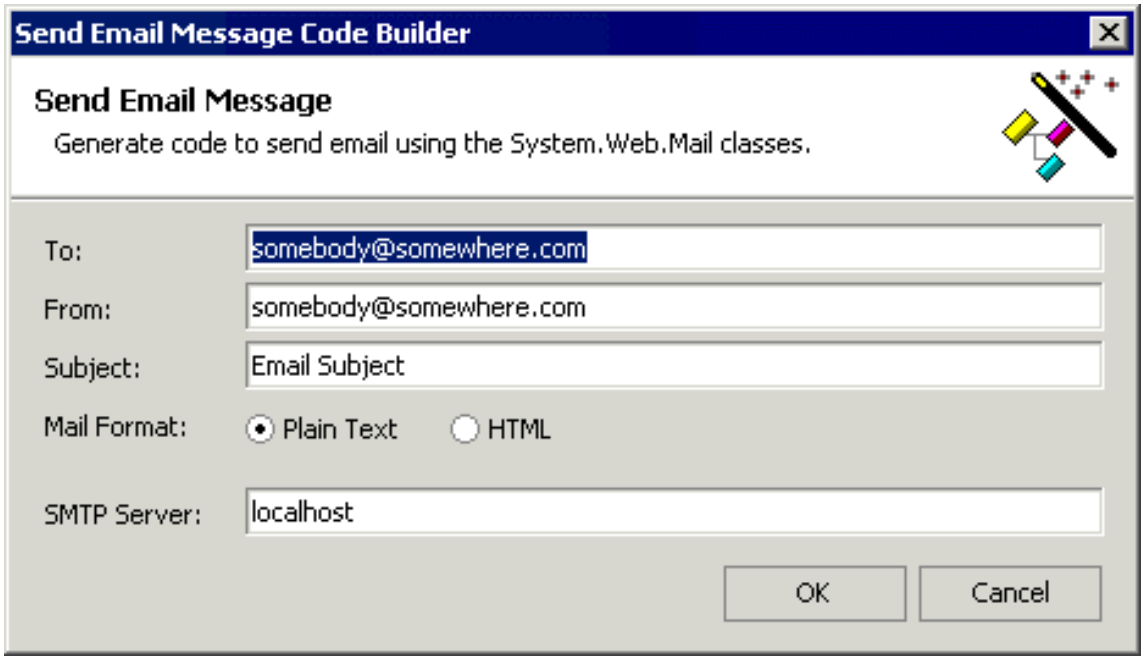


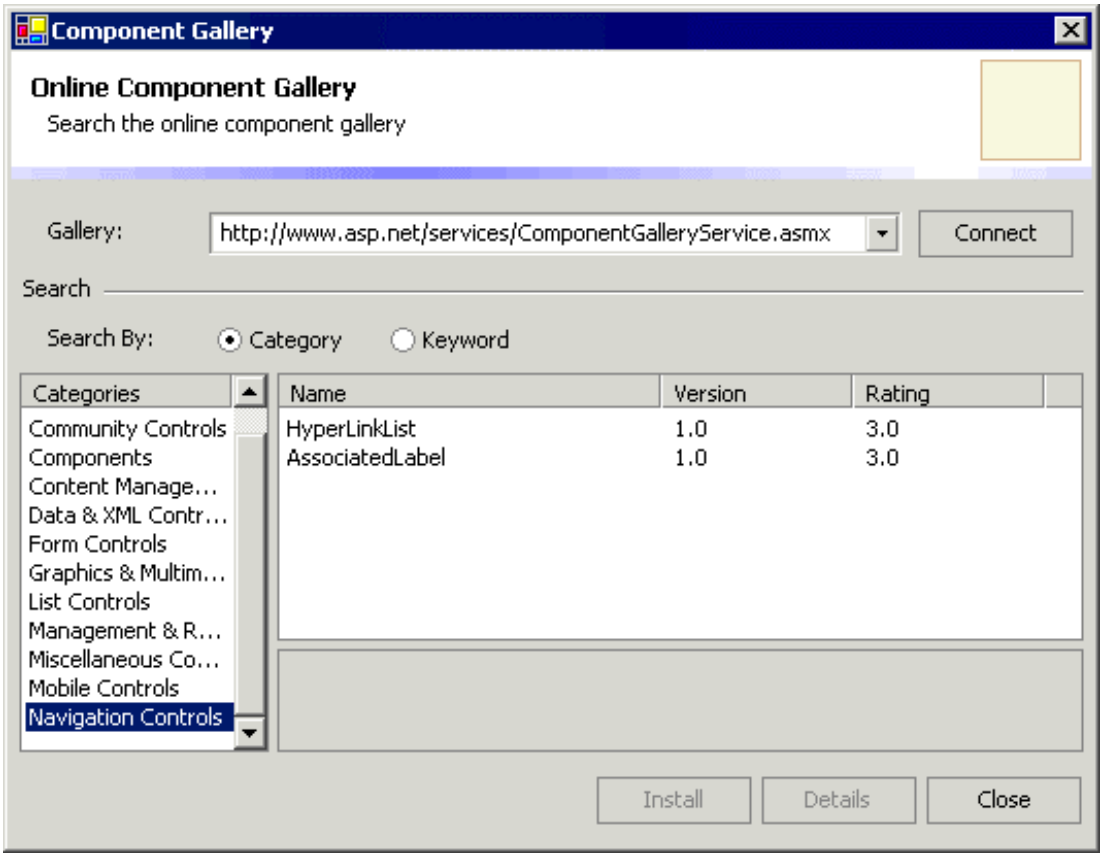
The following code is then created. All that is left to do is to set the text of the message using the `mailMessage.Body` property:

```
' Build a MailMessage
Dim mailMessage As System.Web.Mail.MailMessage = New System.Web.Mail.MailMessage
mailMessage.From = "somebody@somewhere.com"
mailMessage.To = "somebody@somewhere.com"
mailMessage.Subject = "Email Subject"
mailMessage.BodyFormat = System.Web.Mail.MailFormat.Text

' TODO: Set the mailMessage.Body property

System.Web.Mail.SmtpMail.SmtpServer = "localhost"
System.Web.Mail.SmtpMail.Send(mailMessage)
```

Adding Controls to the Toolbox

You can add any type of .NET control to the Toolbox – the controls from the Microsoft Mobile Internet Toolkit (MIT), the Internet Explorer Web Controls, or controls that you install from the Web Matrix community web site. To install controls into the Custom Controls section of the Toolbox, select Add Online Toolbox Components or Add Local Toolbox Components from the main Tools menu, or right-click on the Toolbox itself.

If you select Add Online Toolbox Components, Web Matrix connects to the community web site and shows the controls that are available for download. In the following screenshot we've selected the DHTML Controls section, and you can see that it contains the IE Web Controls (note that the range of Online Component Gallery controls displayed here will expand and change regularly over time):

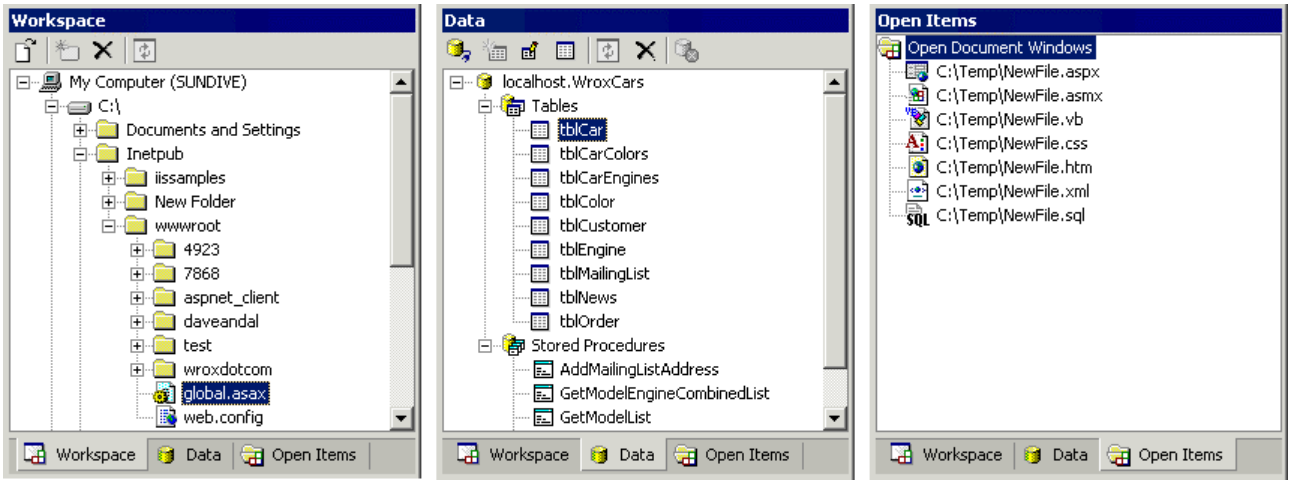


If you select Add Local Toolbox Components, a dialog opens that allows you to select an assembly from the `%WINDOWS%\Microsoft.Net\Framework\`*`version`*`\` folder on your system by default, or from any other folder that contains a suitable Component Library DLL.

The "Project" Window (Upper Half)

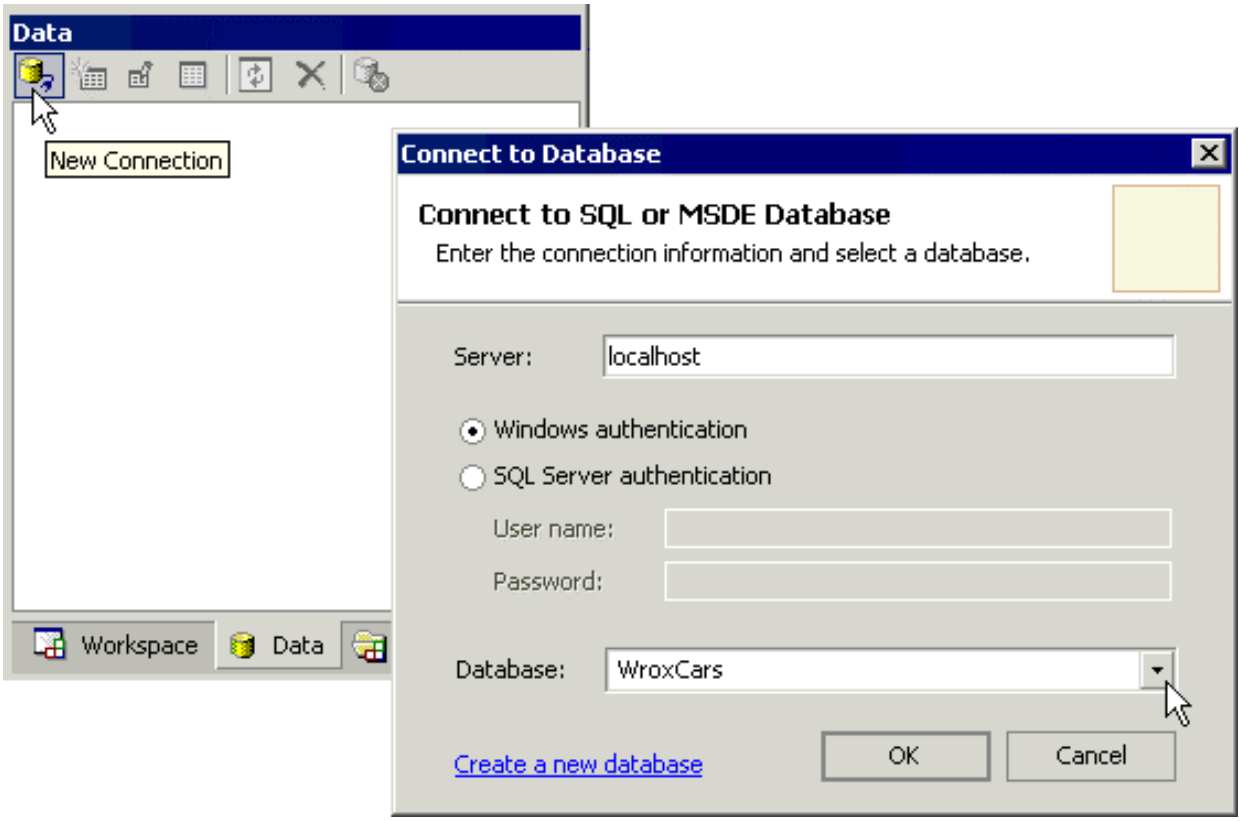
Down the right-hand side of the screen are two windows that together make up what we refer to here as the "project window". The upper half provides three options:

- Workspace – which shows the drives and files on your local machine and, by default, any mapped folders. You can also add shortcuts to other local or network folders, or to other servers via FTP using the commands on the Workspace menu. Double-clicking a file here opens it in a new editing window. By right-clicking in the Workspace window you can add a shortcut, create a new folder, or create a new file based on one of the standard types available in the New File dialog. We'll look at the available file types in *Types of Files and Wizards* later on.
- Data – which allows you to connect to a SQL Server or MSDE database and then view, create, and edit tables and stored procedures within that database. The following screenshot shows a connection to a database named WroxCars, and you can see the tables within this database displayed.
- Open Items – which lists the edit windows that are currently open within the IDE. You can simply click on one to bring it to the front for editing.



Using the Data Window

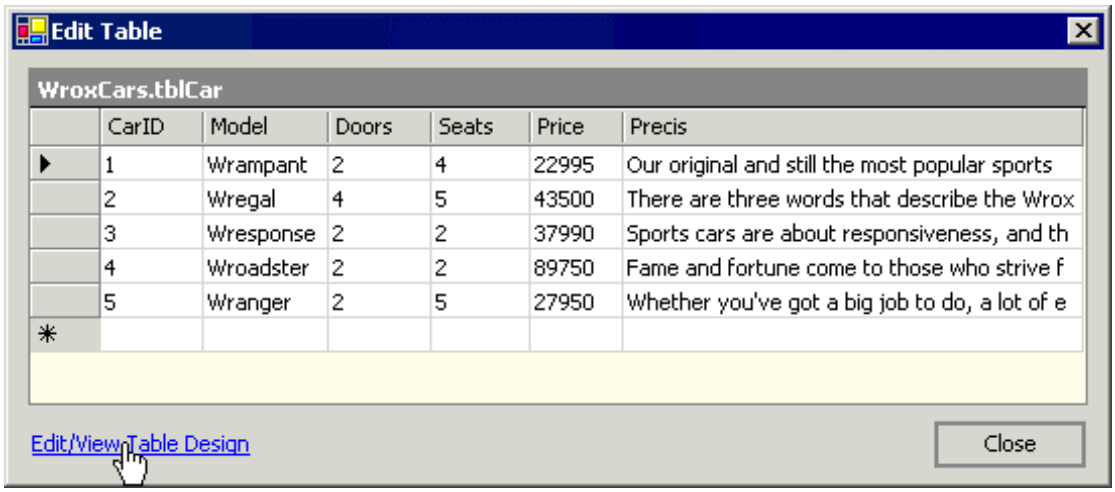
To connect to a data source click the icon at the top of the Data window, as shown in the following screenshot, and enter the connection details for the database. You can generally use Windows authentication for the local SQL Server or MSDE database, or if you prefer you can specify the user name and password for SQL Server authentication. Then, select the database from the drop-down list:



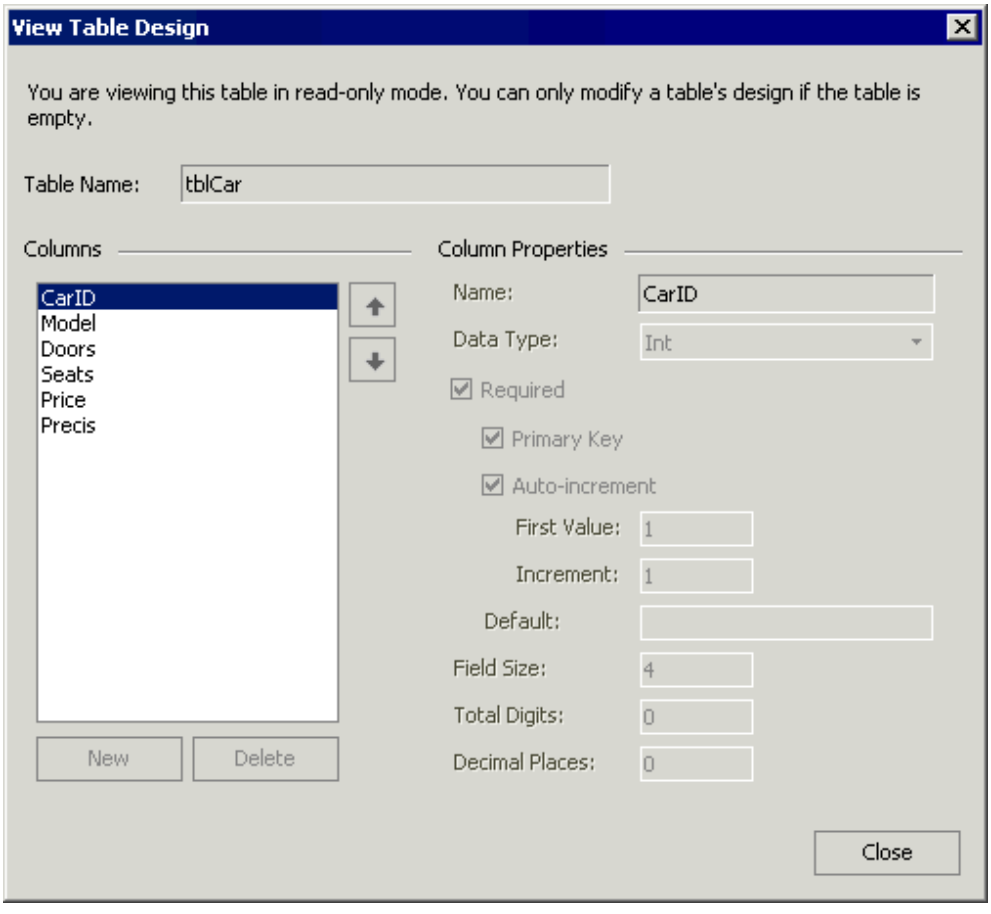
There is also an option to create a new database on the specified server. All you do is enter the name for the new database in the dialog that appears, and the database is created and displayed in the Data window.

Working with a Data Source

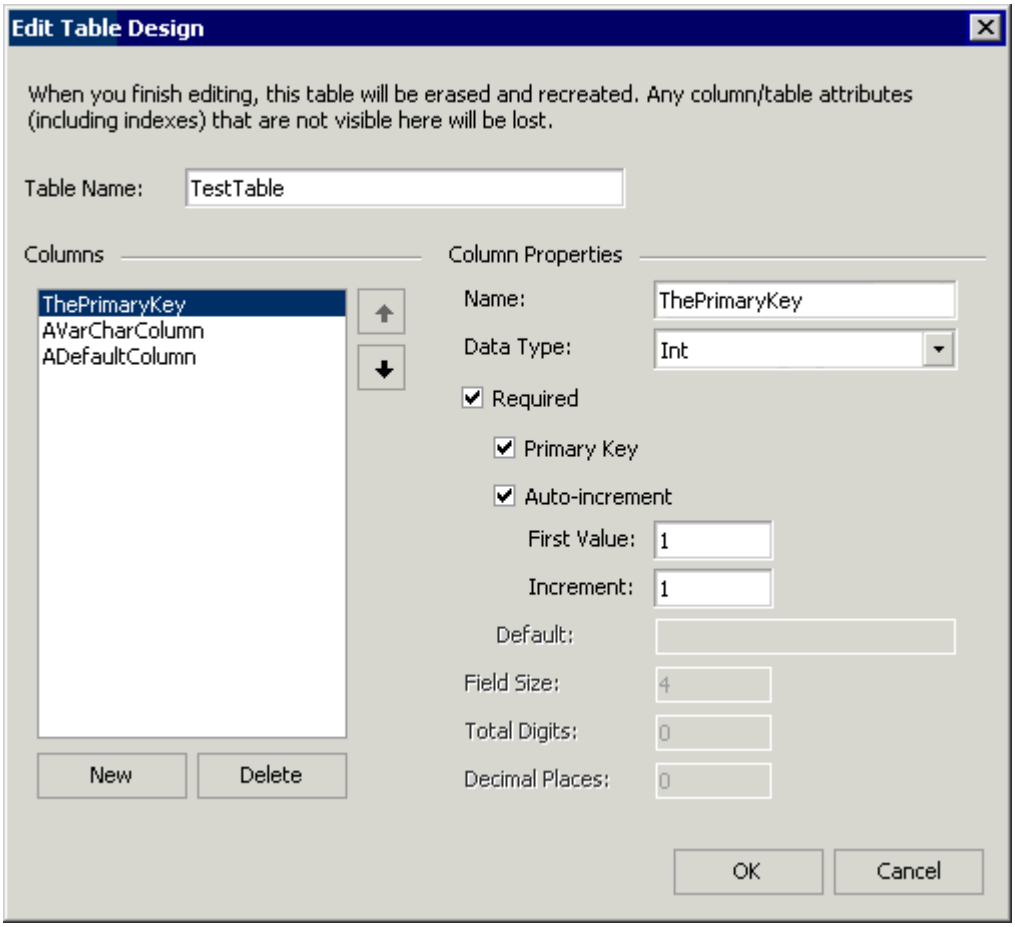
Once you have at least one database open in the Data window (you can connect to more than one at a time if you wish), the remaining icons at the top of this window are enabled. You can use these icons to work with the database – you can create a new table or stored procedure (using the second icon), or edit existing ones (using the third icon). For example, in the following screenshot, we selected our tblCars table and clicked the Edit icon. You can see the contents of the table, and the row values can be edited directly. Note, however, that this only works if the table has a primary key defined:



As well as editing the contents of the table, you can also view the table structure by clicking the link at the bottom of the dialog shown in the previous screenshot:

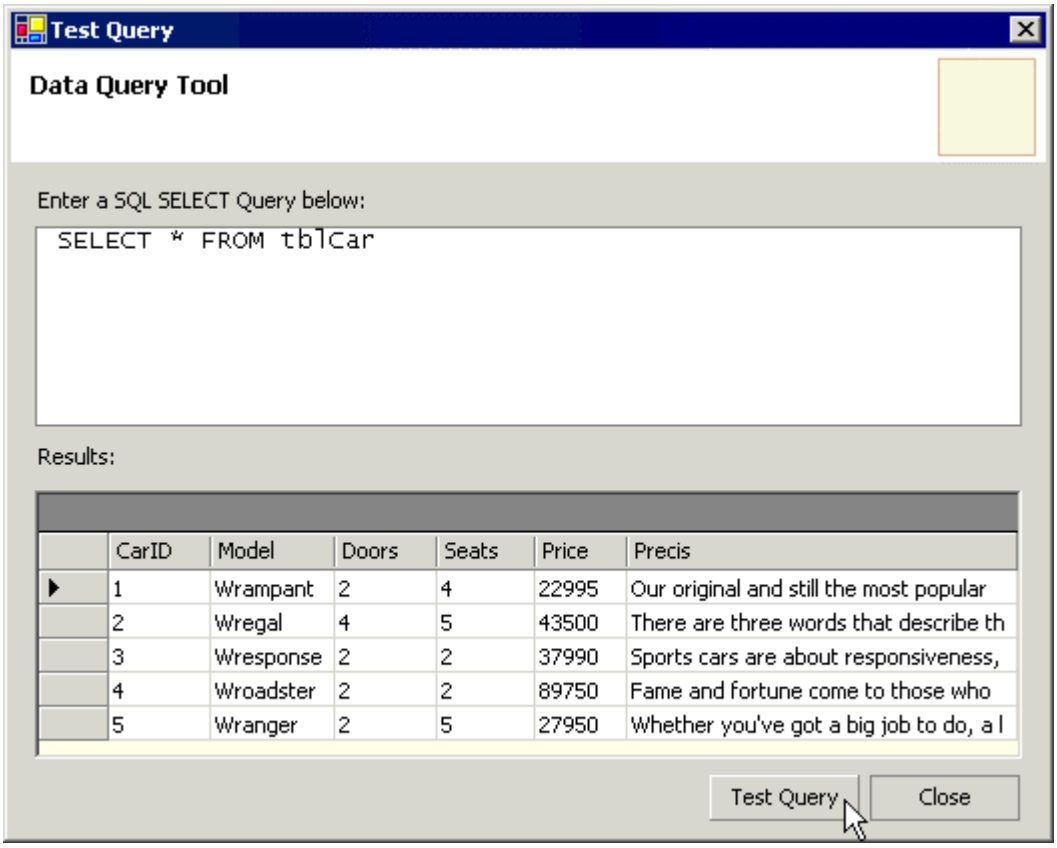


However, you can only edit the table structure if the table is empty. The following screenshot shows a new table that was created by clicking the second (New Item) icon in the Data window. Three columns have been added to it, setting the properties using the controls in the Column Properties section of the dialog:



Querying a Data Source

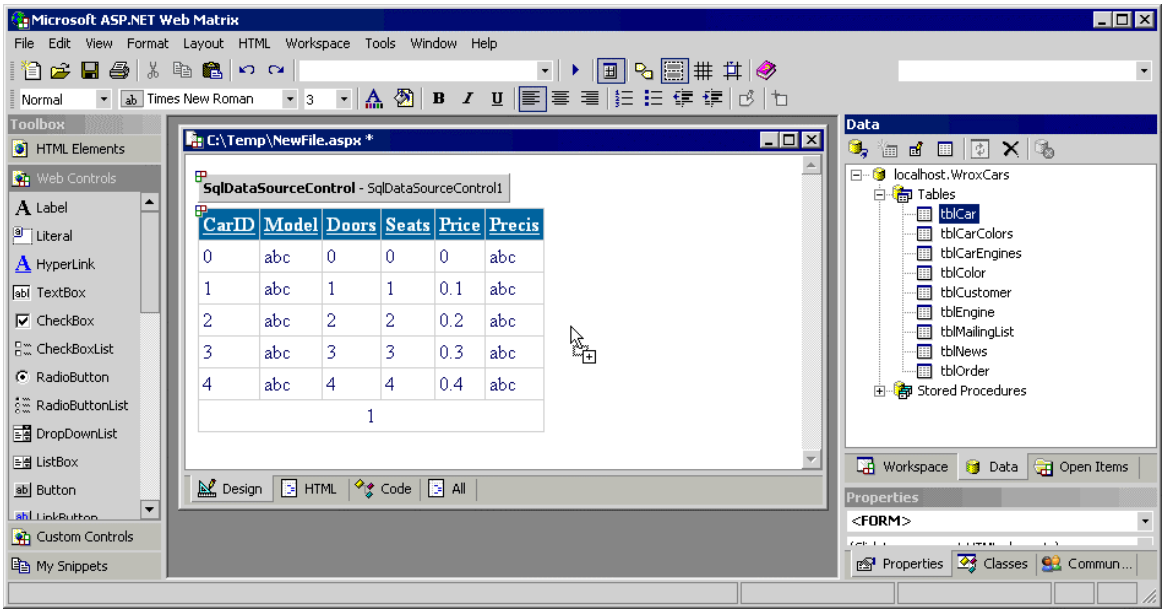
The fourth icon in the Data window is used to query a data source. If you select an existing table in the Data window first, Web Matrix creates a simple SQL statement that will query and display the entire contents of that table. You can, of course, edit the SQL statement to perform more specific queries if required:



The other three icons at the top of the Data window can be used to refresh the contents of the window, delete a table or stored procedure from a database (or even delete an entire database), and disconnect a database from the Data window. This means that the Data window provides a complete environment for working with databases, which is especially useful if you are connecting to MSDE (which, unlike the full version of SQL Server, provides no user interface). Just take care not to select the Delete icon when you only want to disconnect!

Data Tables, Data Controls, and Grid Controls

Data access is at the heart of many web sites and web applications, and Web Matrix makes it easy to perform common tasks that interact with a database. For example, you can drag a table from the list in the Data window and drop it onto a page – whereupon Web Matrix automatically creates a `DataGrid`-based page that displays the data from that table:



If you then switch to Code view, you can see the code that Web Matrix has created within the page (the following is an abridged version):

```
<wmx:SqlDataSourceControl id="SqlDataSourceControl1" runat="server"
  UpdateCommand="UPDATE [tblCar] SET [Model]=@Model,[Doors]=@Doors,[Seats]=@Seats,
                [Price]=@Price,[Precis]=@Precis WHERE [CarID]=@CarID"
  SelectCommand="SELECT * FROM [tblCar]" AutoGenerateUpdateCommand="False"
  ConnectionString="server='localhost'; trusted_connection=true; Database='WroxCars'"
  DeleteCommand="">
</wmx:SqlDataSourceControl>
<wmx:MxDataGrid id="MxDataGrid1" runat="server"
DataSourceControlID="SqlDataSourceControl1"
     BorderColor="#CCCCCC" AllowSorting="True" AutoGenerateFields="False"
DataMember="tblCar"
     AllowPaging="True" BackColor="White" CellPadding="3" DataKeyField="CarID"
     BorderWidth="1px" BorderStyle="None">
  <PagerStyle horizontalalign="Center"
forecolor="#000066" backcolor="White"
             mode="NumericPages"></PagerStyle>
  <FooterStyle forecolor="#000066" backcolor="White"></FooterStyle>
  <SelectedItemStyle font-bold="True" forecolor="White"
                     backcolor="#669999"></SelectedItemStyle>
  <ItemStyle forecolor="#000066"></ItemStyle>
  <Fields>
    <wmx:BoundField DataField="CarID" SortExpression="CarID"
         HeaderText="CarID"></wmx:BoundField>
    <wmx:BoundField DataField="Model" SortExpression="Model"
         HeaderText="Model"></wmx:BoundField>
    <wmx:BoundField DataField="Doors" SortExpression="Doors"
         HeaderText="Doors"></wmx:BoundField>
```

```
      <wmx:BoundField DataField="Seats" SortExpression="Seats"
           HeaderText="Seats"></wmx:BoundField>
      <wmx:BoundField DataField="Price" SortExpression="Price"
           HeaderText="Price"></wmx:BoundField>
      <wmx:BoundField DataField="Precis" SortExpression="Precis"
           HeaderText="Precis"></wmx:BoundField>
    </Fields>
    <HeaderStyle font-bold="True" forecolor="White" backcolor="#006699"></HeaderStyle>
  </wmx:MxDataGrid>
```

The code uses several new server controls that are specially designed for use in Web Matrix – `MxDataGrid` and the associated `BoundField` elements that create the visible part of the display, as well as `SqlDataSourceControl`, which connects to the data source and exposes the data. We'll look at these in more depth later on.

The "Project" Window (Lower Half)

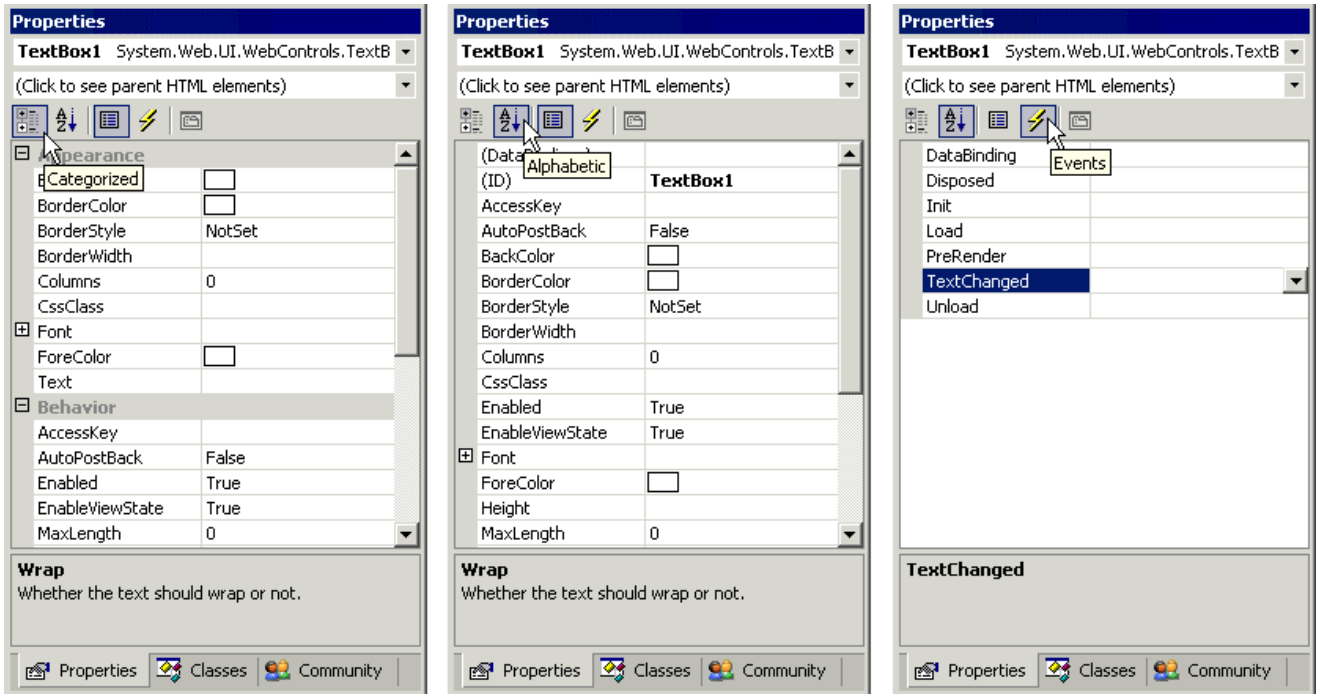
The lower section of the "project" window also provides three main features:

- Properties (and Events) – which allows you to view and edit the properties for the item currently selected in the edit window. For an ASP.NET page or a User Control in Design view, you can select an element (or even the page itself by selecting the <body> element) and access its properties. In all other views, the Properties window shows the ASP.NET-related properties for the document itself. For all other types of file, the Properties window simply displays the path to the document.
- Classes – which provides a list of all the .NET Framework namespaces (displayed either by type or name) that are commonly used in ASP.NET projects. Each namespace can be expanded to show the classes it contains, together with the base class and interfaces that it implements. Double-clicking on any entry in this list opens the Web Matrix Class Browser, which shows details of that item.
- Community – which provides links to useful resources, newsgroups, list servers, and other sites that provide support and references materials for Web Matrix.

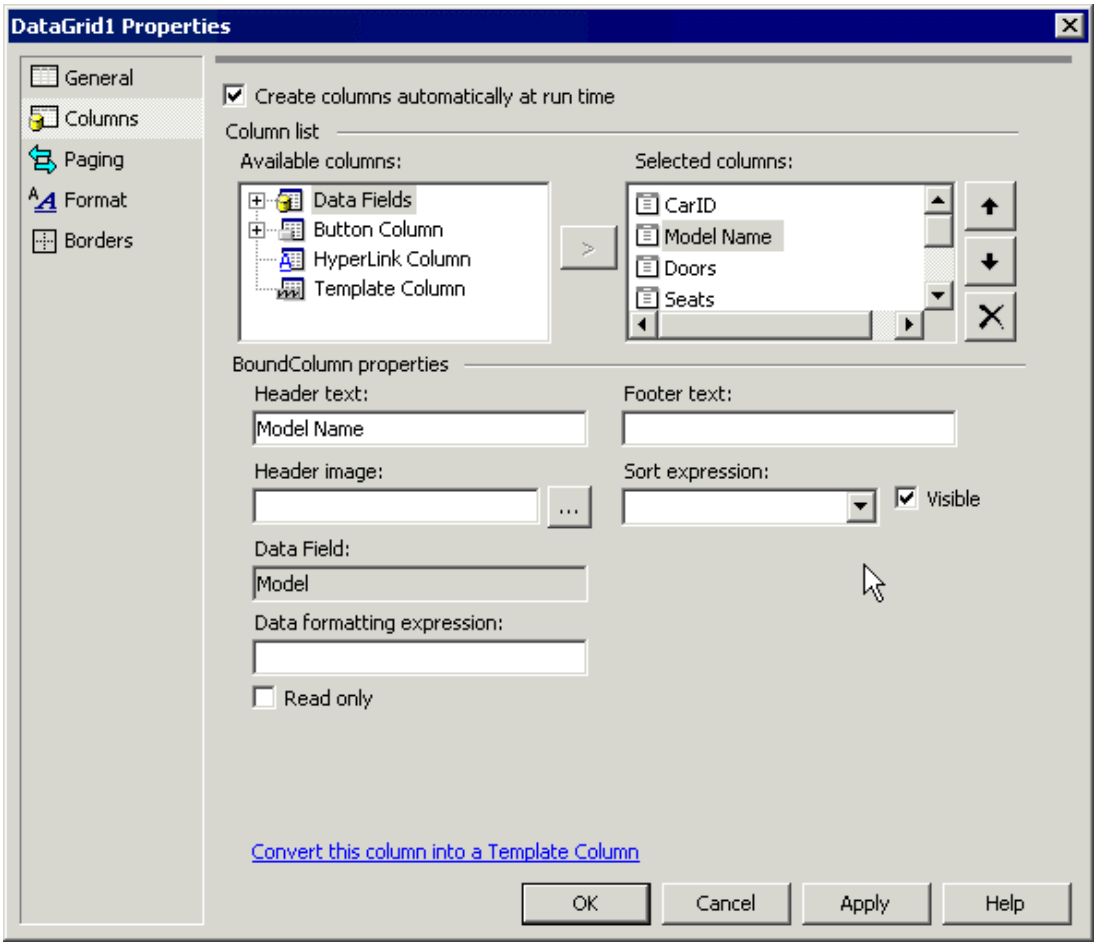
The Properties Window

The Properties window provides three views of the properties and events for an element or other object. The default view is Categorized (as shown in the first of the following three screenshots), which displays the properties within expandable categories such as Appearance, Data, and Layout. The icons in the Properties window allow you to switch between this view and Alphabetic view, which is useful if you are familiar with the property names for the selected object. In both cases, property values can be set by typing in a new value, selecting it from a list of predefined values (such as True or False), or through a "picker" (for example, for selecting a color) or some other dialog.

There is also an Events icon in the Properties window, which you click to switch the view to show the server-side ASP.NET events that are available for the selected object. Double-clicking on an event name automatically inserts the code for an empty event handler in the page, displays the page in Code view, and positions the cursor ready for you to type the code for the event. You can also use the drop-down list for each event to connect it to an existing event handler in the page. We'll see this process in action in *Part 2*.



Clicking the fifth (and final) icon opens any Property Pages associated with the selected control. For example, with a `DataGrid` control, a dialog that allows the templates for the control to be defined is displayed:



When the currently selected control is a `DataGrid` or a `DataList`, the lower section of the Properties window contains two hyperlinks. The second one, Property Builder, opens the "property page" dialog shown in the previous screenshot. The other link, Auto-Format, provides a list of pre-defined colors and styles for the grid or list, and for its content. The style can be applied simply by selecting it in the Auto Format dialog: