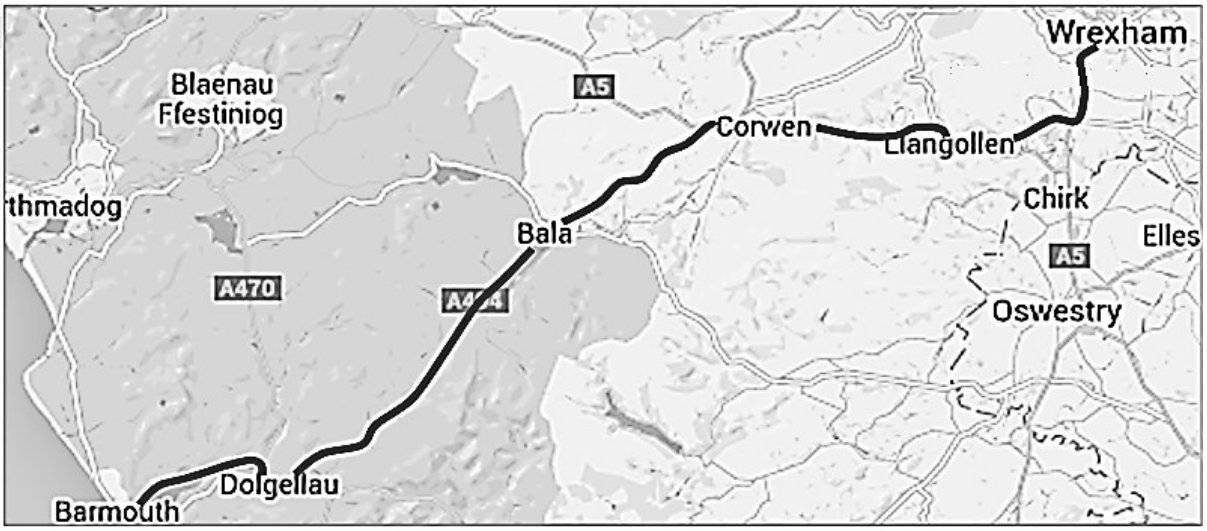
4 Bus timetable

In the next project we will produce a timetable information system for the bus service between Barmouth and Wrexham, which calls on route at the towns of Dolgellau, Bala and Llangollen.



We will set up a database to provide the timetables for three services in each direction along the route:

Wrexham	Llangollen	Bala	Dolgellau	Barmouth
0910	0940	1035	1120	1140
1310	1340	1435	1510	1540
1945	2015	2110	2145	2205

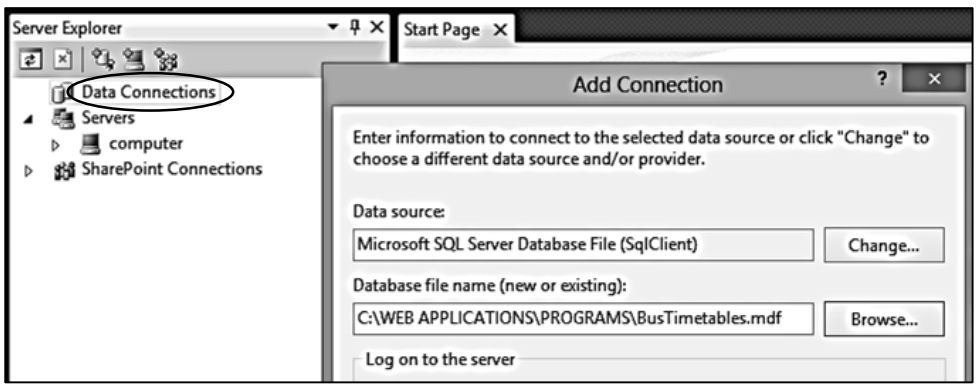
Barmouth	Dolgellau	Bala	Llangollen	Wrexham
0950	1020	1055	1150	1220
1230	1250	1325	1420	1450
1920	1940	2015	2110	2140

Users will be asked to provide the starting point and destination for their journey, along with either:

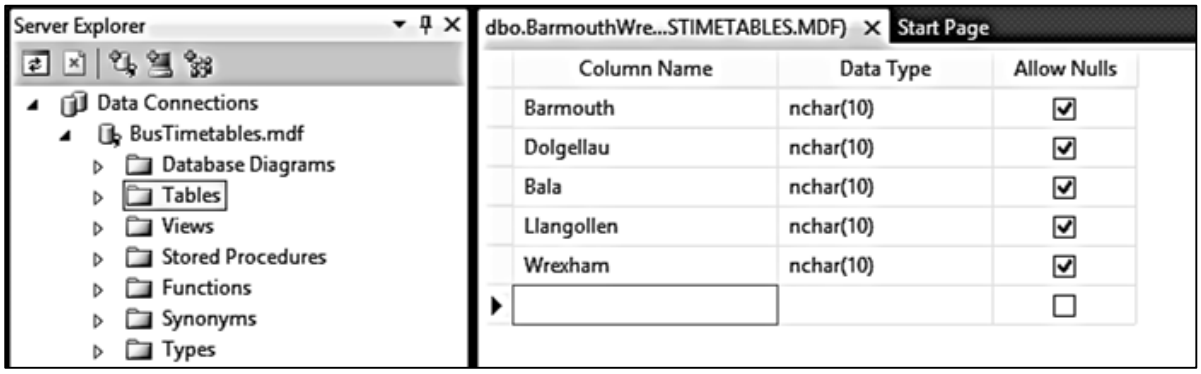
the earliest time at which they can depart, or
the latest time by which they must arrive.

The computer will then check the database for the most suitable departure and arrival times, or give a warning message if there is no scheduled journey meeting the user's requirements.

We begin by creating a database. In Visual Studio, open the ***Server Explorer*** window by selecting the Server Explorer option from the ***View*** menu. Right click ***Data Connections***, and select ***Add Connection***. Create a new database file '***BusTimetables'.***

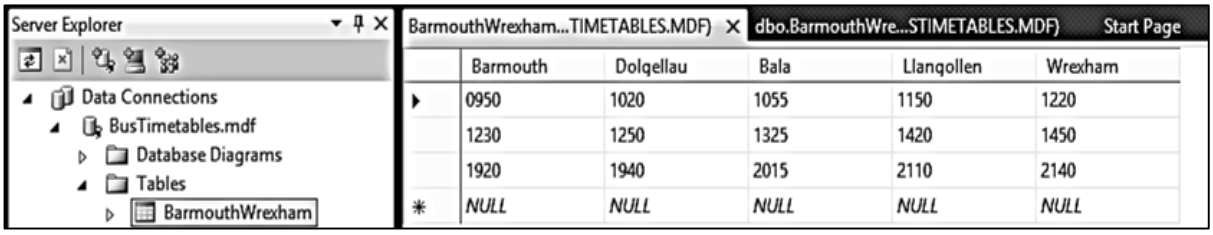


Open the contents list for ***BusTimetables*** and right click ***Tables***. Select ***Add New Table***. We will create a table to store times for journeys from Barmouth to Wrexham. Set up the fields shown:

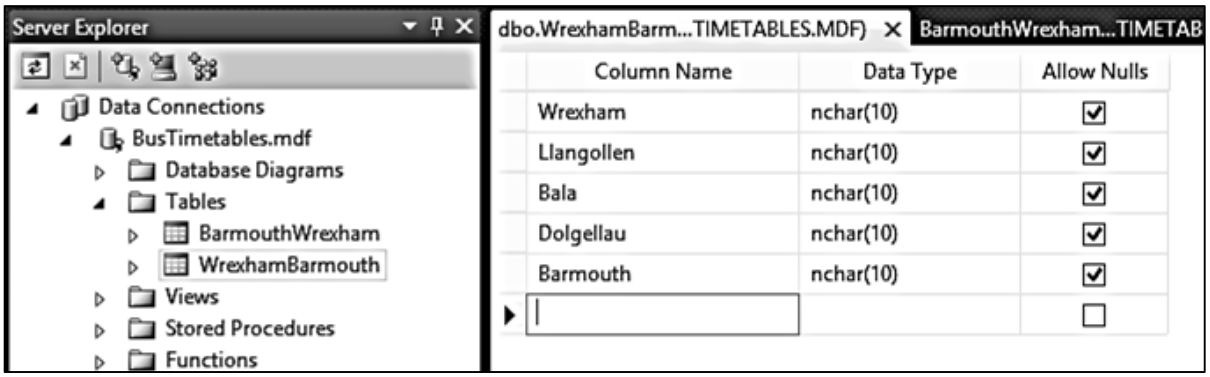


Close the table by clicking the cross at the top right of the tab. When prompted, give the table the name '***BarmouthWrexham***'.

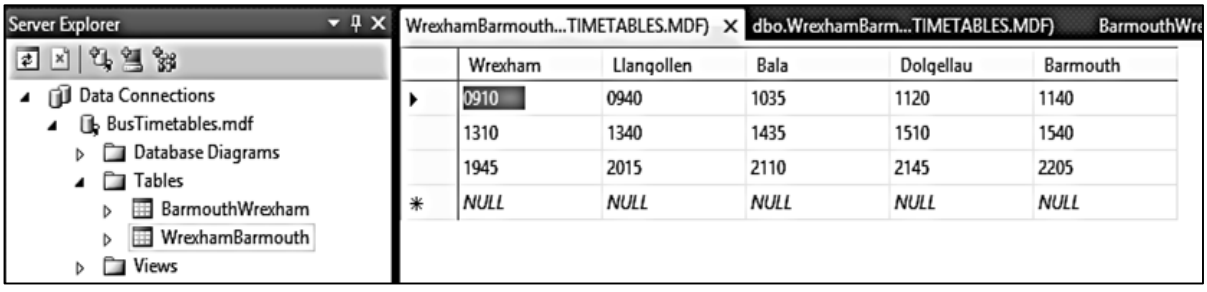
Right click the ***BarmouthWrexham*** table icon and select 'Show Table Data'. Add the three journey timetables:



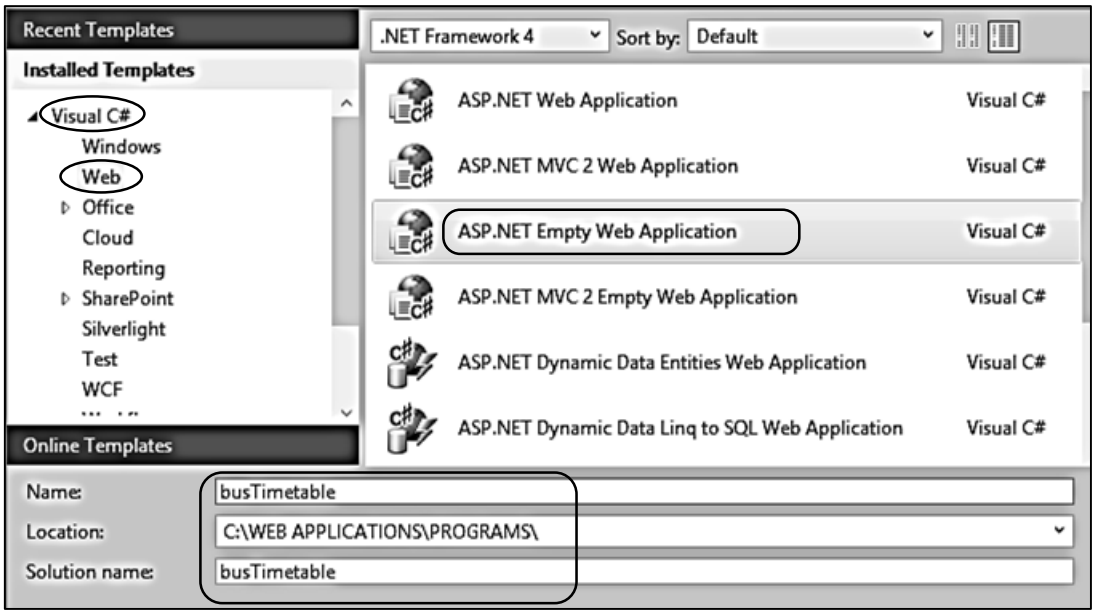
In a similar way, set up a table for journeys in the other direction from Wrexham to Barmouth. Give this table the name '***WrexhamBarmouth***'.



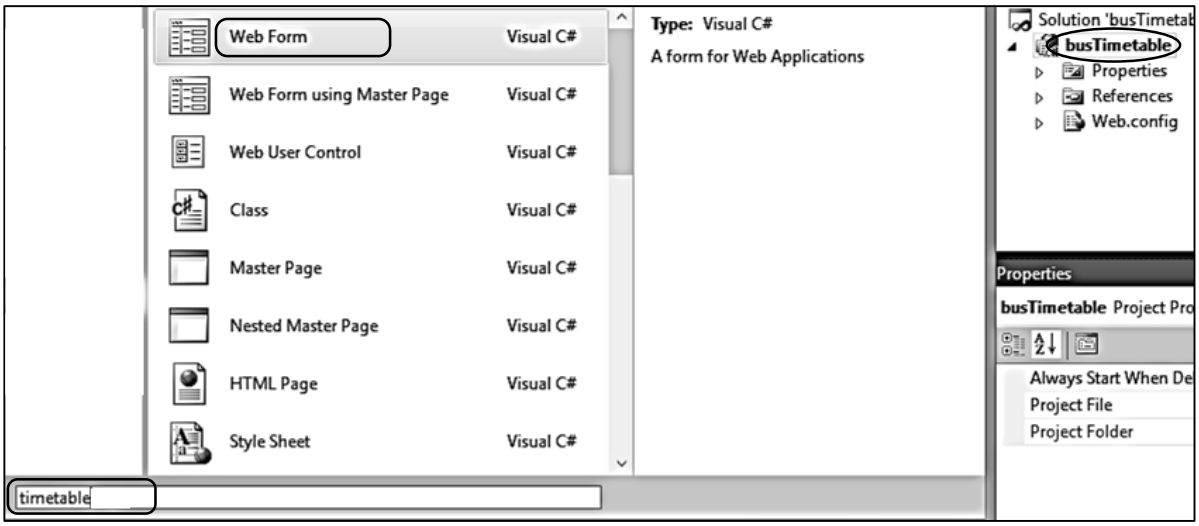
Add the three journey timetables:



Return to the Visual Studio start page by selecting the 'Start page' tab, and click the ***New Project*** option. Select ***Visual C# / Web***, and click '***ASP.NET Empty Web Application***'. Give the name '***busTimetable***'.



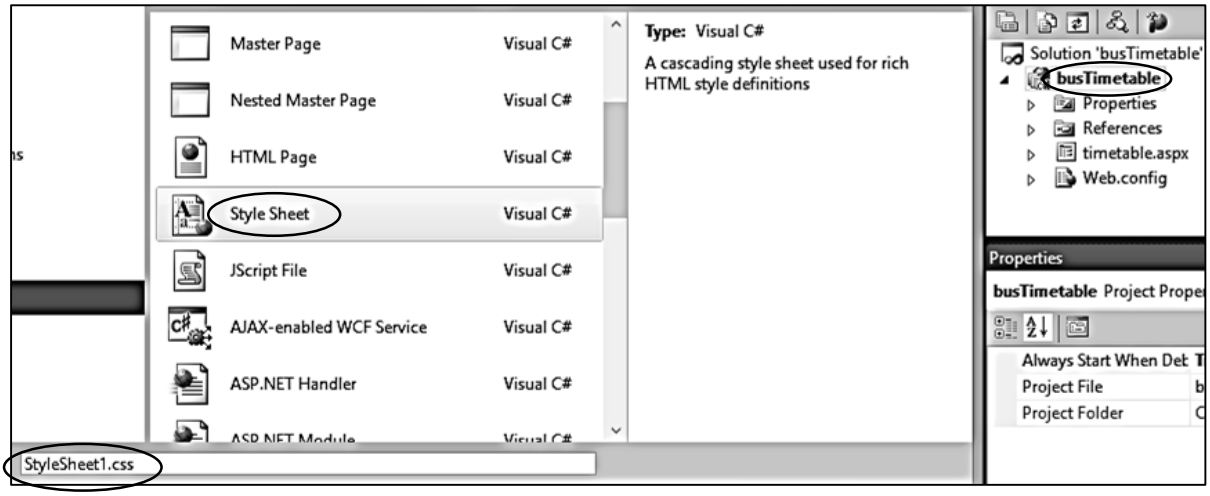
Go to the ***Solution Explorer*** window. Right click the ***busTimetable*** project icon, then select ***Add*** / ***New Item***. Click on ***Web Form***, and give the name '***timetable***'.



Add a title and ***<h1>*** page heading. Give the identification name '***content***' to the *** *** division.

```
<head runat="server">
    <title>Bus Timetable</title>
</head>
<body>
    <form id="form1" runat="server">
    <div id="content">
        <h1>Wrexham - Barmouth Bus Service</h1>
    </div>
    </form>
```

We will add basic formatting to the page using a style sheet. Go to the ***Solution Explorer*** window and right click the ***busTimetable*** project icon. Select ***Add / New Item*** and choose ***Style Sheet***. Accept the name '***StyleSheet1***'.



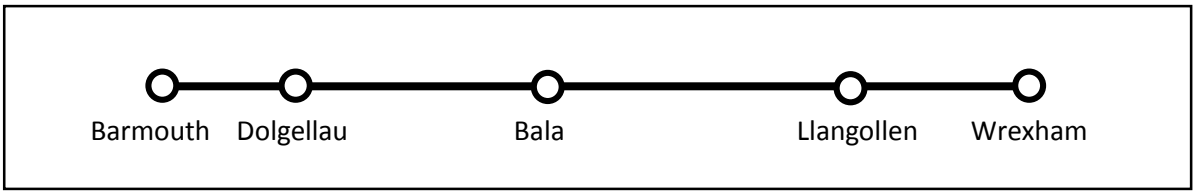
Add code to the syle sheet to set the width of the content area, centre this, and set the font style and size for the heading.

```
body
{
    background: #E9E9E9;
    margin: 0px;
    padding: 0px;
    font-size: .80em;
}

#content
{
  width:1000px;
  background-color: White;
  margin-left: auto;
  margin-right: auto;
  font-family: Arial, Helvetica, sans-serif;
  color: Black;
}

h1
{
    text-align: center;
    font-size: large;
    font-weight:normal;
}
```

Return to the HTML page and add a line of code to the ***<head>*** section to link to the style sheet.

```
<head runat="server">
    <title>Bus Timetable</title>
    <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
```

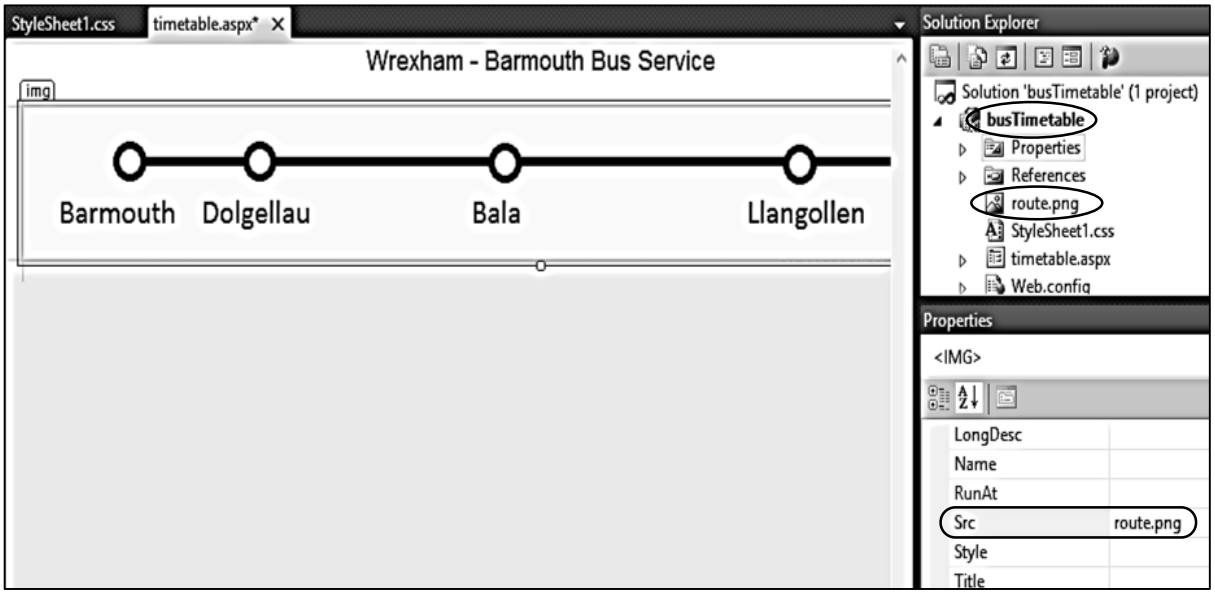
Use Photoshop, Paintshop or a similar graphics application to create a route diagram for the bus service. Save this as the file 'route.png' or 'route.jpg'.



Use the Design button in the bottom left of the program window to go to the screen display. Check that the heading '***Wrexham – Barmouth Bus Service***' is correctly centred.

Right click the ***busTimetable*** project icon and select ***Add / Existing Item***. Locate the ***route*** image and upload this to the project.

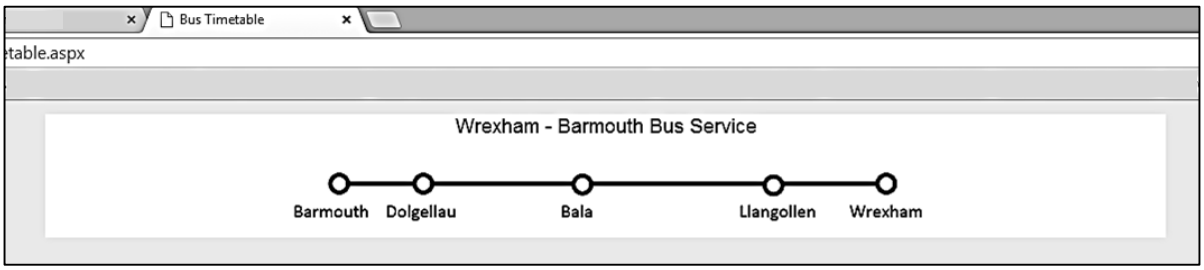
Click after the '***Wrexham – Barmouth Bus Service***' heading and press ***Enter*** to create a new paragraph. Go to the ***Toolbox*** and scroll down to the ***HTML*** component section. Drag the ***Image*** icon to the paragraph area. In the ***Properties*** window, find the ***source (Src)*** property and set this to the name of the ***route*** image file.



Return to the HTML code window. If required, the image can be centred by adding '***align = center***' to the paragraph tag, and the size of the image can be adjusted with a '***width***' value.

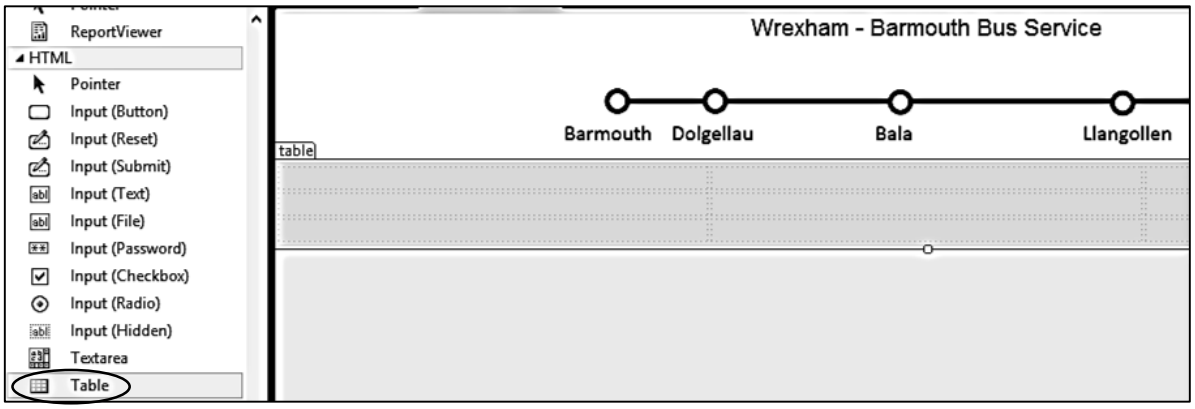
```
<div id="content">
    <h1>Wrexham - Barmouth Bus Service</h1>
    <p align="center">
        <img src="route.png" width="600" /></p>
</div>
```

Build and run the web page. Check that the route map is displayed.



Close the web browser, stop debugging, and return to the ***Design*** view.

Go to the HTML section of the ***Toolbox*** and select the ***Table*** component. Drag and drop at the right-hand end of the image, so that the table is inserted below.



Change to the HTML code page by clicking the ***Source*** button at the bottom left of the window.

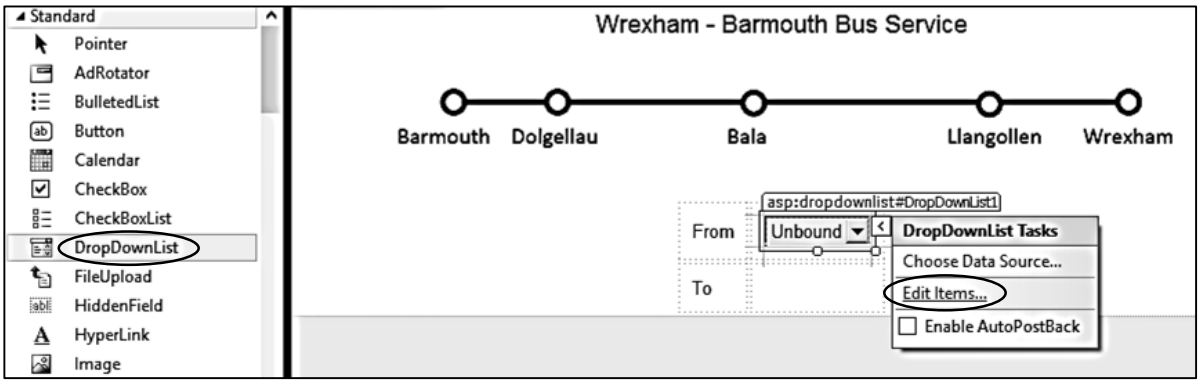
The table will be used to display captions and drop down list boxes where the user can select their departure point and destination. Only two rows of two cells will be needed in the table, so the additional code can be deleted. Also remove the '***width 100%***' value from the ***<table>*** tag.

```
<div id="content">
    <h1>Wrexham - Barmouth Bus Service</h1>
    <p align="center">
        <img src="route.png" width="600" />
    </p>
    <table>
       <tr>
          <td>
              </td>
          <td>
              </td>
       </tr>
       <tr>
          <td>
               </td>
          <td>
               </td>
       </tr>
    </table>
</div>
```

We will format the table. Open the style sheet and add entries for ***table*** and ***td***.

```
h1
{
    text-align: center;
    font-size: large;
    font-weight:normal;
}
table
{
  margin-left: auto;
  margin-right: auto;
  border: none;
}
table td
{
    padding: 10px;
}
```

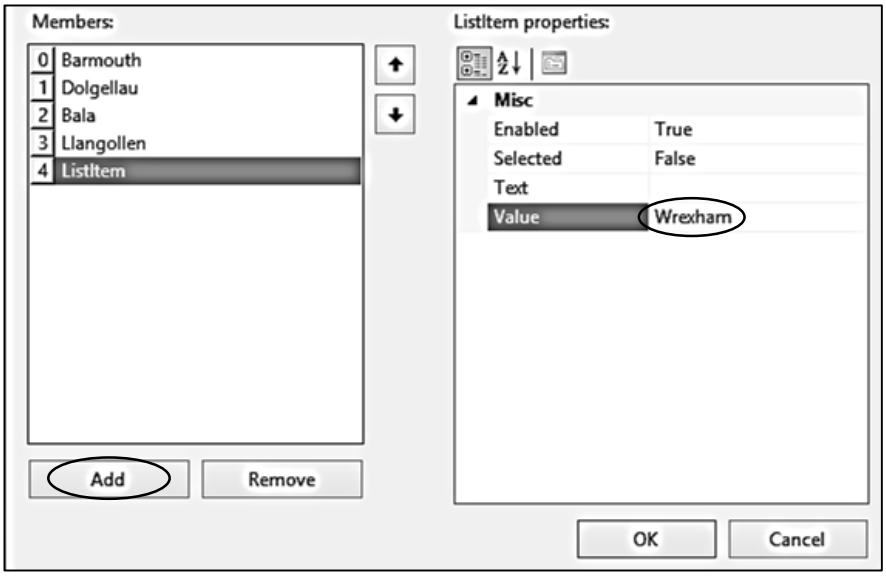
Return to the ***Design*** view of the ***timetable.aspx*** page. Type the captions '***From***' and '***To***' into the first column of the table. Locate the ***DropDownList*** component in the ***Toolbox***, and drag this to the second column of the table alongside the '*From*' caption. Click the small arrow to the right of the DropDownList, and select ***Edit Items***.



Click the ***Add*** button and enter the value '***Barmouth***'.

Click ***Add*** again, and enter '***Dolgellau***'.

Continue for the remaining towns, then click OK to save the list.



Click the ***Source*** button to return to the HTML code. You will see that the computer has inserted the ***DropDownList*** items. Set the ID name of the list to '***listFrom***'.

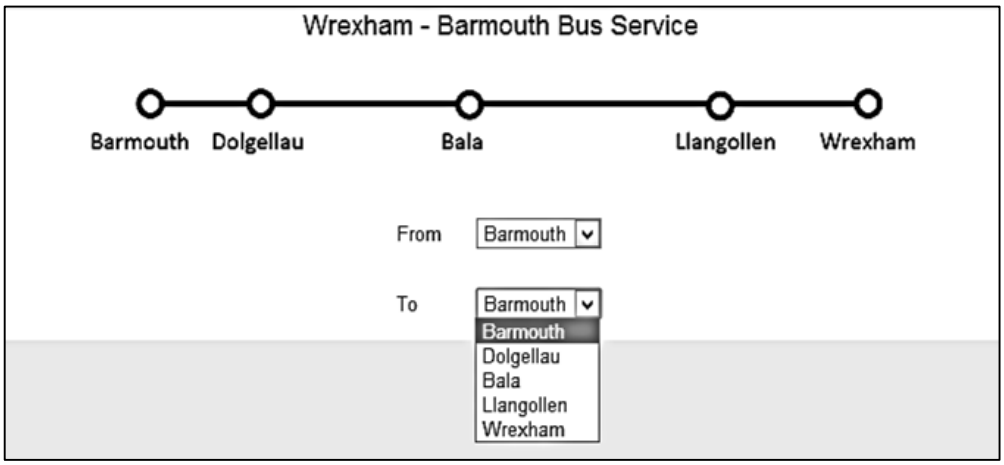
```
<table>
  <tr>
     <td>
         From</td>
     <td>
        <asp:DropDownList ID="listFrom" runat="server">
            <asp:ListItem Value="Barmouth"></asp:ListItem>
            <asp:ListItem Value="Dolgellau"></asp:ListItem>
            <asp:ListItem Value="Bala"></asp:ListItem>
            <asp:ListItem Value="Llangollen"></asp:ListItem>
            <asp:ListItem Value="Wrexham"></asp:ListItem>
        </asp:DropDownList>
      </td>
  </tr>
```

Copy and paste the DropDownList code to the second row of the table. Set the ID name of the second list to '***listTo***'.

```
<table>
    <tr>
       <td>
           From</td>
       <td>
          <asp:DropDownList ID="listFrom" runat="server">
              <asp:ListItem Value="Barmouth"></asp:ListItem>
              <asp:ListItem Value="Dolgellau"></asp:ListItem>
              <asp:ListItem Value="Bala"></asp:ListItem>
              <asp:ListItem Value="Llangollen"></asp:ListItem>
              <asp:ListItem Value="Wrexham"></asp:ListItem>
          </asp:DropDownList>
       </td>
    </tr>
    <tr>
       <td>
           To</td>
       <td>
          <asp:DropDownList ID="listTo" runat="server">
              <asp:ListItem Value="Barmouth"></asp:ListItem>
              <asp:ListItem Value="Dolgellau"></asp:ListItem>
              <asp:ListItem Value="Bala"></asp:ListItem>
              <asp:ListItem Value="Llangollen"></asp:ListItem>
              <asp:ListItem Value="Wrexham"></asp:ListItem>
          </asp:DropDownList>
       </td>
    </tr>
</table>
```

copy

Build and run the web page. Check that the drop down lists show the set of town names correctly.



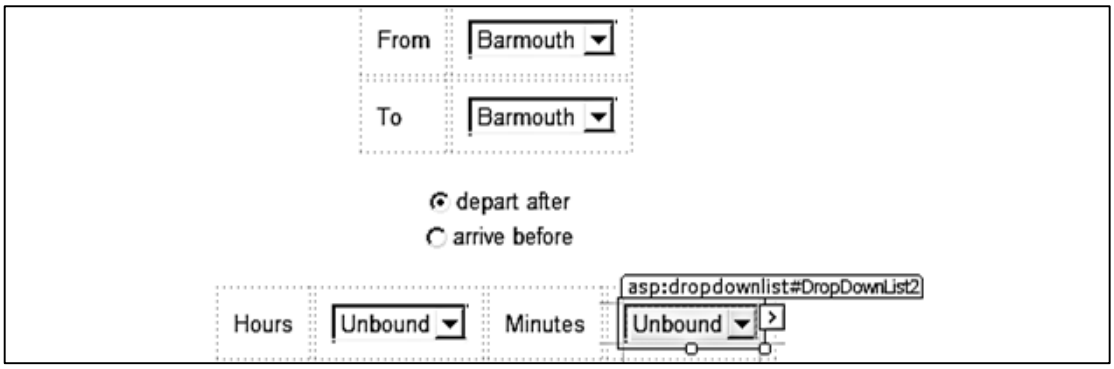
Return to the HTML page, stop debugging, and add code to produce two ***RadioButtons***. These will allow the user to select either a departure time or arrival time for their journey enquiry. Notice that the two radio buttons are give the same ***GroupName***; this ensures that only one of the buttons can be selected at a time.

```
                <asp:ListItem Value="Wrexham"></asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>
</table>
<br />
<center>
<asp:RadioButton ID="rbtnAfter" runat="server" Text="depart after"
                 Checked="True" GroupName="journeyType" />
<br />
<asp:RadioButton ID="rbtnBefore" runat="server" Text="arrive before"
                 GroupName="journeyType" />
</center>
<br />
</div>
```

We will now add the input boxes for the required departure or arrival time. Create a table with one row and four columns. Enter the caption '***hours***' in the first column, and '***minutes***' in the third column.

```
<asp:RadioButton ID="rbtnBefore" runat="server" Text="arrive before"
                 GroupName="journeyType" />
</center>
<br />
<table>
   <tr>
      <td>Hours</td>
      <td></td>
      <td>Minutes</td>
      <td></td>
   </tr>
</table>
```

Change to the page layout view by clicking the ***Design*** button. Select the ***DropDownList*** component and drag this to each of the two empty cells of the table.



Click the ***Source*** button to view the HTML code and give the **ID** names '***listHour***' and '***listMins***' to the drop down list boxes. Add hour and minute values for the drop down lists. You can save time by using ***copy*** and ***paste*** to add the items, then edit the numbers as required.

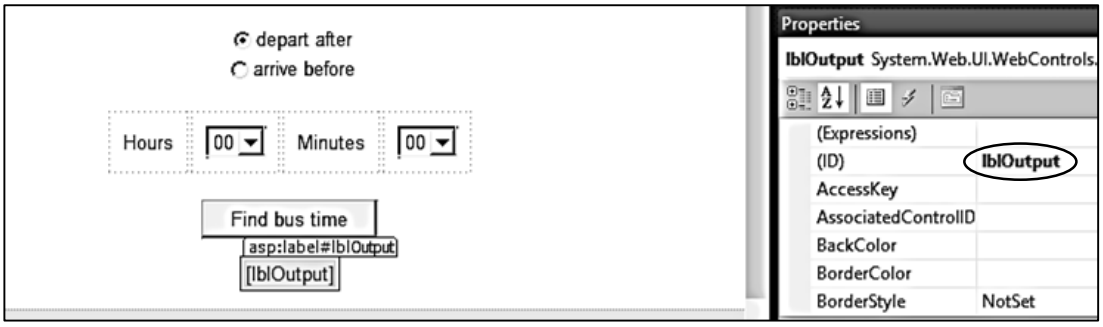
```
<table>
   <tr>
      <td>Hours</td>
      <td>
         <asp:DropDownList ID="listHour" runat="server">
             <asp:ListItem>00</asp:ListItem>
             <asp:ListItem>01</asp:ListItem>
             <asp:ListItem>02</asp:ListItem>
             <asp:ListItem>03</asp:ListItem>
             <asp:ListItem>04</asp:ListItem>
             <asp:ListItem>05</asp:ListItem>
             <asp:ListItem>06</asp:ListItem>
                                        continue copying for
                                        each hour: 07 to 18
             <asp:ListItem>19</asp:ListItem>
             <asp:ListItem>20</asp:ListItem>
             <asp:ListItem>21</asp:ListItem>
             <asp:ListItem>22</asp:ListItem>
             <asp:ListItem>23</asp:ListItem>
         </asp:DropDownList>
      </td>
      <td>
         Minutes </td>
      <td>
         <asp:DropDownList ID="listMins" runat="server">
             <asp:ListItem>00</asp:ListItem>
             <asp:ListItem>15</asp:ListItem>
             <asp:ListItem>30</asp:ListItem>
             <asp:ListItem>45</asp:ListItem>
         </asp:DropDownList>
      </td>
   </tr>
</table>
```

Build and run the web page to check that the drop down lists operate correctly.

Return to the HTML page and stop debugging. Add a button which will call a method to find the most suitable bus time.

```
                <asp:ListItem>30</asp:ListItem>
                <asp:ListItem>45</asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>
</table>
<br />
<center>
    <asp:Button ID="btnGetTime" runat="server" Text="Find bus time" />
</center>
<br />
</div>
```

Use the ***Design*** button to select the page layout view. Press the ***Enter*** key after the button to move down the page, then add a ***Label*** component from the toolbox. In the ***Properties*** window, delete the text from the label and change its name to ***'lblOutput'***. We will use this label as a place holder where the program can output the bus time for the user.



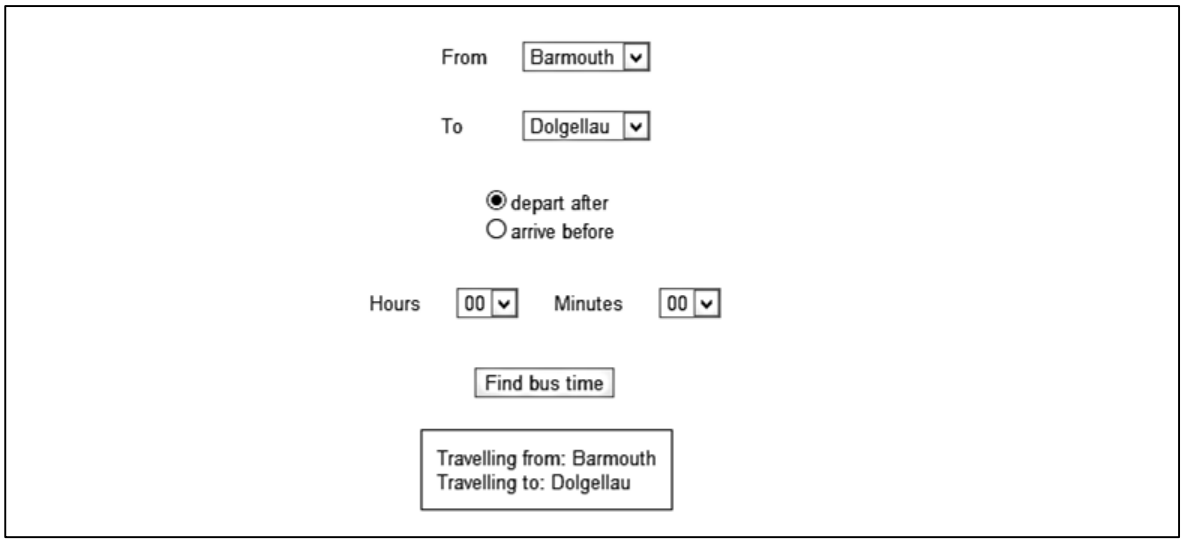
Double click the '***Find bus time***' button to create a C# ***button_click( )*** method.

We will add lines to the method to create a string of HTML code which will display information on the web page. Begin by setting up an empty string ***s***. We can add commands to this string by means of the '***+=***' operator. Once the set of HTML commands is complete, this is transferred to the ***Label Text*** property.

```
protected void btnGetTime_Click(object sender, EventArgs e)
 {
    string s = "";

    s += "<table border=1 cellpadding=10><tr><td>";
    s += "Travelling from: ";
    s += Convert.ToString(listFrom.SelectedItem);
    s += "<br />";
    s += "Travelling to: ";
    s += Convert.ToString(listTo.SelectedItem);

    s += "</table>";
    lblOutput.Text = s;
 }
```

Build and run the web page. Select a departure and destination town, then click the '***Find bus time***' button.

We have created an output table, which echoes the town names as a means of checking that the program is working correctly so far.



Return to the C# code page and stop debugging.

At this stage we can examine the departure and destination points selected by the user:

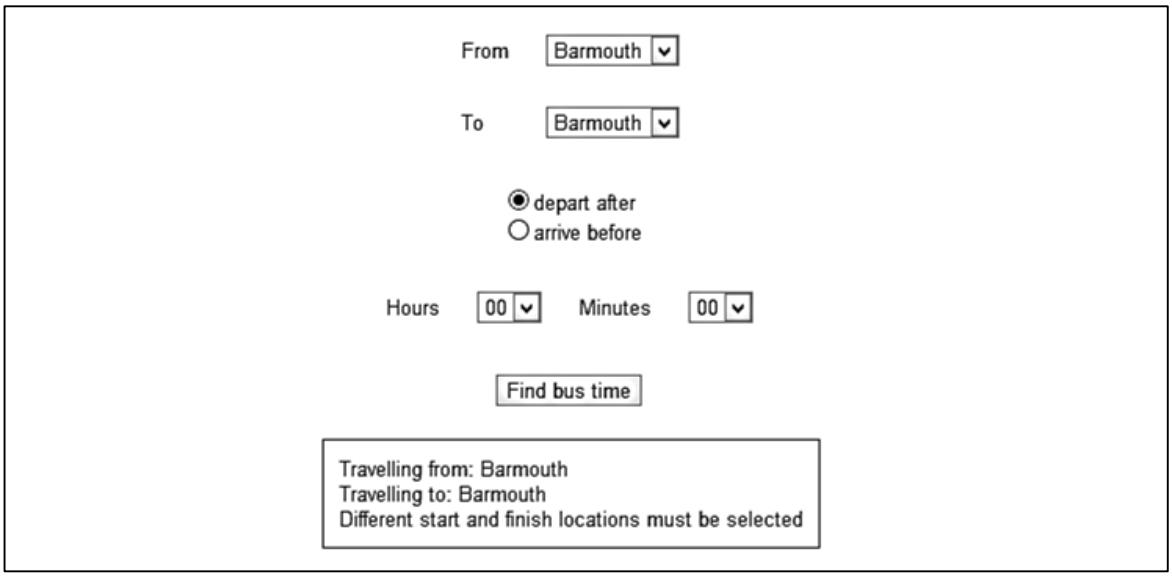
- If both locations are the same, then the user has made an invalid journey request.
- If valid departure and destination points have been entered, then we need to check whether the journey is in an eastwards or westwards direction, so that the correct bus timetable can be selected.

Add code to the ***button_click( )*** method.

```
        s += "Travelling to: ";
        s += Convert.ToString(listTo.SelectedItem);

        int fromIndex = Convert.ToInt16(listFrom.SelectedIndex);
        int toIndex = Convert.ToInt16(listTo.SelectedIndex);
        if (fromIndex == toIndex)
        {
            s += "<br />Different start and finish locations must be selected";
        }

        s += "</table>";
        lblOutput.Text = s;
    }
```

This determines the positions of the selected towns in the drop down lists. If both the departure and destination town have the same list index value, then the user has selected the same departure and destination town and an error message needs to be displayed.

Build and run the web page. Select the same town as the departure and destination point, and check that the error message is displayed.

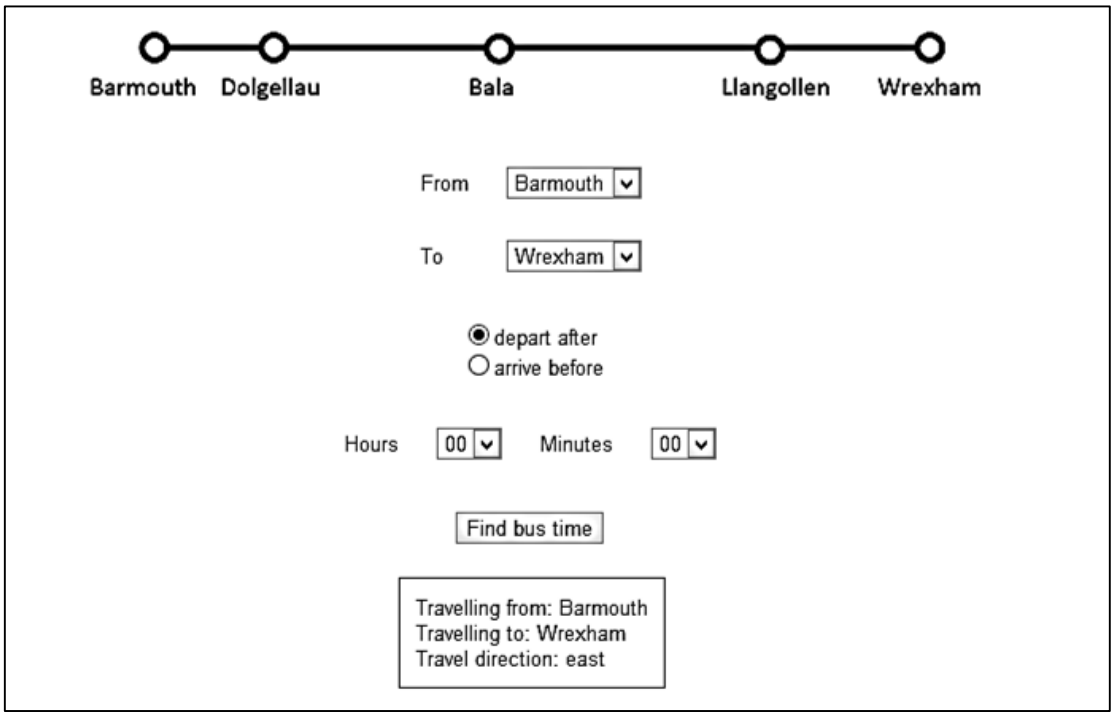


If valid departure and detination towns are selected, we now need to determine whether the journey is eastwards (using the Barmouth to Wrexham timetable) or westwards (using the Wrexham to Barmouth timetable). We can check this by comparing the drop down list index values for the departure and arrival towns. The drop down list index has numbered the towns from west to east, so ***Barmouth = 0*** and ***Wrexham = 4***. The departure point will have a lower index value than the arrival point for an eastwards journey, but a higher index value than the arrival point for a westwards journey.

Return to the C# code page and stop debugging. Add lines to check the journey direction.

```
int fromIndex = Convert.ToInt16(listFrom.SelectedIndex);
int toIndex = Convert.ToInt16(listTo.SelectedIndex);
if (fromIndex == toIndex)
{
    s += "<br />Different start and finish locations must be selected";
}
else
{
    string direction = "";
    if (fromIndex < toIndex)
    {
        direction = "east";
    }
    else
    {
        direction = "west";
    }
    s+="<br />Travel direction: " + direction;
}
s += "</table>";
lblOutput.Text = s;
```

Build and run the web page. Select both ***eastward*** and ***westward*** journeys, and check that the correct travel direction is displayed in each case.



Return to the C# code page. We can now load the correct bus timetable from the database, depending on the journey direction.

Begin by adding '***using Data***' and '***using SqlClient***' directives at the top of the page. Also add the location of the database.

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

using System.Data;
using System.Data.SqlClient;

namespace busTimetable
{
    public partial class timetable : System.Web.UI.Page
    {
        string databaseLocation = "C:\\C#\\BusTimetables.mdf;";

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnGetTime_Click(object sender, EventArgs e)
        {
            string s = "";
```

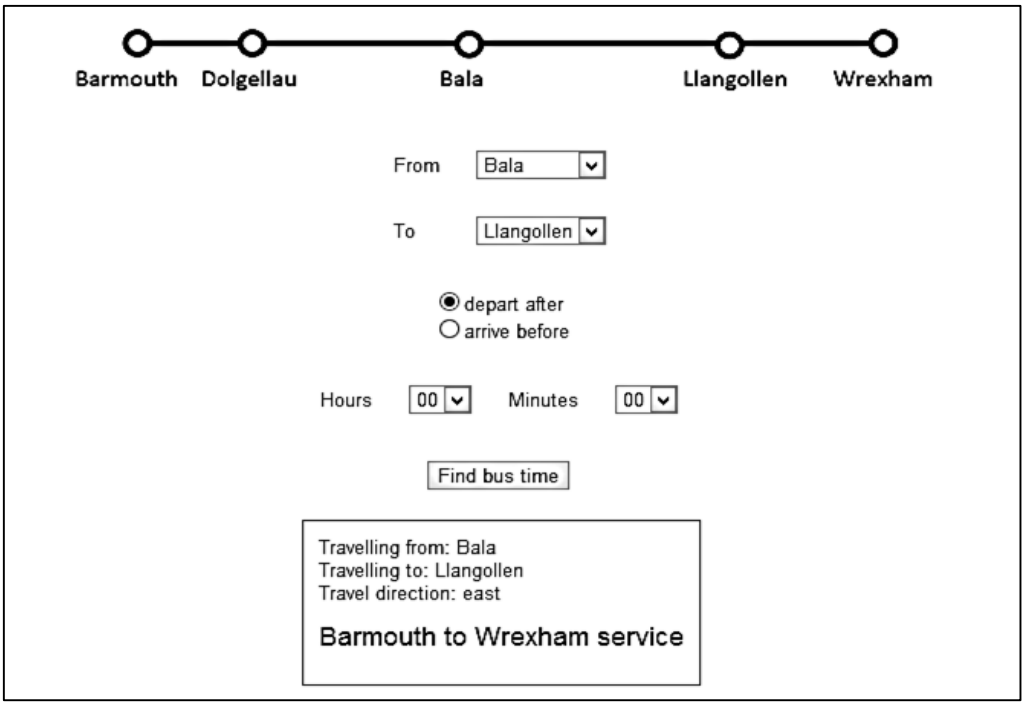
Move down the C# page to the line which outputs the travel direction. Insert code after this line which will open the database and load the required timetable data. A message will also be displayed on the web page to say which timetable is being used. Note: the section of code:

```
SqlConnection cnTB = new SqlConnection(.. ..);
```

should be entered without any line breaks.

```
        direction = "west";
    }
    s+="<br />Travel direction: " + direction;

    DataSet dsBus = new DataSet();
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
         AttachDbFilename="+databaseLocation + "Integrated Security=True;
         Connect Timeout=30; User Instance=True");
    cnTB.Open();
    SqlCommand cmBus = new SqlCommand();
    cmBus.Connection = cnTB;
    cmBus.CommandType = CommandType.Text;
    if (direction == "west")
    {
        cmBus.CommandText = "SELECT * FROM WrexhamBarmouth";
        s += "<h1>Wrexham to Barmouth service</h1>";
    }
    else
    {
        cmBus.CommandText = "SELECT * FROM BarmouthWrexham";
        s += "<h1>Barmouth to Wrexham service</h1>";
    }
    SqlDataAdapter daBus = new SqlDataAdapter(cmBus);
    daBus.Fill(dsBus);
    cnTB.Close();
}
```

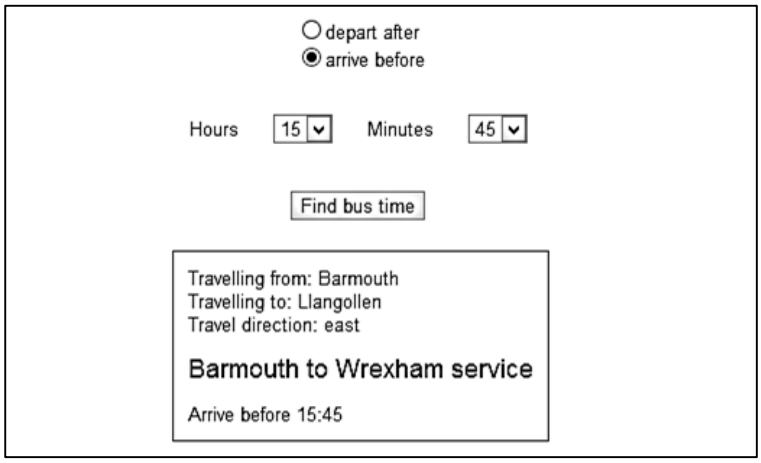
Build and run the web page. Check that the correct bus service is selected for the journey direction.



Return to the C# page and stop debugging. We now add code to obtain the time in ***hours*** and ***minutes*** entered by the user, and whether this represents the required ***departure*** or ***arrival*** time.

```
    SqlDataAdapter daBus = new SqlDataAdapter(cmBus);
    daBus.Fill(dsBus);
    cnTB.Close();

    string hour = Convert.ToString(listHour.SelectedItem);
    string minutes = Convert.ToString(listMins.SelectedItem);
    string journeytype;
    if (rbtnBefore.Checked == true)
    {
        journeytype = "before";
        s += "Arrive before " + hour + ":" + minutes;
    }
    else
    {
        journeytype = "after";
        s += "Depart after " + hour+":"+minutes;
    }

}
s += "</table>";
lblOutput.Text = s;
```

Build and run the web page to check that the required time is displayed.



Return to the C# page and stop debugging. We can now find the most suitable bus departure time. We begin with some preliminary lines of code:

- ***countRecords*** will tell us the number of journeys listed in the bus timetable.
- ***found*** will be set to **true** when a suitable bus time has been found which meets the user's requirements.
- ***busTimeWanted*** is a number version of the time entered by the user, for example: 09:30 becomes 930, 14:00 becomes 1400.
- The bus timetable for westwards journeys has columns showing the towns from Wrexham to Barmouth, which is the opposite order to the drop down lists on the web page. It is therefore necessary to reverse the index values in the case of westward journeys, so that the order of the towns matches the order of the columns in the database table.

```
    else
    {
        journeytype = "after";
        s += "Depart after " + hour+":"+minutes;
    }

    int countRecords = dsBus.Tables[0].Rows.Count;
    Boolean found;
    int busTimeWanted = Convert.ToInt16(hour + minutes);
    if (direction == "west")
    {
        toIndex = 4 - toIndex;
        fromIndex = 4 - fromIndex;
    }

}
s += "</table>";
lblOutput.Text = s;
```

We now add a block of code which will operate only if the user specifies a ***time after which they wish to depart***. We begin by setting the ***found*** variable to '**false**', as a suitable journey time has not yet been identified. The program then carries out a loop in which each journey record from the timetable will be examined in the search for a suitable bus time.

```
        toIndex = 4 - toIndex;
        fromIndex = 4 - fromIndex;
    }

    if (journeytype == "after")
    {
        found = false;
        for (int i = 0; i < countRecords; i++)
        {
            DataRow drBus = dsBus.Tables[0].Rows[i];
        }
    }
}
s += "</table>";
```

The code to check for the most suitable bus time can now be added.

This begins by accessing the current bus journey record and selecting the bus times at the departure point and the destination. These are stored as the string variables ***BusDepartTime*** and ***BusArriveTime***. Since the user has specified a time after which they wish to depart, the value of ***BusDepartTime*** is the important piece of data.

We convert ***BusDepartTime*** into number format, so that it can be compared with the required departure time, ***BusTimeWanted***.

As soon as a bus departing on or after the required time is identified, the ***found*** variable is set to ***true***. This ensures that no later bus times are considered.

The bus departure and arrival times are added to the output string and will be displayed.

In the event of ***no buses*** departing on or after the time specified by the user, the ***found*** variable will still be set to **false** when all records have been checked. In this case, an information message will be displayed.

```
if (journeytype == "after")
{
    found = false;
    for (int i = 0; i < countRecords; i++)
    {
        DataRow drBus = dsBus.Tables[0].Rows[i];

        string BusDepartTime = Convert.ToString(drBus[fromIndex]);
        string BusArriveTime = Convert.ToString(drBus[toIndex]);
        int busTimeFound = Convert.ToInt16(BusDepartTime);
        if (busTimeFound >= busTimeWanted && found == false)
        {
            found = true;
            s += "<h1>Depart ";
            s += Convert.ToString(listFrom.SelectedItem);
            s += ": " + BusDepartTime + "<br />";
            s += "Arrive ";
            s += Convert.ToString(listTo.SelectedItem);
            s += ": " + BusArriveTime + "</h2>";
        }
    }
    if (found == false)
    {
        s += "<h1>No service available at this time</h1>";
    }

}
```

Build and run the web page. Test the program for a series of journeys, both in an eastwards and westwards direction. In each case select a time after which to depart. Use the timetables on page 72 to check that the program output is correct. Also test the case of a departure time which is too late to catch the last bus.

We can now return to the C# page, stop debugging, and add a similar block of code to handle the case where the user specifies a time by which they wish to arrive at their destination.

```
            s += "<h1>No service available at this time</h1>";
        }
    }
    else
    {
        found = false;
        for (int i = countRecords-1; i>=0 ; i--)
        {
            DataRow drBus = dsBus.Tables[0].Rows[i];
            string BusDepartTime = Convert.ToString(drBus[fromIndex]);
            string BusArriveTime = Convert.ToString(drBus[toIndex]);
            int busTimeFound = Convert.ToInt16(BusArriveTime);
            if (busTimeFound <= busTimeWanted && found == false)
            {
                found = true;
                s += "<h1>Depart ";
                s += Convert.ToString(listFrom.SelectedItem);
                s += ": " + BusDepartTime + "<br />";
                s += "Arrive ";
                s += Convert.ToString(listTo.SelectedItem);
                s += ": " + BusArriveTime + "</h2>";
            }
        }
        if (found == false)
        {
            s += "No service available at this time<br>";
        }
    }
}
s += "</table>";
```

Notice that in this case the important piece of data is the arrival time of the bus, and this is compared to the time specified by the user. We have to modify the loop command:

```
for (int i = countRecords-1; i>=0 ; i--)
```

so that the records are searched in reverse order, beginning with the last bus of the day. This ensures that the user is offered only the latest departure which will get them to their destination on time.

Build and test the completed program. Check the arrival times against the bus timetables give on page 72.