively. There are two buttons named `Clip` and `FullScreen`. When you click the `Clip` button, the size of the Silverlight plug-in is reduced while the Silverlight content is displayed in full-screen mode on clicking the `FullScreen` button.

3. Open the code-behind file and add the click event handlers for the `Clip` and `FullScreen` buttons. Listing 26.2 shows the code for the code-behind file of `Page.xaml` file of `ResizeSilverlight` application:

Listing 26.2: Showing the Code for the Code-Behind File

```
Partial Public Class Page
    Inherits UserControl
    Public Sub New()
        InitializeComponent()
    End Sub
    Public Sub ellipseLoaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
        storyboard1.Begin()
    End Sub
    Public Sub clipPlugin(ByVal sender As Object, ByVal e As RoutedEventArgs)
        Dim pluginEle As System.Windows.Browser.HtmlElement =
        System.Windows.Browser.HtmlPage.Plugin
        pluginEle.SetProperty("height", 350)
    End Sub
    Public Sub fullScreenPlugin(ByVal sender As Object, ByVal e As RoutedEventArgs)
        App.Current.Host.Content.IsFullScreen = True
    End Sub
End Class
```

In Listing 26.2, the `ellipseLoaded()` method is called when the ellipse is loaded and is used to begin the storyboard for animating the ellipse. In the `clipPlugin()` method, which is the Click event handler of the Clip button, a variable of the `System.Windows.Browser.HtmlElement` class is defined and set to the Silverlight plug-in. The `SetProperty()` method of the Silverlight plug-in is then used to clip the height and width of the plug-in to 500 and 300 pixels, respectively. You may also note that the `App.Current.Host.Content.IsFullScreen` property is set to `True` for the Click event handler of the `FullScreen` button.

4. Now, compile the application and open the Solution Explorer. Click the `Show All Files` button in the Solution Explorer, expand Bin→Debug node, and double-click the `TestPage.html` file to open it. Change the value of the background parameter of the <object> tag that represents the Silverlight plug-in, as given in the following code:

```
<param name="background" value="#CC99CC" />
```

5. Save all the changes and run the application by pressing the F5 key. The output is as shown in Figure 26.16:

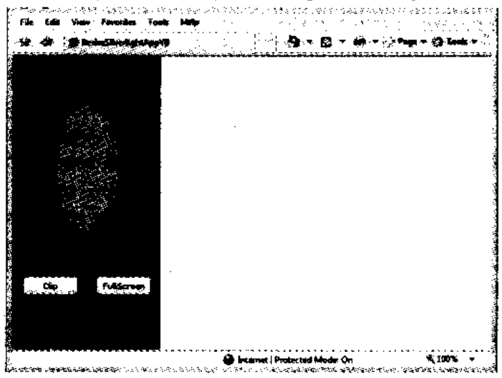


Figure 26.16: Displaying the Output of the ResizeSilverlight Application

As shown in Figure 26.16, the Silverlight plug-in, by default, occupies the entire browser window.

6. To see how the size of the Silverlight plug-in is clipped, click the Clip button (Figure 26.17):

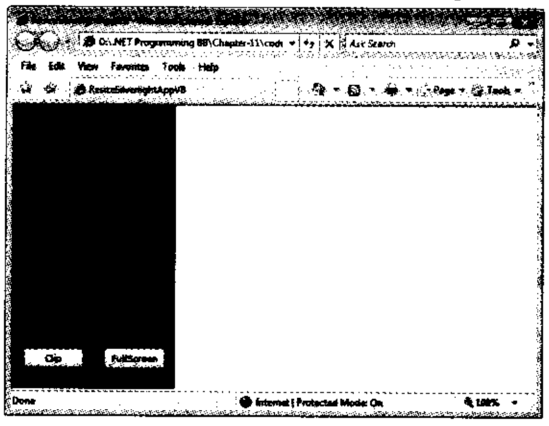

D:\.NET Programming BB\Chapter-11\code
Ask Search
File Edit View Favorites Tools Help
ResizeSilverlightAppVB
Page
Tools
Clip
FullScreen
Done
Internet | Protected Mode: On
100%

Figure 26.17: Clipping the Silverlight Plug-In

As shown in Figure 26.17, the size of the Silverlight plug-in is reduced to occupy only a particular area of the entire browser window.

You can also click the FullScreen button to view the Silverlight content in full-screen mode.

Controls in Silverlight Applications

Visual Studio 2008 provides numerous controls to build the UI of the Silverlight 2.0 applications. These controls can be added to the Silverlight applications through the XAML and code-behind files. As stated earlier, by default, a Page.xaml file is created for the UI of the Silverlight application.You can use various XAML elements in this file to design the application. You can nest the elements within one another and set different properties of these elements to get the desired appearance.

The controls in Silverlight correspond to different XAML elements. The controls or elements that you add to the Silverlight application become a part of a hierarchical tree that is then rendered to display it to the clients. You may recall that in Silverlight applications, a UserControl XAML element is the root element of the hierarchical tree. By default, the UserControl element has a Grid element as its child. You need to place the controls inside the Grid element such that those controls become the child elements of the Grid element. As and when you keep adding controls to the Silverlight applications, the hierarchical tree structure keeps growing.

NOTE

Grid is a container control just as Canvas and StackPanel. This control or element assists you in easy and convenient arrangement of the controls encompassed within them.

Silverlight 2.0 has the conventional controls, such as Button, TextBox, RadioButton, and ScrollBar and shape controls or elements, such as Ellipse, Rectangle, and Line. However, Silverlight 2.0 also offers some unique controls, such as Calendar, DatePicker, and TabControl. These new controls assist you in giving a dynamic, interactive, and engrossing UI for your Silverlight-based Web applications. In Silverlight 2.0, most of the basic controls are offered by the Silverlight runtime (plug-in); however, few of the controls that provide advanced functionality are offered by the Silverlight SDK and hence require you to reference the appropriate assemblies and namespaces. Table 26.4 lists the controls that are available in Silverlight 2.0 (plug-in as well as SDK):

Table 26.4: Controls in Silverlight 2.0

[illegible]	[illegible]	[illegible]
`Border`	Allows you to add a border and a background to some other control.	<Border BorderBrush="Blue" BorderThickness="3"/>
`Button`	Allows you to provide a button that	<Button Content="Hello"

Table 26.4: Controls in Silverlight 2.0

	responds to mouse and keyboard events.	ClickMode="Press"/>
`Calendar`	Allows you to view a calendar and work with days, months, and years. The Calendar control is available in `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.	<sdkcon:Calendar DisplayMode="Month" IsTodayHighlighted="True"/> Here, sdkcon is the prefix for `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.
`Canvas`	Allows you to position and arrange the containing controls at given locations.	<Canvas> <Button Content="Hello" Canvas.Left="50" Canvas.Top="50"/> </Canvas>
`CheckBox`	Allows you to provide check boxes that facilitate the selection or clearing of options. Check boxes can be checked, unchecked, or indeterminate state at a time.	<CheckBox ClickMode="Press" IsChecked="False" Content="Apple"/>
`ContentControl`	Refers to a control that contains only one piece of content of any type. Several controls are derived from the ContentControl.	<ContentControl Content="Silverlight"/>
`DataGrid`	Allows you to display data in the form of a grid with several rows and columns. This control is available in `System.Windows.Controls`.Data.dll in the `System.Windows.Controls` namespace.	<sdkdata:DataGrid ItemsSource="Silverlight" AutoGenerateColumns="true" ColumnHeaderHeight="30" AlternatingRowBackground="Orchid" /> Here, sdkdata is the prefix for the `System.Windows.Controls`.Data assembly in the `System.Windows.Controls` namespace.
`DatePicker`	Allows you to select a particular date from a given calendar or type the desired date. This control is also available in `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.	<sdkcon:DatePicker IsTodayHighlighted="True" SelectedDateFormat="Short"/> Here, sdkcon is the prefix for `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.
`Ellipse`	Allows you to provide an elliptical shape.	<Ellipse Fill="Red" Height="50" Width="100"/>
`Grid`	Allows you to arrange the containing controls in rows and columns.	<Grid Background="Yellow" ShowGridLines="True"> <Grid.RowDefinitions> <RowDefinition Height="50*"/> <RowDefinition Height="50*"/> </Grid.RowDefinitions> <Button Content="Silverlight" Grid.Row="0"/> </Grid>
`GridSplitter`	Allows you to provide a UI element that the users can move to adjust the space between the rows and columns of a grid. This control is also available in `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.	<sdkcon:GridSplitter Grid.Row="1" ShowsPreview="True" Height="10" Width="300"/> Here, sdkcon is the prefix for `System.Windows.Controls.dll`in the `System.Windows.Controls` namespace.

Table 26.4: Controls in Silverlight 2.0

[illegible]	[illegible]	[illegible]
`HyperlinkButton`	Allows you to display a button with hyperlink.	
`Image`	Allows you to display a .png or .jpg image.	<Image Height="300" Width="300" Source="Garden.jpg"/>
`ItemsControl`	Allows you to display a collection of items. Many controls that allow you to display a collection of items inherit the properties of this control.	<ItemsControl ItemsSource="Silverlight" Height="270" Margin="100,0,0,0"/>
`Line`	Allows you to display a line.	<Line Stroke="Red" StrokeThickness="5" X1="100" Y1="100" X2="300" Y2="100"/>
`ListBox`	Allows you to display a list of items.	<ListBox> <ListBoxItem Content="Apple" Background="Yellow"/> <ListBoxItem Content="Mango" Background="Orange"/> <ListBoxItem Content="Strawberry" Background="Plum"/> </ListBox>
`MediaElement`	Allows you to host audio and video files.	<MediaElement Height="100" Width="100" Source="Butterfly.wmv">
`Popup`	Allows you to temporarily display information on top of some other control.	<Popup IsOpen="True" HorizontalOffset="20" VerticalOffset="50" > <Popup.Child> <Button x:Name="button1" Content="hello" Height="30" Width="100"/> </Popup.Child> </Popup>
`RadioButton`	Allows you to display a radio button to represent an option.	<RadioButton ClickMode="Press" Content="Apple" Margin="10,10,0,0"/> <RadioButton ClickMode="Press" Content="Mango" Margin="10,50,0,0"/> <RadioButton ClickMode="Press" Content="Strawberry" Margin="10,90,0,0"/>
`Rectangle`	Allows you to display a rectangle.	<Rectangle Fill="Violet" Height="50" Width="100"/>
`RepeatButton`	Allows you to provide a button that periodically raises its Click event until the mouse is released.	<RepeatButton Content="Hello" Delay="30" Interval="90" Height="30"/>
`ScrollBar`	Allows you to provide a scroll bar with a Thumb that you can slide to change the value.	<ScrollBar Minimum="0" Maximum="10" Value="5" ViewportSize="1" Height="30" Width="200"/>
`ScrollViewer`	Allows you to view some content that can be scrolled using the horizontal and/or vertical scroll bars.	<ScrollViewer Content="Silverlight" Height="100" Width="100" HorizontalScrollBarVisibility="Visible" VerticalScrollBarVisibility="Visible"/>
`Slider`	Allows you to provide a slider that has a Thumb, which you can move to select a particular value from the given range of	<Slider Orientation="Horizontal" Value="5" Maximum="100" Minimum="0"/>

Table 26.4: Controls in Silverlight 2.0

	values. This control is available in System.Windows.dll of the `System.Windows.Controls` namespace.	
`StackPanel`	Allows you to arrange the containing controls in a horizontal or vertical stack.	<StackPanel> <Button Content="Apple"/> <Button Content="Mango"/> <Button Content="Strawberry"/> </StackPanel>
`TabControl`	Allows you to display information with the help of tabs. Each tab can hold various other controls. This control is available in `System.Windows.Controls.dll` in the `System.Windows.Controls` namespace.	<sdkcon:TabControl Height="200" Width="250"> <sdkcon:TabItem Header="General" Content="This tab has general information"/> <sdkcon:TabItem Header="Academics"/> <sdkcon:TabItem Header="Professional"/> </sdkcon:TabControl> Here, sdkcon is the prefix for `System.Windows.Controls.dll`in the `System.Windows.Controls` namespace
`TextBlock`	Allows you to display text.	<TextBlock Text="Hello"/>
`TextBox`	Allows you to display text or retrieve text from user. In Silverlight 2.0, the TextBox control can display text in multiple lines and wrap text around the control.	<TextBox Text="Hello, this is the TextBox control in Silverlight 2 " IsReadOnly="False" Height="75" Width="75" TextWrapping="Wrap"/>
`ToggleButton`	Allows you to provide a button that can toggle between clicked and non-clicked states.	<ToggleButton Content="Hello" ClickMode="Press" IsThreeState="True" Height="30"/>

Silverlight 2.0 includes some new controls, such as the StackPanel and the MultiScaleImage controls that allow you to work with the layout of the Silverlight content and view and change the size and position of a multi-resolution image, respectively. These controls have its prime use with the Deep Zoom technology. Silverlight 2.0 also incorporates a new property named `ToolTipService` for the controls. This property replaces the ToolTip control in Silverlight 2.0 Beta 1.

NOTE

The Silverlight 2.0 controls indirectly inherit the FrameworkElement class, which itself derives from the UIElement class. Therefore, the properties, methods, and events defined by the FrameworkElement class and its base class are available to the controls.

Using Silverlight Controls

Silverlight offers numerous controls that you can use to define the look and feel of the applications. Most of the controls are the same as WPF; and therefore, provide the same functionality as provided by controls such as, `Button`, `TextBox`, `Rectangle`, and `ListBox`. However, Silverlight 2.0 also offers some exclusive controls to provide additional functionality for Web applications. Now, let's learn how to work some of the controls in Silverlight 2.0:

- The DatePicker control
- The GridSplitter control
- The MediaElement control

Using the DatePicker Control

The `DatePicker` control has a text box and drop-down calendar that allows you to type or select a date respectively. Perform the following steps to see how you can work with dates using the `DatePicker` control in Silverlight 2.0:

1. Create a Silverlight application named `SubjectScheduleVB`. This application is hosted in a single HTML Web page. You can find the code of `SubjectScheduleVB` application in the Code\ASP.NET\Chapter 26\`SubjectScheduleVB` folder on the CD. Note that since the `DatePicker` control is available in the `System.Windows.Controls`, you need to add a reference to the assembly and an xmlns attribute in the root `UserControl` element in the `Page.xaml` file. The UI of the `SubjectSchedule` application is defined in the `Page.xaml` file. Listing 26.3 shows the code for the `Page.xaml` file of `SubjectScheduleVB` application:

Listing 26.3: Showing the Code for the `Page.xaml` File

```
<UserControl x:Class="SubjectScheduleAppVB.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:sdkcon="clr-
   namespace:System.Windows.Controls;assembly=System.Windows.Controls" Width="600"
   Height="450">
  <UserControl.Resources>
      <Style TargetType="TextBlock" x:Name="textStyle">
          <Setter Property="FontSize" Value="18"/>
          <Setter Property="Foreground" Value="Maroon"/>
      </Style>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="White">
      <Rectangle x:Name="back" Height="450" Width="600">
          <Rectangle.Fill>
              <LinearGradientBrush SpreadMethod="Reflect">
                  <LinearGradientBrush.GradientStops>
                      <GradientStop Offset="0.25" Color="Beige"/>
                      <GradientStop Offset="0.75" Color="Burlywood"/>
                  </LinearGradientBrush.GradientStops>
              </LinearGradientBrush>
          </Rectangle.Fill>
      </Rectangle>
      <TextBlock x:Name="text1" FontSize="30" Text="Make Your Own Schedule"
      Margin="125,10,0,0" Foreground="Maroon" />
      <ListBox x:Name="list1" Margin="20,75,20,100">
          <ListBoxItem Background="HotPink">
              <Grid>
                  <TextBlock Text="Subjects"  FontSize="24"
                  Foreground="LightYellow" Margin="10,0,0,0"></TextBlock>
                  <TextBlock Text="Select Date" FontSize="24"
                  Foreground="LightYellow" Margin="250,0,0,0"></TextBlock>
              </Grid>
          </ListBoxItem>
          <ListBoxItem x:Name="bioItem" Background="LightBlue">
              <StackPanel>
              <TextBlock x:Name="bioText" Text="Biology"
              Style="{StaticResource textStyle}" Margin="10,0,0,0"/>
              <sdkcon:DatePicker x:Name="bioDate"  Height="30"  Width="200"
              Margin="200,-20,0,0"  IsDropDownOpen="False"
              SelectedDateFormat="Long"/>
              </StackPanel>
          </ListBoxItem>
          <ListBoxItem x:Name="chemItem"  Background="LightBlue">
          <StackPanel>
```

```
                <TextBlock x:Name="chemText" Text="Chemistry"
                Style="{StaticResource textStyle}" Margin="10,0,0,0"/>
                <sdkcon:DatePicker x:Name="chemDate" Height="30" Width="200"
                Margin="200,-20,0,0" IsDropDownOpen="False"
                SelectedDateFormat="Long"/>
            </StackPanel>
        </ListBoxItem>
        <ListBoxItem x:Name="engItem"  Background="LightBlue">
            <StackPanel>
                <TextBlock x:Name="engText" Text="English"
                Style="{StaticResource textStyle}" Margin="10,0,0,0"/>
                <sdkcon:DatePicker x:Name="engDate" Height="30" Width="200"
                Margin="200,-20,0,0" IsDropDownOpen="False"
                SelectedDateFormat="Long"/>
            </StackPanel>
        </ListBoxItem>
        <ListBoxItem x:Name="finItem"  Background="LightBlue">
            <StackPanel>
                <TextBlock x:Name="finText" Text="Finance"
                Style="{StaticResource textStyle}" Margin="10,0,0,0"/>
                <sdkcon:DatePicker x:Name="finDate" Height="30" Width="200"
                Margin="200,-20,0,0" IsDropDownOpen="False"
                SelectedDateFormat="Long"/>
            </StackPanel>
        </ListBoxItem>
        <ListBoxItem x:Name="mathItem"  Background="LightBlue">
            <StackPanel>
                <TextBlock x:Name="mathText" Text="Mathematics"
                Style="{StaticResource textStyle}" Margin="10,0,0,0"/>
                <sdkcon:DatePicker x:Name="mathDate" Height="30" Width="200"
                Margin="200,-20,0,0" IsDropDownOpen="False"
                SelectedDateFormat="Long"/>
            </StackPanel>
        </ListBoxItem>
        <ListBoxItem x:Name="phyItem"  Background="LightBlue">
            <StackPanel>
                <TextBlock x:Name="phyText" Text="Physics" Style="{StaticResource
                textStyle}" Margin="10,0,0,0"/>
                <sdkcon:DatePicker x:Name="phyDate" Height="30" Width="200"
                Margin="200,-20,0,0" IsDropDownOpen="False"
                SelectedDateFormat="Long"/>
            </StackPanel>
        </ListBoxItem>
        </ListBox>
    </Grid>
</UserControl>
```

In Listing 26.3, you can see that an `xmlns` attribute in the root `UserControl` element is added for `System.Windows.Controls.dll` assembly in order to work with the `DatePicker` control. Note that there are six `DatePicker` controls in the application.

2. Now, open the code-behind file of `SubjectScheduleVB` application and write the code as shown in Listing 26.4:

Listing 26.4: Showing the Code for the Code-Behind File

```
Partial Public Class Page
    Inherits UserControl
    Public Sub New()
        InitializeComponent()
        Dim firstDay As New DateTime(2008, 8, 1)
        Dim endDay As DateTime = firstDay.AddDays(15)
        'specifying the first visible and selectable date
        bioDate.DisplayDateStart = firstDay
```

```
            chemDate.DisplayDateStart = firstDay
            engDate.DisplayDateStart = firstDay
            finDate.DisplayDateStart = firstDay
            mathDate.DisplayDateStart = firstDay
            phyDate.DisplayDateStart = firstDay
            'Specifying the last visible and selectable date
            bioDate.DisplayDateEnd = endDay
            chemDate.DisplayDateEnd = endDay
            engDate.DisplayDateEnd = endDay
            finDate.DisplayDateEnd = endDay
            mathDate.DisplayDateEnd = endDay
            phyDate.DisplayDateEnd = endDay
        End Sub
    End Class
```

In Listing 26.4, the `firstDay` variable denotes the date 1st August 2008 and the `endDay` variable represents the date 15 days after 1st August 2008, that is, 16th August 2008. The `firstDay` variable is assigned the first date that is visible and selectable for the `bioDate`, `chemDate`, `engDate`, `finDate`, `mathDate`, and `phyDate` `DatePicker` controls. Similarly, the `endDay` variable is assigned the last visible and selectable date for the `DatePicker` controls by using their `DisplayDateEnd` property.

3. Now, run the application to see the output, as shown in Figure 26.18:

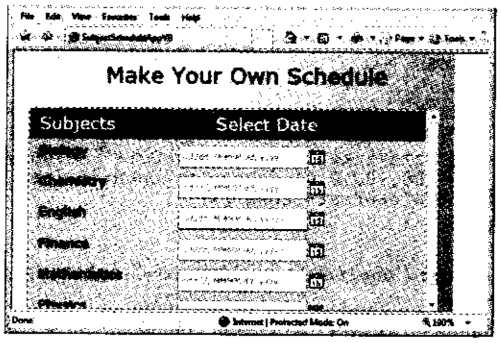


Figure 26.18: Output of the SubjectScheduleVB Application

As shown in Figure 26.18, there are six `DatePicker` controls to select a date for each subject.

4. Click the icon at the extreme right of the `DatePicker` controls to view the drop-down calendar and select a date, as shown in Figure 26.19:

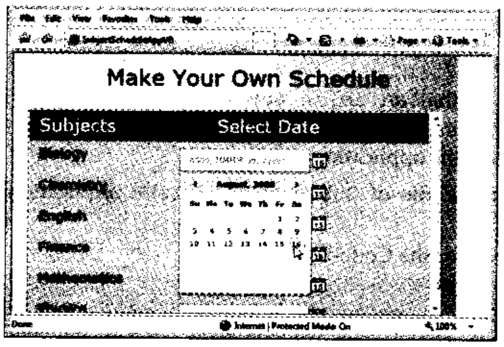
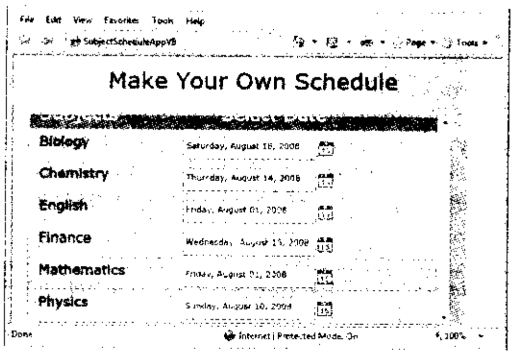


Figure 26.19: Selecting a Date from the Drop-down Calendar in a DatePicker Control

5. Repeat Step 4 to select the date for each subject. Figure 26.20 shows the output of the application when dates have been selected from all the `DatePicker` controls in the `SubjectScheduleVB` application:


Make Your Own Schedule

Biology Saturday, August 16, 2008

Chemistry Thursday, August 14, 2008

English Friday, August 01, 2008

Finance Wednesday, August 13, 2008

Mathematics Friday, August 01, 2008

Physics Sunday, August 10, 2008

File Edit View Favorites Tools Help
SubjectScheduleAppVB
Page Tools
Done
Internet | Protected Mode: On
100%

Figure 26.20: Selected Dates in all DatePicker Controls in the SubjectScheduleVB Application

Using the GridSplitter Control

The `GridSplitter` control allows users of your Silverlight application to resize the rows and columns of a Grid control. Perform the following steps to see how the `GridSplitter` control accomplishes it:

1. Create an HTML page-hosted Silverlight application and name it `TextImageVB`. You can find the code of `TextImageVB` application in the Code\ASP.NET\Chapter 26\`TextImageVB` folder on the CD. Add a reference to the `System.Windows.Controls` assembly and an `xmlns` attribute for the assembly in the root `UserControl` element in the `Page.xaml` file. Listing 26.5 shows the code for the `Page.xaml` file of `TextImageVB` application:

Listing 26.5: Showing the Code for the `Page.xaml` File

```
<UserControl x:Class="TextImageAppVB.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:sdkcon="clr-
  namespace:System.Windows.Controls;assembly=System.Windows.Controls"
  Width="400" Height="300">
  <Grid x:Name="LayoutRoot" Background="Beige">
      <Grid.RowDefinitions>
          <RowDefinition Height="50"/>
          <RowDefinition Height="125"/>
          <RowDefinition Height="125"/>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
          <ColumnDefinition Width="150%"/>
          <ColumnDefinition Width="250"/>
      </Grid.ColumnDefinitions>

      <TextBlock Grid.Row="0" Grid.ColumnSpan="2" Text="TextImage Application"
      HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="Red"/>
      <TextBox Grid.Row="1" Margin="10,10,10,10" TextWrapping="Wrap" Text="Common
      Name:Green Sea Turtle Scientific Name:Chelonia mydas Genus:Chelonia Location:
      Atlantic,Pacific Ocean" IsReadOnly="True" Background="LightBlue"/>
      <Image Source="GreenSeaTurtle.jpg" Grid.Row="1" Grid.Column="1"
      Margin="10,10,10,10"/>

      <TextBox Grid.Row="2" Margin="10,10,10,10" TextWrapping="Wrap" Text="Winters
      Begins:Solstice  Ends: Equinox" IsReadOnly="True" Background="LightBlue"/>
      <Image Source="WinterLeaves.jpg" Grid.Row="2" Grid.Column="1"
      Margin="10,10,10,10"/>
      <sdkcon:GridSplitter x:Name="split1" Height="20" Width="250" Grid.Row="2"
      Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Stretch"
      VerticalAlignment="Top" Background="SteelBlue"/>
      <sdkcon:GridSplitter x:Name="split2" Height="250" Width="20" Grid.Row="1"
      Grid.Column="1" Grid.RowSpan="2" HorizontalAlignment="Left"
      VerticalAlignment="Stretch" Background="SteelBlue"/>
```

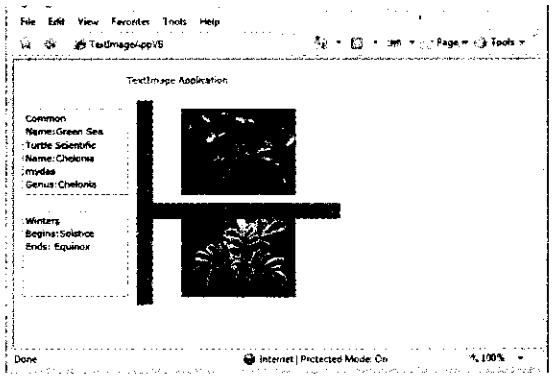
```
    </Grid>
  </UserControl>
```

In Listing 26.5, you can see that two `GridSplitter` controls are added in the third row and second row. The first `GridSplitter` control, `split1`, exists in the third row of the Grid control and spans two columns. The `HorizontalAlignment` property of `split1` is set to `Stretch`, which implies that it adjusts the size of the row by expanding or contracting its contents. The second `GridSplitter` control, `split2`, has its `VerticalAlignment` property set to `Stretch`, which allows the columns to be resized.

NOTE

The GreenSeaTurtle.jpg and WinterLeaves.jpg images are added to the current project folder by using the Add Existing Item dialog box.

2. Press the F5 key to run the application and see the output, as shown in Figure 26.21:


File Edit View Favorites Tools Help
TextImageAppVB
Page Tools
Done
Internet | Protected Mode: On
100%

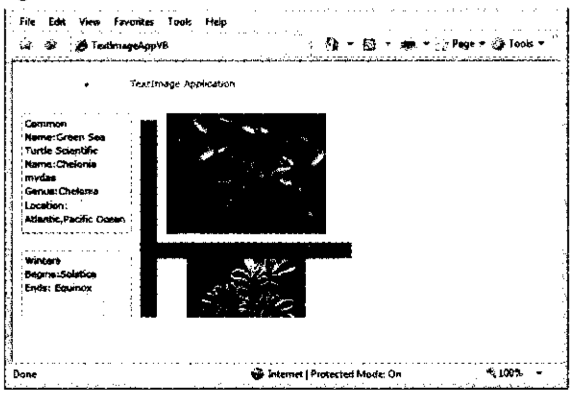
TextImage Application

Common Name:Green Sea Turtle Scientific Name:Chelonia mydas Genus:Chelonia

Winters Begins:Solstice Ends: Equinox

Figure 26.21: Output of the TextImageVB Application

3. Click and drag either of the GridSplitter controls. Notice that the rows and columns are automatically resized, as shown in Figure 26.22:


File Edit View Favorites Tools Help
TextImageAppVB
Page Tools
Done
Internet | Protected Mode: On
100%

TextImage Application

Common Name:Green Sea Turtle Scientific Name:Chelonia mydas Genus:Chelonia Location: Atlantic,Pacific Ocean

Winters Begins:Solstice Ends: Equinox

Figure 26.22: Resizing the Rows and Columns on the TextImageVB Application

Using the MediaPlayer Control to Embed Media

You have learned that the `MediaPlayer` server control can be used to embed and manage media in ASP.NET applications. You can embed both audio and video in applications. Perform the following steps to see the use of the `MediaPlayer` control to embed media:

1. Create an ASP.NET website named `PlayVideoVB` and embed media using the `MediaPlayer` server control. You can find the code of `PlayVideoVB` application in the Code\ASP.NET\Chapter 26\`PlayVideoVB` folder on the CD. For this, create a website named `PlayVideoVB` on the localhost. In the

Default.aspx page, drag a label from the Standard tab in the Toolbox and drop it on the page. Next, expand the AJAX Extensions tab in the Toolbox and drag-and-drop a ScriptManager control on to the page. Now, expand the Silverlight controls tab in the Toolbox and drag-and-drop a MediaPlayer control.

2. Add the MediaPlayer control in the Default.aspx page. You can find the code for the Default.aspx page in the Listing 26.6:

Listing 26.6: Showing the Code for the Default.aspx Page

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
  Inherits="_Default" %>
<%@ Register Assembly="System.Web.Silverlight"
  Namespace="System.Web.UI.SilverlightControls"
  TagPrefix="asp" %>
<!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 id="Head1" runat="server">
  <title>PlayVideo Application</title>
</head>
<body>
  <form id="form1" runat="server">
      <div style=" height:650px; width:900px">
      <div style=" margin-left:350px; margin-top:15px">
      <asp:Label ID="Label1" runat="server" Text="Playing a Video"
      ForeColor="#990099" Font-Bold="True" Font-Size="Larger"></asp:Label>
      </div><br />
      <div style="height:500px; width:750px; margin-left:25px; margin-right:25px">
      <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
      <div style="margin-left:25px">
          <asp:MediaPlayer ID="MediaPlayer1" runat="server" Height="500px"
          Width="700px" MediaSource="~/myVideo.wmv"></asp:MediaPlayer>
      </div></div>
      </div>
  </form>
</body>
</html>
```

In Listing 26.6, you can see the markup for the ScriptManager control and the MediaPlayer server control. The MediaPlayer control is configured to play the myVideo.wmv file, which is a custom Windows Media Video file. Note that you need to add this video file in the current project (Web site). You can do this by using the Add Existing Item dialog box that appears when you select website→Add Existing Item on the menu bar.

3. Now, run the application by pressing the F5 key. The output of the application is shown in Figure 26.23:

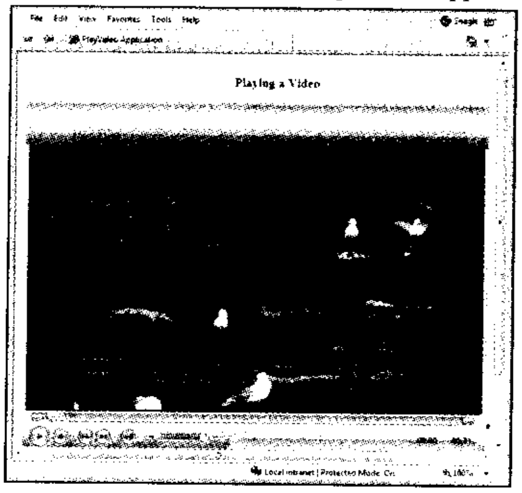


Figure 26.23: Working with the MediaPlayer Control

As shown in Figure 26.23, the media player is displayed when the Web page is loaded. The total duration of the video is also visible at the bottom right corner of the media player.

Slider moves towards the right as soon as you click the start button. You can pause, rewind, or fast-forward the playback of the video.

TIP

If you want to start playing the video (or audio) as soon as the MediaPlayer control is loaded on the Web page, then set the AutoPlay property of the MediaPlayer control to true.

NOTE

Another way of including information about the media (media source and chapters) is to use a media definition file. A media definition file is an XML file that specifies information about the media through various XML elements, such as <MediaDefinition> and <mediaItem>. You need to set the MediaDefinition property of the MediaPlayer control to the path of the media definition file.

Silverlight and ASP.NET

Silverlight can easily permeate into Web applications built with existing Web technologies, such as ASP.NET. With ASP.NET applications, Silverlight offers a consistent application model to implement the logic of the application and a rich and interactive set of controls to design the UI of the application. Blending Silverlight with ASP.NET websites not only improves their appearance by making them more dynamic but also enhances their behavior by making them more interactive.

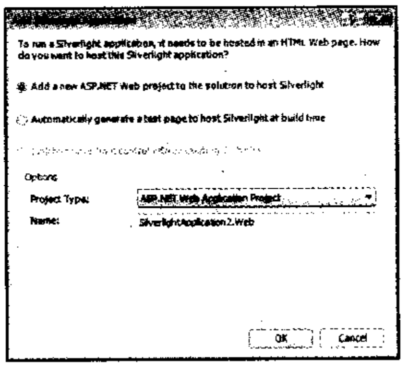
You can host Silverlight content on any ASP.NET website using certain ASP.NET server controls. Note that the Silverlight content hosted in an ASP.NET website uses XAML for the UI, while the rest of the ASP.NET content uses standard HTML for UI. You can host Silverlight content in an ASP.NET website in two ways:

- Create a Silverlight application using the Visual Studio 2008 template and use an ASP.NET website to host it.
- Create an ASP.NET website and use the Silverlight-enabling server controls to host the Silverlight content.

Now, let's create a Silverlight 2.0 application and use a website to host the Silverlight content by following these steps:

1. Open the Visual Studio 2008 window and select File→New→Project, which displays the New Project dialog box.
2. In the Project types pane, select Visual Basic→Silverlight
3. Then select Silverlight Application in the Templates pane.
4. Now, specify a name and location for the Silverlight application and click the OK button.

As soon as you click the OK button, the `Add Silverlight Application` dialog box appears, as shown in Figure 26.24:



To run a Silverlight application, it needs to be hosted in an HTML Web page. How do you want to host this Silverlight application?

[✓ Add a new ASP.NET Web project to the solution to host Silverlight]

Automatically generate a test page to host Silverlight at build time

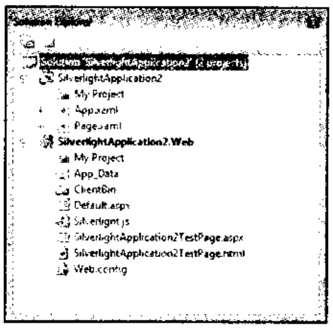
Options

Project Type: ASP.NET Web Application Project

Name: SilverlightApplication2.Web

Figure 26.24: Creating a Website to Host the Silverlight Content

5. Click the OK button. This will create a Silverlight application along with an ASP.NET website to host the Silverlight content, as you can see in the Solution Explorer of the application, as shown in Figure 26.25:



```
Solution 'SilverlightApplication2' (2 projects)
  SilverlightApplication2
    My Project
    App.xaml
    Page.xaml
  SilverlightApplication2.Web
    My Project
    App_Data
    ClientBin
    Default.aspx
    Silverlight.js
    SilverlightApplication2TestPage.aspx
    SilverlightApplication2TestPage.html
    Web.config
```

Figure 26.25: Displaying the Files of the Silverlight Application in the Solution Explorer

As shown in Figure 26.25, there are two projects created in the application, one project is for the Silverlight content itself and the other project is the ASP.NET website that hosts the Silverlight content. The Silverlight project has the `App.xaml` and `App.xaml.vb` files for working with the settings of the Silverlight project as a whole and the `Page.xaml` and `Page.xaml.vb` files for working with the UI of the Silverlight project.

The ASP.NET website project has the `Default.aspx` and code-behind file for the default Web page of the website. The `Default.aspx` file contains the HTML markup for designing the Web page, while the code-behind file contains code to respond user interactions with the Web page. The Web application also has the web.config file that is an XML file storing the configuration of the website.

This website project also has two test pages created for the entire Silverlight application. Both these test pages (one .aspx and the other .html) are used to host the Silverlight application. The test pages have the same name as the Silverlight application suffixed with TestPage (Figure 26.25). Note that both the .aspx and the .html test pages are created by default. The .html test page largely contains the same code as the TestPage.html file of Silverlight applications that are hosted in a single HTML page, that is, the .htm test page has the code to create and initialize the Silverlight plug-in. The .aspx test page contains the equivalent ASP.NET code of the .html test page. This .aspx test page is the first page that appears when users run the application.

Here is the code of the SilverlightApplication2TestPage.aspx file:

```
    <%@ Page Language="" AutoEventWireup="true" %>
    <%@ Register Assembly="System.Web.Silverlight"
    Namespace="System.Web.UI.SilverlightControls" TagPrefix="asp" %>
    <!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" style="height:100%;">
    <head runat="server">
    <title>Test Page For SilverlightApplication2</title>
    </head>
    <body style="height:100%;margin:0;">
    <form id="form1" runat="server" style="height:100%;">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <div  style="height:100%;">
    <asp:Silverlight ID="Xaml1" runat="server"
    Source="~/ClientBin/SilverlightApplication2.xap" MinimumVersion="2.0.31005.0"
    Width="100%" Height="100%" />
    </div>
    </form>
</body>
</html>
```

In the preceeding code, you can see that there are two ASP.NET server controls, ScriptManager and Silverlight, used in the .aspx file. The ScriptManager is an essential ASP.NET AJAX control, while the Silverlight is a control available to integrate Silverlight with an ASP.NET website. You can see that the source attribute of the Silverlight control refers to a .xap file, which is the application package that is created when you compiled the application. You will learn more about the Silverlight control later in this chapter.

As stated earlier, you can also embed Silverlight content in existing ASP.NET websites without much effort. You just need to use the Silverlight server control to embed the Silverlight content in a Web page.

Now, let's learn about ASP.NET Server Controls for Silverlight.

ASP.NET Server Controls for Silverlight

In ASP.NET 3.5, there are two server controls, Silverlight and `MediaPlayer`, which allow you to work with the Silverlight content. These two server controls assist you in building attractive, dynamic, and interactive websites that is not possible through ASP.NET AJAX, HTML, CSS, or JavaScript. You can see these two server controls in the Silverlight controls tab in the Toolbox, as shown in Figure 26.26:

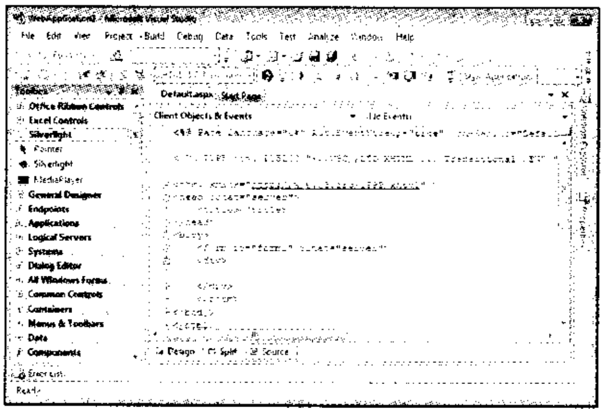


Figure 26.26: Displaying the ASP.NET Server Controls in the Toolbox

Now, let's discuss both the server controls in detail starting with the Silverlight server control in ASP.NET.

Silverlight Server Control

The Silverlight control in ASP.NET 3.5 allows you to add XAML content in the ASP.NET applications. You can also embed managed assemblies, application packages, and client-side JavaScript libraries along with the XAML content. When you create a Silverlight application that is hosted in a website by using the Silverlight Application template in Visual Studio 2008, you can see that the Silverlight control is automatically added to the .aspx test page, as shown in Figure 26.27:

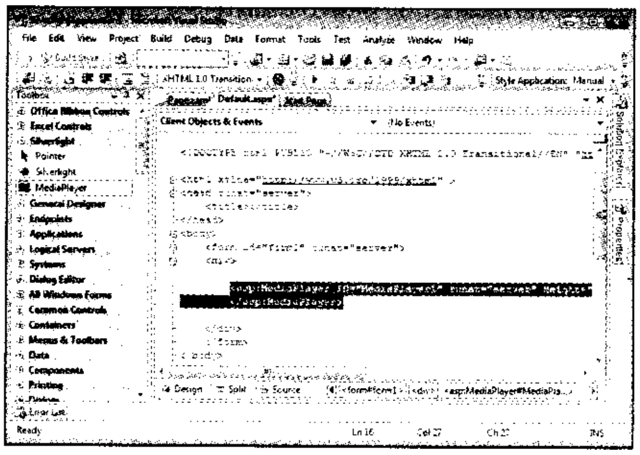


Figure 26.27: The .aspx Test Page Showing the Markup for the Silverlight Control

NOTE

In order to use the Silverlight control in your ASP.NET Web page, you need to first include the ScriptManager control, which is an ASP.NET AJAX control.

The Silverlight control basically offers a medium to work with the settings of the Silverlight plug-in in an ASP.NET website. The control references an XAML file or application package for its presentation. For the logic or functionality, you need to use the Silverlight class (in the System.Web.UI.SilverlightControls namespace). The Silverlight class represents an instance of the Silverlight control and provides the functionality to allow you to work with the Silverlight content in the Web application.

NOTE

The Silverlight class inherits the System.Web.UI.Controls.WebControl class and implements the System.Web.UI.IScriptControl interface.

The inheritance hierarchy of the Silverlight class is:

```
System.Object
  System.Web.UI.Control
      System.Web.UI.WebControls.WebControl
          System.Web.UI.SilverlightControls.Silverlight
```

Table 26.5 lists some of the noteworthy properties of the Silverlight class:

Table 26.5: Noteworthy Properties of the Silverlight class

Property	Description
`AutoUpgrade`	Sets or retrieves a value that indicates whether the Silverlight plug-in is automatically upgraded
`EnableFrameRateCounter`	Sets or retrieves a value that indicates whether to show the current frame rate in the status bar of the Web browser on the client-side
`EnableRedrawRegions`	Sets or retrieves a value that indicates whether to show the portions of the Silverlight plug-in that are redrawn for every frame
`HtmlAccess`	Sets or retrieves a value that indicates whether the Silverlight content has access to the objects of the HTML DOM
`InitParameters`	Sets or retrieves a list of the optional custom parameters at the time of initializing the Silverlight plug-in
`MaxFrameRate`	Sets or retrieves the maximum frame rate for rendering the Silverlight content
`MinimumVersion`	Sets or retrieves the minimum `client/host` version of the Silverlight plug-in
`PluginBackground`	Sets or retrieves the background color of the Silverlight plug-in
`PluginNotInstalledTemplate`	Sets or retrieves an object containing the HTML content that is to be displayed to the users in case the required Silverlight plug-in is not installed
`ScaleMode`	Sets or retrieves a value that indicates how the Silverlight plug-in scales itself
`ScriptType`	Sets or retrieves the type of the JavaScript object that is used to create and initialize the Silverlight plug-in
`Source`	Sets or retrieves the URL of the XAML file or the application package of the Silverlight content
`SplashScreenSource`	Sets or retrieves the URL of the splash screen file that appears when the XAML file of the Silverlight content is in the process of loading
`Windowless`	Sets or retrieves a value that indicates whether the Silverlight plug-in appears without the window

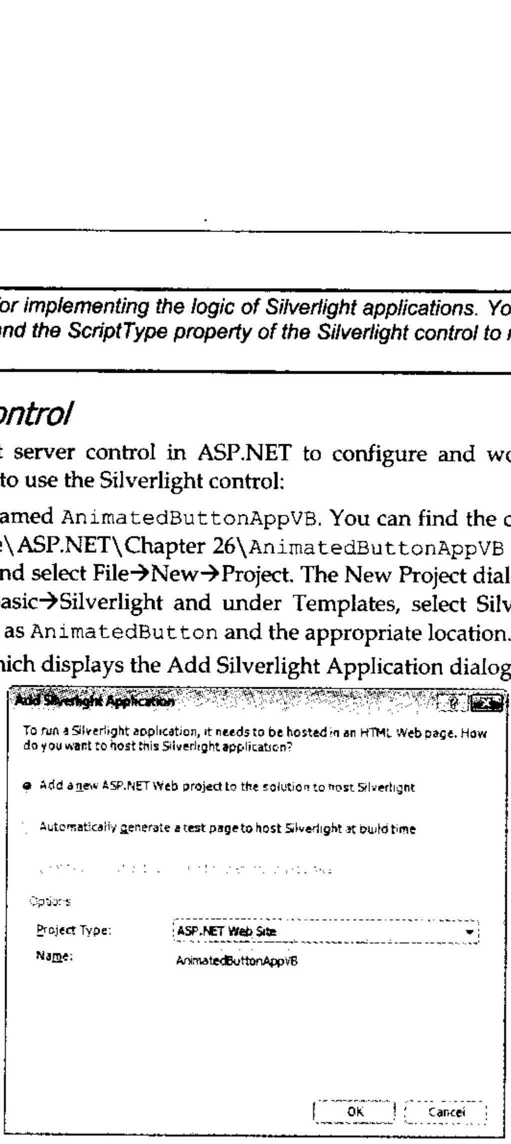
NOTE

You can also use JavaScript for implementing the logic of Silverlight applications. You need to use the Scripts collection of the ScriptManager control and the ScriptType property of the Silverlight control to reference the JavaScript library and object type, respectively.

Using the Silverlight Control

You can use the Silverlight server control in ASP.NET to configure and work with the Silverlight plug-in. Perform the following steps to use the Silverlight control:

1. Create an application named `AnimatedButtonAppVB`. You can find the code of `AnimatedButtonAppVB` application in the Code\ASP.NET\Chapter 26\`AnimatedButtonAppVB` folder on the CD. For this, open the Visual Studio IDE and select File→New→Project. The New Project dialog box appears. Under **`Project types`**, select Visual Basic→Silverlight and under Templates, select Silverlight Application. Specify the name of the application as `AnimatedButton` and the appropriate location.
2. Click the OK button, which displays the Add Silverlight Application dialog box, as shown in Figure 26.28:



Add Silverlight Application

To run a Silverlight application, it needs to be hosted in an HTML Web page. How do you want to host this Silverlight application?

● Add a new ASP.NET Web project to the solution to host Silverlight

○ Automatically generate a test page to host Silverlight at build time

Options

Project Type: ASP.NET Web Site

Name: AnimatedButtonAppVB

Figure 26.28: Selecting the Option to Host the Silverlight Content in a Website

3. Select the Add a new Web to the solution for hosting the control option and click the OK button.
4. Open the `Page.xaml` file of `AnimatedButtonAppVBapplication` and add the code as shown in Listing 26.7:

Listing 26.7: Showing the Code for the `Page.xaml` File

```
<UserControl x:Class="AnimatedButtonAppVB.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="450" Height="450" >
    <Grid x:Name="LayoutRoot" Background="Red">
        <Grid.Triggers>
            <EventTrigger RoutedEvent="Grid.Loaded">
                <EventTrigger.Actions>
                <BeginStoryboard>
                <Storyboard RepeatBehavior="Forever" AutoReverse="True">
                    <DoubleAnimation To="185" Duration="0:0:9"
                    Storyboard.TargetName="button1"
                    Storyboard.TargetProperty="Height"/>
                    <DoubleAnimation To="185" Duration="0:0:9"
                    Storyboard.TargetName="button1"
                    Storyboard.TargetProperty="Width"/>
                </Storyboard>
                </BeginStoryboard>
                </EventTrigger.Actions>
        </EventTrigger>
      </Grid.Triggers>
      <Button x:Name="button1" Height="425" Width="425" Content="Silverlight and
      ASP.NET" Background="DarkCyan" Foreground="Red" FontWeight="SemiBold"
      FontSize="15"/>
    </Grid>
  </UserControl>
```

5. Now, open the `.aspx` test page of the application, that is, the `AnimatedButtonAppVBTestPage.aspx` file. Modify the markup for the Silverlight server control (added by default to the test page) as follows:

```
<asp:Silverlight ID="Xaml1" runat="server"
Source="~/ClientBin/AnimatedButtonAppVB.xap" MinimumVersion="2.0.31005.0" Width="1000"
  Height="1000" PluginBackground="LightYellow"/>
```

6. Right-click the `AnimatedButtonAppVBTestPage.aspx` file (for VB) in the Solution Explorer and select the Set as Start Page option.
7. Run the application by pressing the F5 key. The output of the application is as shown in Figure 26.29:

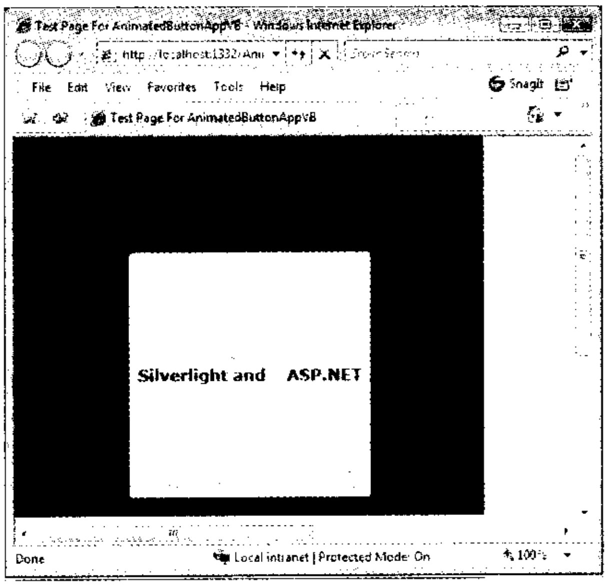

Test Page For AnimatedButtonAppVB - Windows Internet Explorer
File Edit View Favorites Tools Help
Test Page For AnimatedButtonAppVB
Silverlight and ASP.NET
Done
Local intranet | Protected Mode: On
100%

Figure 26.29: Animated Button Implemented Through Silverlight in ASP.NET Web Application

Note that the output shown in Figure 26.29 corresponds to the instance when the button has a height and width of 185 pixels.

MediaPlayer Server Control

The `MediaPlayer` server control in ASP.NET 3.5 provides a media player to playback media in your Web pages. With the `MediaPlayer` control, you can include audio and video in the ASP.NET Web pages. The audio can be in Windows Media Audio (WMA) or MP3 format and the video can be in Windows Media Video (WMV) format.

The UI of a media player is defined by its media skin, which presents the buttons to view and control the playback of the media. There are various predefined media skins at `C:\Program Files\Microsoft SDKs\Silverlight\v2.0\Libraries\Server\MediaPlayerSkins` folder that you can use on the media player. The MediaPlayerSkins folder contains several XAML files that define the media skins. In order to use a particular media skin, you need to first add the corresponding XAML file in the current website project and then reference it in the Web page. The default media skin for media player is Classic (Classic.xaml). You can open and modify the XAML files of these predefined media skins to customize the skin according to your preferences. You can also create your own media skin and refer it in the Web page.

With the media player, you can also add markers to indicate particular portions or instances in the media file. Furthermore, you can use the markers to include media chapters (or simply chapters) in the media file. You can include captions in the media file.

When you create a `MediaPlayer` control in ASP.NET Web applications, an object of the `MediaPlayer` class is created. As stated earlier, the `MediaPlayer` class (in the System.Web.UI.SilverlightControls namespaces) inherits the Silverlight class and has several properties that allow you to work with the `MediaPlayer` control. The inheritance hierarchy of the MediaPlayer class is:

```
System.Object
  Control
    WebControl
      System.Web.UI.SilverlightControls.Silverlight
        System.Web.UI.SilverlightControls.MediaPlayer
```

Table 26.6 lists the noteworthy properties of the `MediaPlayer` class:

Table 26.6: Noteworthy Properties of the MediaPlayer Class

Property	Description
`AutoLoad`	Sets or retrieves a value that indicates whether to start loading a media file just after the media player appears or to delay the loading of media file until the user clicks the start button on the media player.
`AutoPlay`	Sets or retrieves a value that indicates whether to start playing a media file as soon as the file is opened in the media player.
`Chapters`	Retrieves a list of the media chapters present in a media file.
`EnableCaptions`	Sets or retrieves a value that indicates whether to use captions.
`MediaDefinition`	Sets or retrieves the path to the media definition file.
`MediaSkinSource`	Sets or retrieves the skin identifier for the media skin used in the media player.
`MediaSource`	Sets or retrieves the media file that is being played in the media player.
`Muted`	Sets or retrieves a value that indicates whether the media file is muted.
`PlaceholderSource`	Sets or retrieves the placeholder image (.jpg or .png) that appears while the media file is being loaded.
`ScaleMode`	Sets or retrieves the mode of display for the media player. The value of this property can be one of the ScaleMode enumeration values.
`Source`	Sets or retrieves the XAML file that is used as the media skin for the media player.
`Volume`	Sets or retrieves the volume of the media. The value of this property can range between 0 (inclusive) to 1(inclusive).

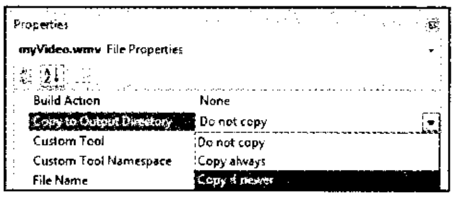
NOTE

When a MediaPlayer server control is created in an ASP.NET application, the Sys.UI.Silverlight.MediaPlayer JavaScript object is created at the client-side. This JavaScript object regulates how the user interacts with the MediaPlayer control through its properties and methods.

Using the MediaElement Control

As you know, the `MediaElement` control allows you to host an audio or video or both in a Silverlight application. Perform the following steps to see the use of the `MediaElement` control:

1. Create a Silverlight application named `MyMediaPlayerAppVB` that is hosted in a single HTML page. You can find the code of `MyMediaPlayerAppVB` application in the Code\ASP.NET\Chapter 26\`MyMediaPlayerAppVB` folder on the CD. Before that, you need to add the media file that you want to play in the current project folder. For this, you need to use the `Add Existing Item` dialog box. After adding the media file, select it in the Solution Explorer and press the F4 key. The Properties window appears, as shown in Figure 26.30:



Properties

myVideo.wmv File Properties

Build Action	None
Copy to Output Directory	Do not copy
Custom Tool	Do not copy
Custom Tool Namespace	Copy always
File Name	Copy if newer

Figure 26.30: Properties Window for the myVideo.wmv Media File

2. Select the `Copy to Output Directory` property in the Properties window. As shown in Figure 26.30, the default value of this property is `Do not` copy.
3. Now, click the down arrow next to the property and select the `Copy if newer` option. This option copiesthe media file to the current project folder if the media file is new.
4. Now, open the `Page.xaml` file of the application and type the code to use `MediaElement` control. Listing 26.7 shows the code for the `Page.xaml` file of `MyMediaPlayerAppVB` application:

Listing 26.7: Showing the Code for the `Page.xaml` File

```
<UserControl x:Class="MyMediaPlayerAppVB.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="600" Height="600">
    <Grid x:Name="LayoutRoot" Background="SteelBlue">
        <Border BorderBrush="Yellow" BorderThickness="3"/>
        <TextBlock Text="My Media Player" HorizontalAlignment="Center" FontWeight="Bold"
        FontSize="24" Margin="0,10,0,0" Foreground="Yellow"/>
        <Button x:Name="play" Height="50" Width="50" Content="Play" Margin="0,0,100,425"
        Background="Yellow" Click="mediaPlay"/>
        <Button x:Name="pause" Height="50" Width="50" Content="Pause" Margin="0,0,0,425"
        Background="Yellow" Click="mediaPause"/>
        <Button x:Name="stop" Height="50" Width="50" Content="Stop" Margin="100,0,0,425"
        Background="Yellow" Click="mediaStop"/>
        <Border Height="425" Width="525" BorderBrush="Yellow" BorderThickness="3"
        Margin="0,100,0,0"/>
        <MediaElement x:Name="mediaEle" Height="400" Width="500" Margin="0,100,0,0"
        Source="myVideo.wmv" Stretch="UniformToFill" Volume="0.75" AutoPlay="False"
        Balance="1" MediaOpened="mediaOpen" MediaFailed="mediaError"/>
        <TextBlock x:Name="errorText" Text="" Height="50" Width="100" Foreground="Yellow"/>
        <TextBlock x:Name="mediaFile" Height="50" Width="200" Foreground="Yellow"
        Text="Media File: " Margin="0,565,0,0" FontWeight="Bold"/>
        <TextBlock x:Name="duration" Height="50" Width="200" Foreground="Yellow"
        Text="Total Time: " Margin="400,565,0,0" FontWeight="Bold"/>
    </Grid>
</UserControl>
```

In Listing 26.7, a `MediaElement` control is added to host a video file (myVideo.wmv). The `Stretch` property of the `MediaElement` control is set to `UniformToFill`, which implies that the video occupies entire control without changing the original aspect ratio. If the control is smaller than the video, the video is clipped to fit the control. The `Volume` property of the `MediaElement` control is set to `0.75`, which implies that the volume at which the media plays is 75% of the maximum volume. In addition, the `AutoPlay` property of the `MediaElement` control is set to `False`, which implies that the playback of the media hosted in the control does not start automatically after the media file is opened but you need to manually start the playback. In addition, the `Balance` property is set to `1`, which means that the right stereo speaker has full volume. There are event handlers associated with the `MediaOpened` and `MediaFailed` events of the `MediaElement` control. Note that there are three buttons named `Play`, `Pause`, and `Stop`, which when clicked starts, pauses, and stops the playback of the media, respectively.

5. Open the code-behind file of the `Page.xaml` file of `MyMediaPlayerAppVB` application and type the code as shown in Listing 26.8:

Listing 26.8: Showing the Code for the Code-Behind File

```
Partial Public Class Page
    Inherits UserControl
    Public Sub New()
        InitializeComponent()
    End Sub
    Public Sub mediaOpen(ByVal sender As Object, ByVal e As RoutedEventArgs)
        mediaFile.Text += mediaEle.Source.ToString()
        duration.Text += mediaEle.NaturalDuration.TimeSpan.Hours.ToString() & ":" &
        mediaEle.NaturalDuration.TimeSpan.Minutes.ToString() & ":" &
        mediaEle.NaturalDuration.TimeSpan.Seconds.ToString()
    End Sub
    Public Sub mediaError(ByVal sender As Object, ByVal e As RoutedEventArgs)
        errorText.Text = "Failed to open the media"
    End Sub
    Public Sub mediaPlay(ByVal sender As Object, ByVal e As RoutedEventArgs)
        mediaEle.Play()
        play.Content = "Play"
    End Sub
    Public Sub mediaPause(ByVal sender As Object, ByVal e As RoutedEventArgs)
        mediaEle.Pause()
        play.Content = "Resume"
    End Sub
    Public Sub mediaStop(ByVal sender As Object, ByVal e As RoutedEventArgs)
        mediaEle.Stop()
        play.Content = "Play"
    End Sub
End Class
```

In Listing 26.8, the `mediaOpen()` method executes when the media in the `MediaElement` control is opened. This method displays the name and the length (total time) of the media file in the application by using the `Source` and `NaturalDuration` properties of the `MediaElement` control. The `mediaPlay()` method executes when the play button is clicked. This method calls the `Play()` method of the `MediaElement` control. Similarly, when you click the `pause` and `stop` buttons in the application, the `Pause()` and `Stop()` methods of the `MediaElement` control is called in the `mediaPause()` and `mediaStop()` methods, respectively. In case, there are any errors while opening or playing the media, the `mediaError()` method executes.

6. Run the application by pressing the F5 key, which displays the output as shown in Figure 26.31:

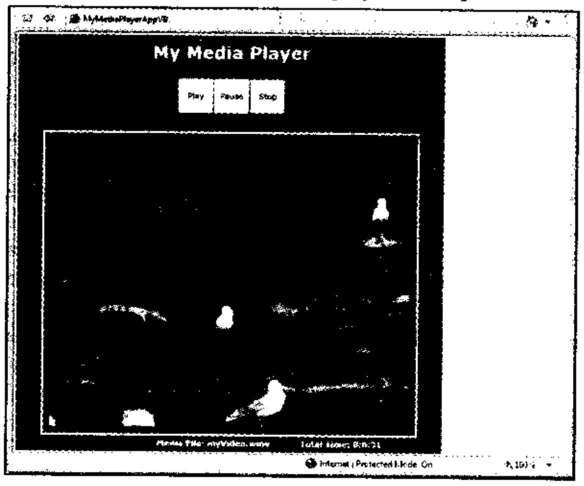


Figure 26.31: Output of the MyMediaPlayerAppVB Application

You can click the Play, Pause, and Stop buttons in the application to play, pause, and stop the media file, respectively.

Summary

In this chapter, we have described the Silverlight technology, along with its main features and architecture. We have also learned how this technology offers a cross-platform, cross-browser, and cross-device environment to develop interactive and dynamic Web applications. This chapter further discusses how to install the Silverlight plug-in and SDK and develop Silverlight applications by using various controls, such as DatePicker, GridSplitter, MediaElement, in Visual Studio 2008. In addition, we learned how ASP.NET supports Silverlight content in Web applications. In the next chapter, we will discuss about new technology i.e. XML, used for creating web applications.

Quick Revise

Q1. What is Silverlight? Why it is introduced?

Ans: Silverlight is a Web-based technology that allows you to develop Web applications that contain high-fidelity multimedia content and eye-catching visual effects. Silverlight is an integration of the rich user interface (UI) of desktop applications and the transparency of other Web languages, such as HTML and JavaScript. Silverlight is introduced to overcome the limitations of the technologies and languages used to develop interactive Web pages with the help of animations and multimedia. As these technologies are expensive to deploy, varied in performance, and also have inadequate compatibility with different platforms. However, Silverlight is inexpensive small plug-in software that can work in a cross-platform, cross-browser, and cross-device environment.

Q2. What is Deep Zoom technology in Silverlight?

Ans: Deep Zoom technology in Silverlight 2.0 facilitates the zooming in and out of high-resolution images. With the Deep Zoom technology, developers can deliver unbelievably creative and uniquely interactive Web applications because it allows users to smoothly and simultaneously zoom in or out of multiple photographs. The photographs have a highly detailed and fine view.

Q3. What is the use of the MediaPlayer server control in ASP.NET?

Ans: The MediaPlayer server control in ASP.NET 3.5 provides a media player to playback media in Web pages. With the MediaPlayer control, both audio and video can be included in the Web pages. The UI (also called the media skin) of the MediaPlayer control can be changed. Various markers can be added in the MediaPlayer control to indicate particular instances or chapters in the media file.

Q4. What is the functionality provided by the Silverlight server control in ASP.NET?

Ans: The Silverlight server control enables the developers to display any Silverlight content, which is basically XAML content. This control can also be used to embed managed assemblies, application packages, and client-side JavaScript libraries.

Q5. Enlist some of the scenarios where the Silverlight can be used?

Ans: Silverlight is designed for developing rich and interactive Web-based content. The following are some of the Web applications and real-world scenarios, where Silverlight can be used:

- Playing Dynamic videos with ads, audio playback, etc.
- Mini application such as Casual games and gadgets

Q6. What are the primary languages that Silverlight uses?

Ans: Silverlight uses the following languages:

- Hypertext Markup Language (HTML)
- JavaScript
- eXtensible Markup Language (XAML)

Q7. Name the control that can be used to enable the users to adjust the space in a grid?

Ans: The GridSplitter control in Silverlight is 2.0 enables users of the Silverlight application to to redistribute the space between the rows and columns of the grid.

Q8. What are the networking services that are supported by Silverlight?

Ans: The various networking services that Silverlight supports are:

- Hypertext Transfer Protocol (HTTP)
- Really Simple Syndication (RSS)
- Representational State Transfer (REST) services

Q9. What is the role of XAML files in Silverlight?

Ans: The design and UI of Silverlight application are defined through XAML Every Silverlight application has a few XAML files, such as App.xaml, App.xaml.vb, Page.xaml, and or Page.xaml.vb, which define the UI and basic behavior of a Silverlight application. The App.xaml and the App.xaml.vb files represent the application and the Page.xaml and Page.xaml.vb files represent the UI of the application.

Q10. What are application packages in Silverlight?

Ans: When you compile Silverlight 2.0 applications, they are compiled into application packages. The application packages are files with the .xap extension. A .xap (pronounced as zap) file consists of all the necessary files and resources for running a Silverlight application

PART 5
CREATING XML APPLICATIONS