Custom Class-based Web Service

Web Services can also be based on custom classes. The code contained in the Web Matrix Custom Class template uses `OrderDetails` and `OrderItem` classes to detail orders:

```
<%@ WebService language="VB" class="menuCustom" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization
Imports System.Web.Services.Protocols

Public Class menuCustom

    <WebMethod> Public Function GetOrderDetails()
        Dim orderDetails As New OrderDetails()

        ' Set values for OrderDetails class
        orderDetails.OrderNumber = 35
        orderDetails.CompanyName = "ACME Paint"
        orderDetails.CustomerFirstName = "John"
        orderDetails.CustomerLastName = "Smith"
        orderDetails.CustomerEmail = "John.Smith@IBuySpy.com"

        ' Return an array of 2 OrderItems
        orderDetails.OrderItems As New OrderItem[1];

        orderDetails.OrderItems[0].ItemName = "Sunset Yellow"
        orderDetails.OrderItems[0].ItemId = 12
        orderDetails.OrderItems[0].ItemName = "at the beach"

        orderDetails.OrderItems[1].ItemName = "Seattle Sky Blue"
        orderDetails.OrderItems[2].ItemId = 93
        orderDetails.OrderItems[3].ItemName = "A rare shade of blue"

        Return orderDetails
    End Function

End Class

Public Class OrderDetails

     ' Serialize as an attribute named OrderId
    <XmlAttribute("OrderId")> Public OrderNumber As Integer

    ' Serialize as an attribute
    <XmlAttribute()]> Public CompanyName As String

    Public CustomerFirstName As String

    Public CustomerLastName As String

    ' Serialize as an element, but change the name
    <XmlElement("email")> Public CustomerEmail As String
```

```
    ' Rename the array OrderItemDetail
    <XmlArray("OrderItemDetail")> Public OrderItem[] OrderItems
End Class

Public Class OrderItem

    ' Serialize all member variables as attributes

    <XmlAttribute()> Public ItemName As String

    <XmlAttribute()> Public ItemId As Integer

    <XmlAttribute()> Public ItemDescription As String
End Class
```

Both classes (`OrderDetails` and `OrderItem`) have pubic variables that are decorated with attributes that specify how the property should be serialized. The Web Method of the custom class, `GetOrderDetails`, returns an instance of the `OrderDetails` class, which is serialized according to the specified attributes.

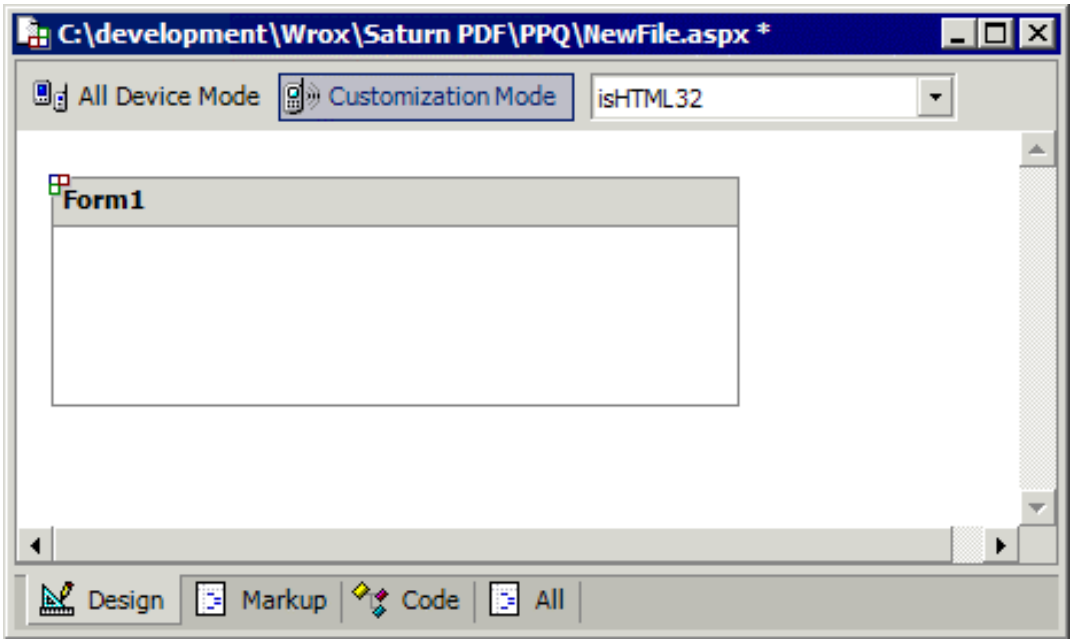
Using Other Components

In addition to the standard HTML and ASP.NET Server controls, there are plenty of other controls that can be used with Web Matrix. The market for third-party controls is growing quickly, and these can be added to Web Matrix by customizing the Toolbox. You can even create your own custom controls and add them to the Toolbox. There are also two additional sets of controls supplied by Microsoft: the Mobile Internet Toolkit Controls for creating applications for mobile devices and the Internet Explorer Web Controls for creating rich content for IE.

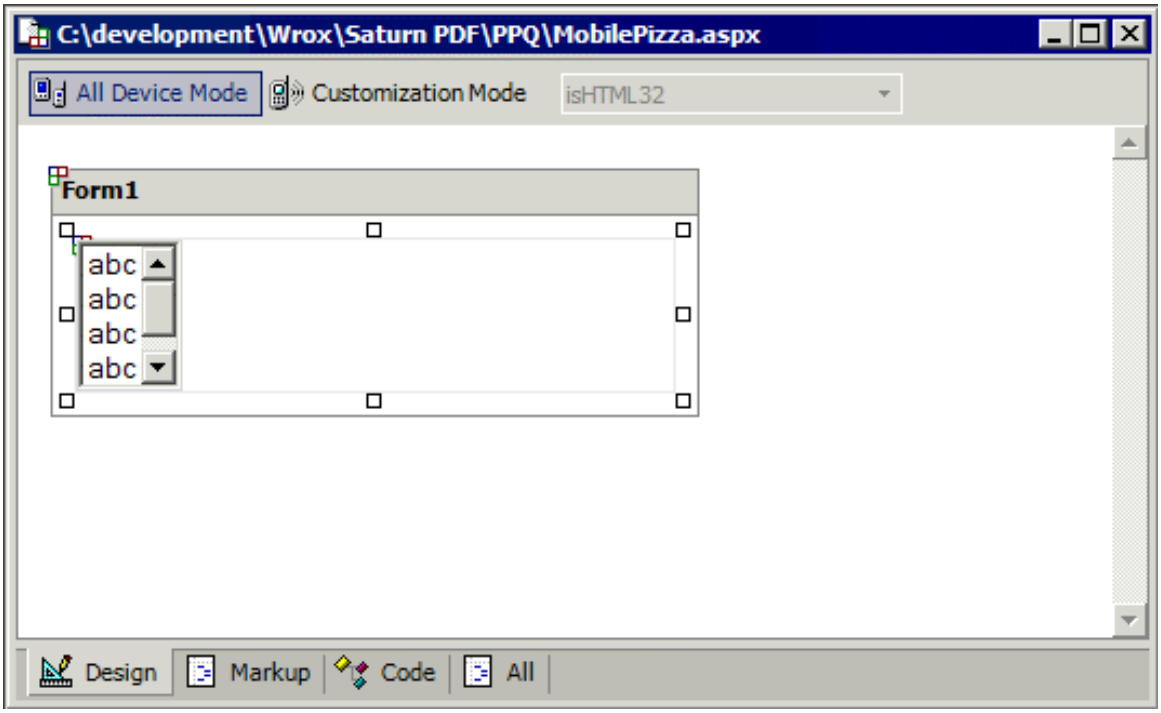
Web Matrix uses the same control designer model as Visual Studio .NET, so controls will work in both tools.

Mobile Internet Toolkit Controls

Support for mobile devices is just as good as for traditional browser pages, but you will need to install the Mobile Internet Toolkit first. You have the choice of a Simple Mobile Page or a Simple Mobile User Control, both of which look like this:



By default this page will support all mobile devices, but you can select the Customization Mode, which allows you to target specific implementations of devices. Although the designer differs in look, its use is exactly the same as for other pages. You drag and drop the mobile controls onto the design surface, and then add your own code. For example, we can add a selection list for pizzas:



Here we've added a `SelectionList` control to the form and set its `SelectType` property to `MultiSelectListBox`, and the `DataTextField` property to `name` (for the name of the pizza). The code to run when the form loads is:

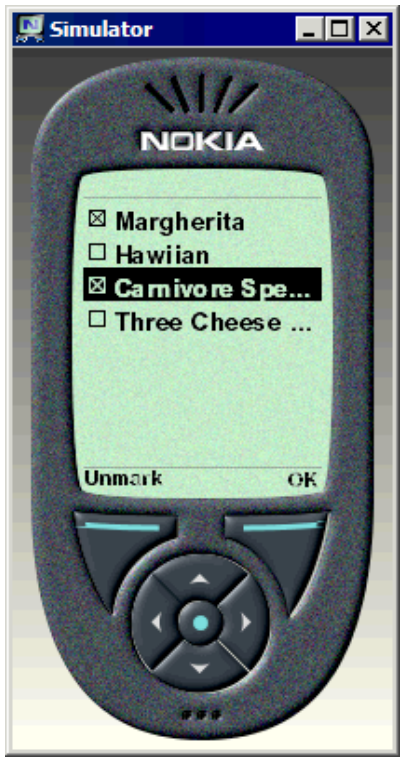
```
Sub Form1_Load(sender As Object, e As EventArgs)

    Dim ds As DataSet = DataLayer.GetData()

    Pizzas.DataSource = ds.Tables("pizza").DefaultView
    Pizzas.DataBind()

End Sub
```

The following screenshot shows how the resulting page is displayed by the Nokia emulator:



This example shows how developing mobile applications with Web Matrix is just as easy as standard web applications development.

Internet Explorer Web Controls

The Internet Explorer Web Controls are a set of server controls that provide rich DHTML support for use in Internet Explorer (version 5.5 and above). These can be accessed by customizing the Web Matrix Toolbox. You can then use them as you would any other control, by dragging them onto the design surface. Of the four controls supplied in the IE set, only the TabStrip has designer support – the rest must be customized in either HTML or Code view.

Building and Use a Custom Control

Building custom server controls can easily be done using Web Matrix, since such controls are just classes. However, Web Matrix doesn't support the automatic building of class files, or some of the advanced features that you might want to use when creating custom controls, such as the ability to embed resources such as a bitmap.

Despite these limitations, and the fact that Web Matrix is designed primarily for editing ASP.NET pages, it's still a fine choice for creating custom controls. Here you can see a simple site content control I've created (which supports templating) by adding a class and using the code editor:

```
using System;
using System.Web;
using System.Web.UI;
using System.Collections;
using System.Web.UI.WebControls;
using System.ComponentModel;

[assembly:TagPrefix("ipona.Controls", "ipona")]
[assembly: AssemblyKeyFile("..\\..\\ipona.snk")]

namespace ipona.Controls
{
  [ToolboxData("<{0}:SiteContent runat=\"server\"></{0}:SiteContent>")]
  public class SiteContent : WebControl, INamingContainer
  {
    . . .
```

Since this control will be added to the toolbar the control needs a strong name, which needs to be created manually using the `sn.exe` utility (you can find more information about `sn.exe` in the .NET Framework SDK documentation). Then we can compile the class manually, and use the Add Local Toolbox Components option on the Tools menu to add this control to the Toolbox:

We haven't added any designer support, but we can drag this control onto a page and then add the content via the HTML view:

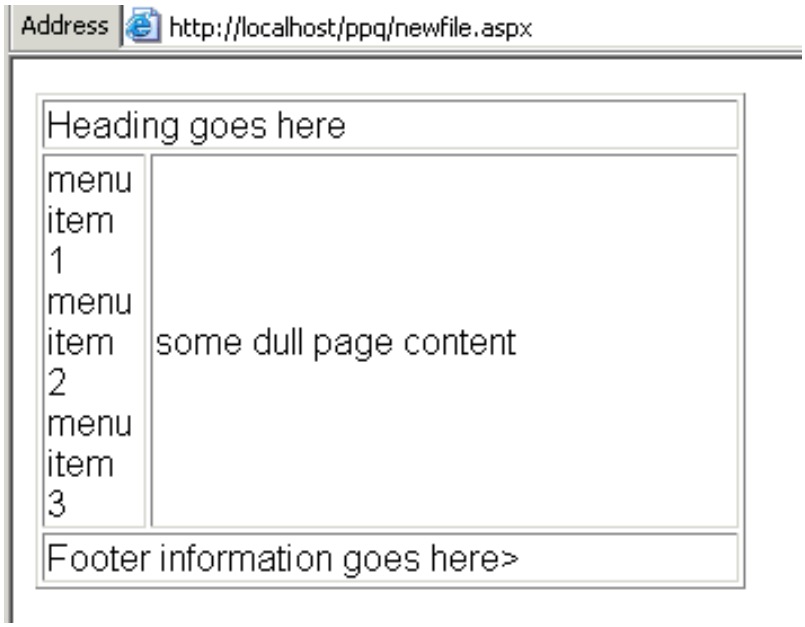
```
<ipona:SiteContent id="SiteContent1" style="WIDTH: 315px; HEIGHT: 151px"
       runat="server">
    <HeadingTemplate>
        Heading goes here
    </HeadingTemplate>

    <MenuTemplate>
        menu item 1<br/>
        menu item 2<br/>
        menu item 3<br/>
    </MenuTemplate>
```

```
    <ContentTemplate>
        some dull page content
    </ContentTemplate>

    <FootingTemplate>
        Footer information goes here>
    </FootingTemplate>
</ipona:SiteContent>
```

If we view the page we'll see that the control acts just like any other:



Part 3 – Configuring and Extending Web Matrix

Web Matrix is designed to be extensible and highly configurable, which allows it to be tailored to suit individual requirements. In this final section, we'll look at the different ways that we can take advantage of this. We'll cover:

- The Web Matrix configuration settings files
- The Web Matrix Preferences Dialog
- Creating and modifying the document templates that appear in the New File dialog
- Installing and using Add-ins and Code Builders, which can be used to achieve specific tasks
- Customizing the user interface in general

The Web Matrix Configuration Files

Just like ASP.NET, Web Matrix obtains most of its run-time settings from a configuration file. This file, named `WebMatrix.exe.config`, is located in the Matrix program folder: `\Program Files\Microsoft ASP.NET Web Matrix\`*`version`*`\`

This folder also contains a file named `ClassBrowser.exe.config`, which contains the run-time settings for the Class Browser tool. The `ClassBrowser.exe.config` file is similar in format to the corresponding section of the `WebMatrix.exe.config` file.

The format and content of Web Matrix's configuration files is likely to change as the product develops, but the current version of `WebMatrix.exe.config` contains the following sections. Note that "saturn" was the early project/development name for Web Matrix, and is still used internally:

```
<configuration>

  <configSections />        - config file sections and corresponding handlers
  <runtime />               - the assemblies used by Web Matrix
  <appSettings />           - application-specific values for Web Matrix

  <microsoft.saturn>

    <packages />            - the packages that actually implement Web Matrix
    <projects />            - the assemblies to create each project type
    <documentTypes />       - the types of document and template available
    <toolbox />             - the sections available in the Toolbox
    <addIns />              - the Add-ins that are available
```

```
  <webLinks />            - the list of Help and Community links
      <classView />           - the assemblies and folders in the Classes window
      <dataObjectMappings />  - data type conversion formats for cutsom objects

    </microsoft.saturn>

  </configuration>
```

By editing the contents of these sections of the configuration file, you can change the appearance of Web Matrix, as well as controlling what items appear where in the IDE. We'll look at some examples later in this section.

Note that Web Matrix caches the actual run-time settings generated from the configuration file in a separate file named `WebMatrix.settings`, which is located in your own "user profile" folder. By default this is:

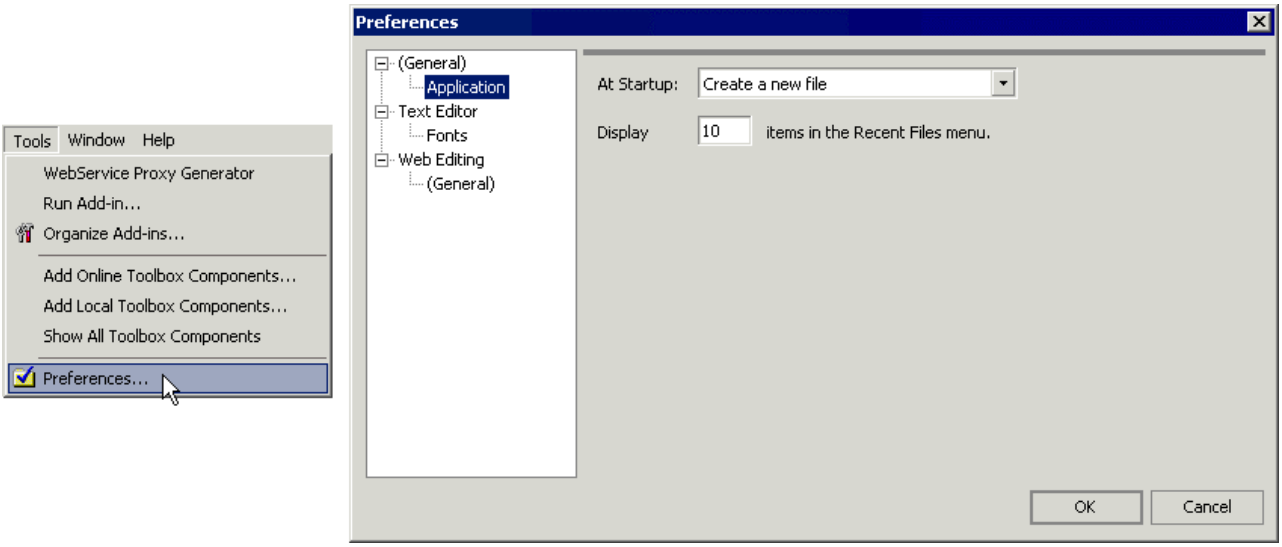
```
\Documents and Settings
 \[username]
  \Application Data
   \Microsoft Corporation
    \ ASP.NET Matrix Project
     \[version]
      \WebMatrix.settings
```

This file is generated automatically each time you close Web Matrix, and then read when you restart Web Matrix.

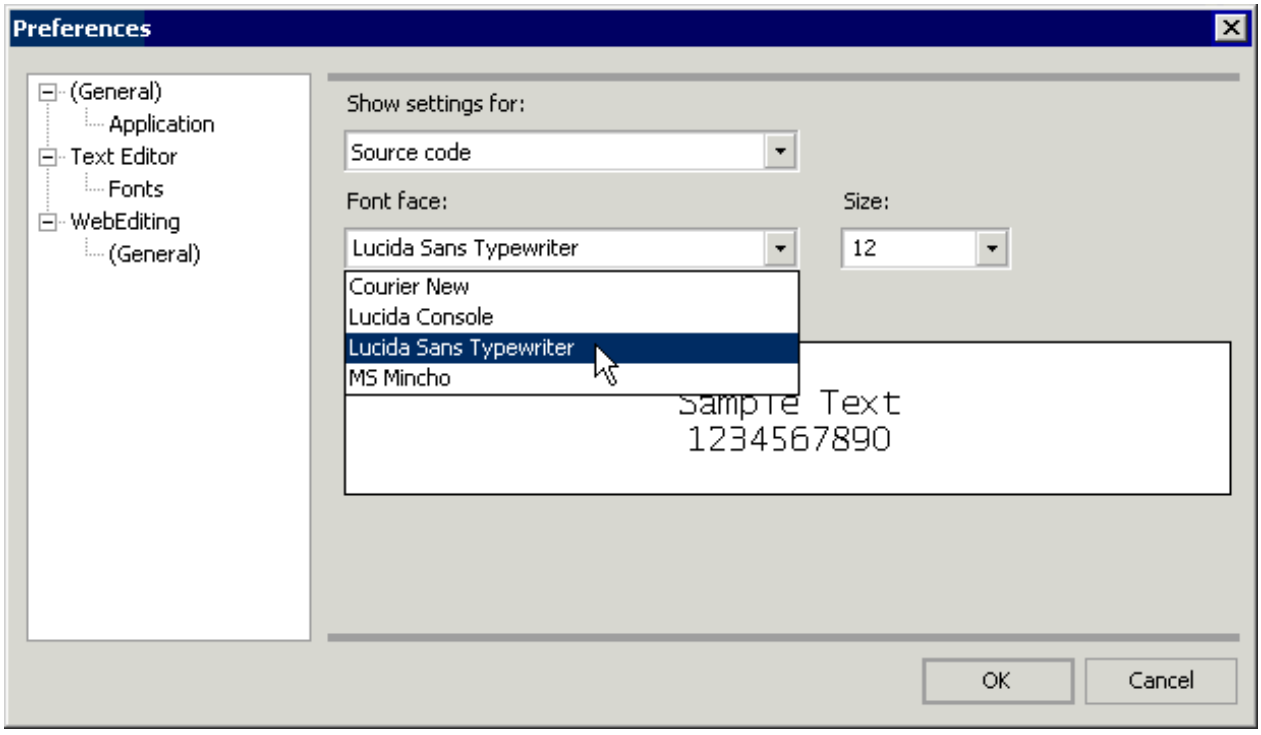
If you find that your changes to the `WebMatrix.exe.config` file are not picked up when you next start Web Matrix, you should manually delete the `Matrix.settings` file in order to force Web Matrix to re-read the `WebMatrix.exe.config` file and generate a new `WebMatrix.settings` file when you close it down again. Note that this will remove any assemblies that you have added to the Classes window using the Customize dialog. To permanently add assemblies to the `Classes` window, you must edit the `WebMatrix.exe.config` file as shown in *Customizing the Classes Window*.

The Preferences Dialog

You can access the Web Matrix Preferences dialog from the Tools menu. It allows you to modify several features on the IDE. In the (General) section, under the single current entry Application, you can specify what action should take place when Web Matrix starts (show the New File dialog, the Open File dialog or neither) and control how many entries should appear in the list of recent files on the File menu:

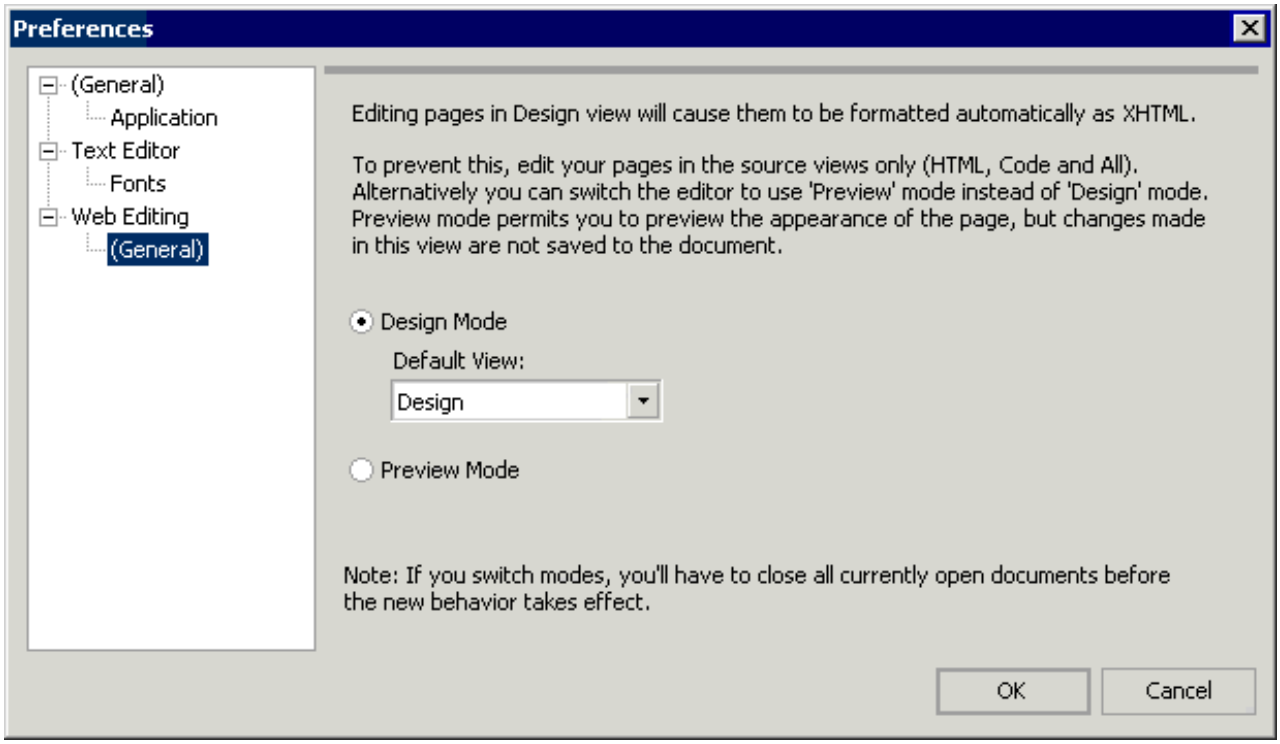


At the moment the Text Editor section just contains the Fonts page. Here you can specify the font to be used for the Edit window in both Source and Code view, and a different font to use for printing the source or code. Only a few fixed-width (mono-space) fonts are available, but you can specify the size and preview the result:



The Web Editing section has a (General) page where you can specify how Web Matrix should treat your code when it's displayed in the edit window. We discussed the two options – Design Mode and Preview Mode – in *The Edit Window* and *Using Preview Mode* in *Part 1*.

In Design Mode, you can edit ASP.NET pages, user controls, and HTML pages in Design view and HTML view (and Code view for ASP.NET pages and user controls). However, in this mode Web Matrix might reformat the code to XHTML when you switch to Design view (so that the resulting rendered appearance can be shown in the edit window). To prevent this, you can switch Web Matrix to Preview Mode, in which case the edit window only has the Source and Preview tabs for ASP.NET pages and user controls, and the HTML and Preview tabs for HTML pages. That way, content can only be edited in Source or HTML view and switching to Preview view does not cause it to be reformatted:



The previous screenshot also shows a drop-down list in which you can specify which view (tab) should be the default when you are using Design Mode. If you prefer you can change the Default View from Design (where the Design view tab is open in the edit window by default) to Source. With this setting, the edit window opens in Code, All, or HTML view by default – depending on the type of file you are editing.

The list of options that are currently available will undoubtedly expand as Web Matrix evolves and develops during its Beta program.

Creating and Modifying Document Templates

Document templates are used to define the items that appear in the New File dialog, and they contain the content for the pages you create from this dialog. You can edit the document templates that are provided with Web Matrix, and also add your own to suit the kinds of pages and files you regularly create. The template files are stored in a series of subfolders within:

```
\Program Files
 \Microsoft ASP.NET Web Matrix
  \[version]
   \Templates\
```