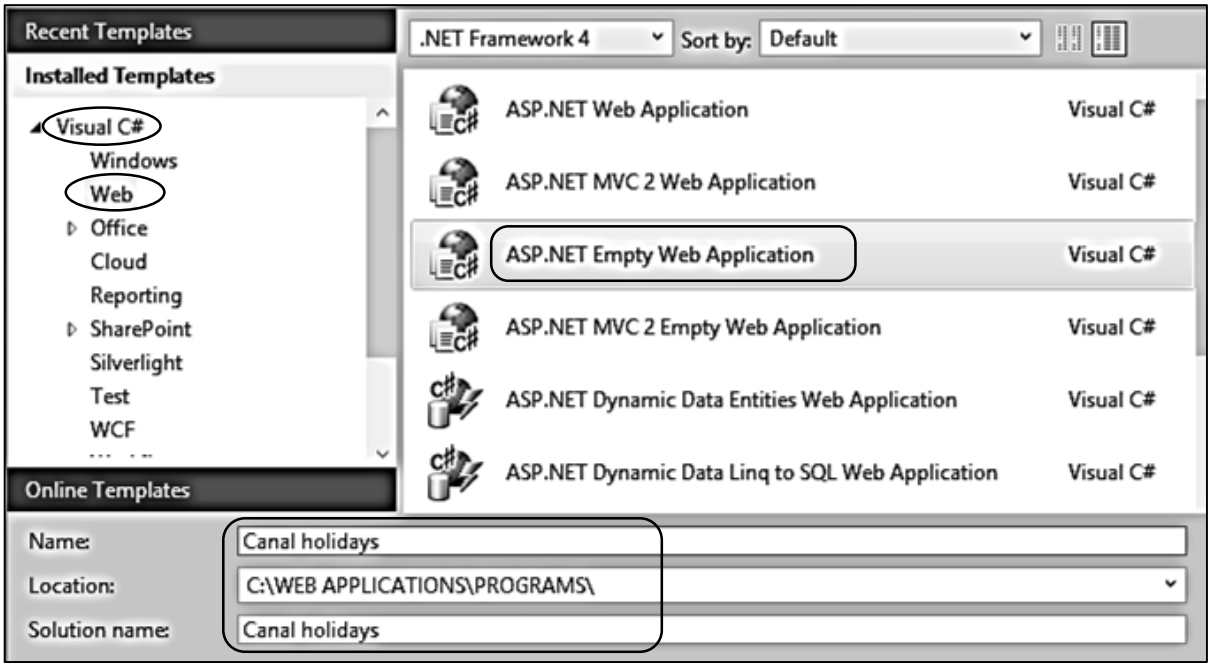
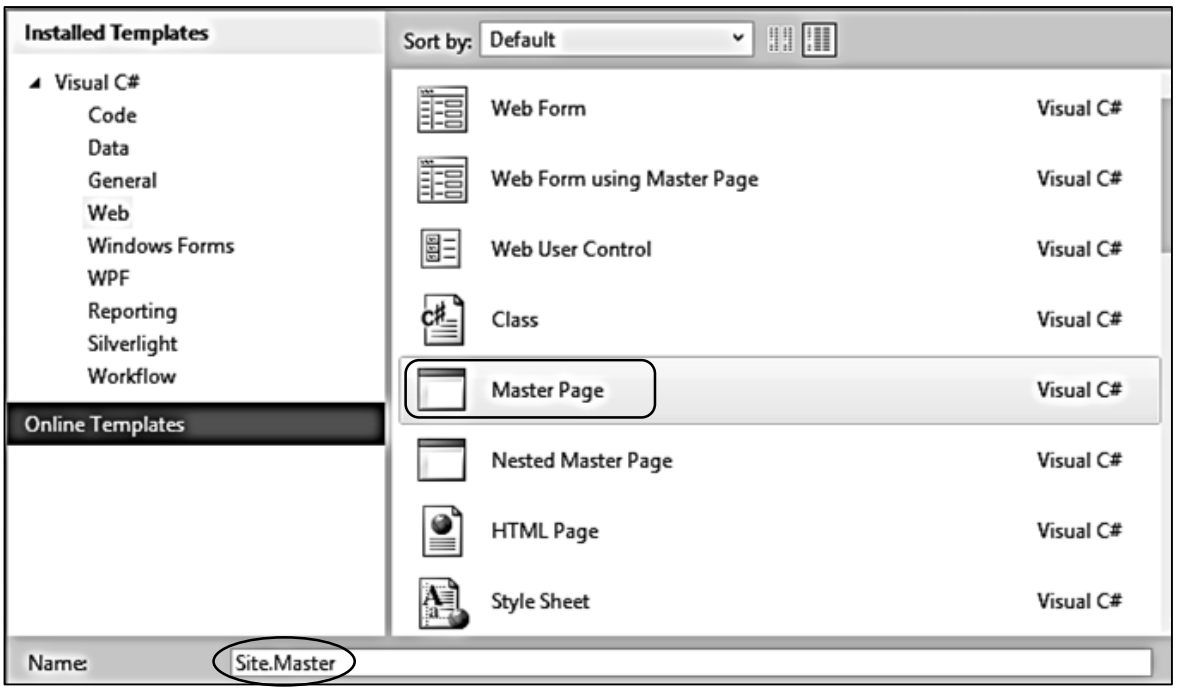
10 Canal Boat Holidays

In this final project, we will make again use of an object oriented approach for data handling , as well as incorporating an interactive calendar component in a similar way to the holiday cottage booking system of chapter 7. We will explore some new graphics techniques for web page screen displays.

Begin by opening ***Microsoft Visual Studio*** and select ***New Project***. Choose ***Visual C# / Web*** as the application type, and click on '***ASP.NET Empty Web Application***'. Give the project name '***Canal holidays***' and select a folder location for the project.

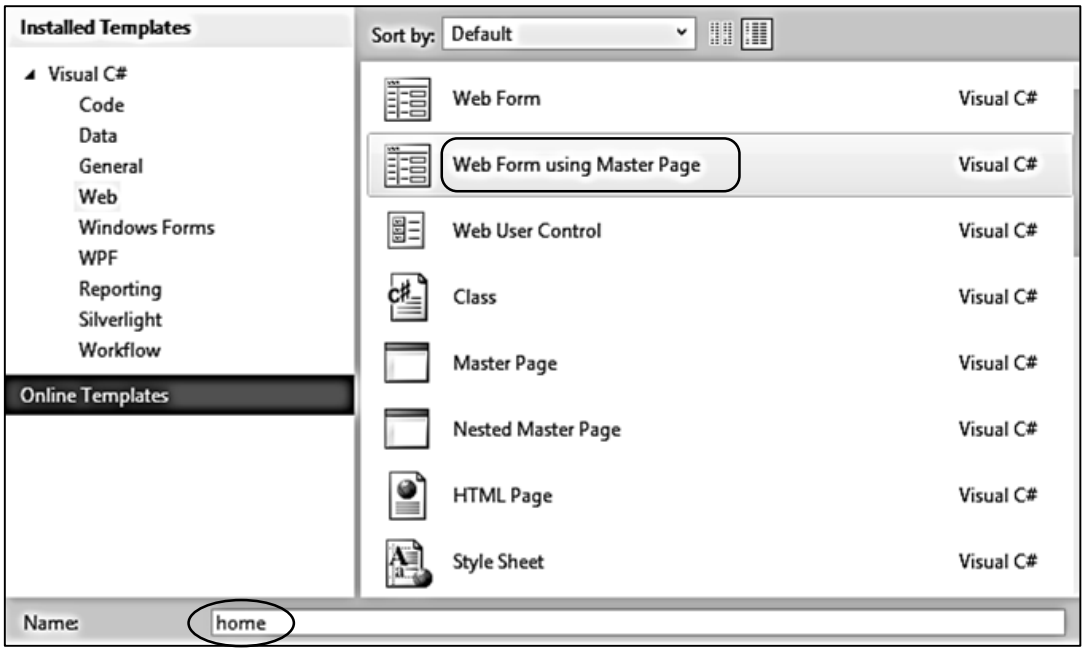


Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Master Page***. Give the name '**Site.Master**'.

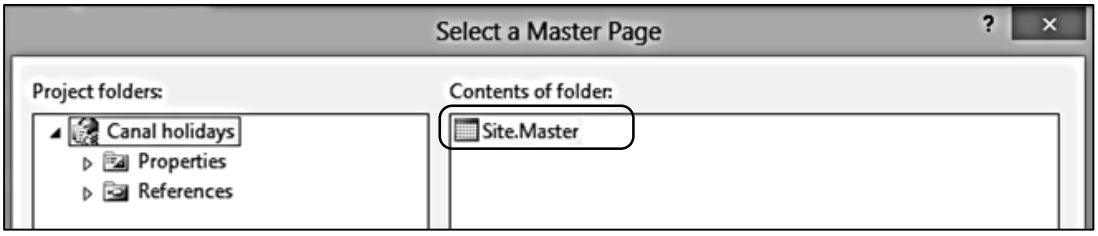


The site master will display the name and logo of the canal boat company and provide the menu system.

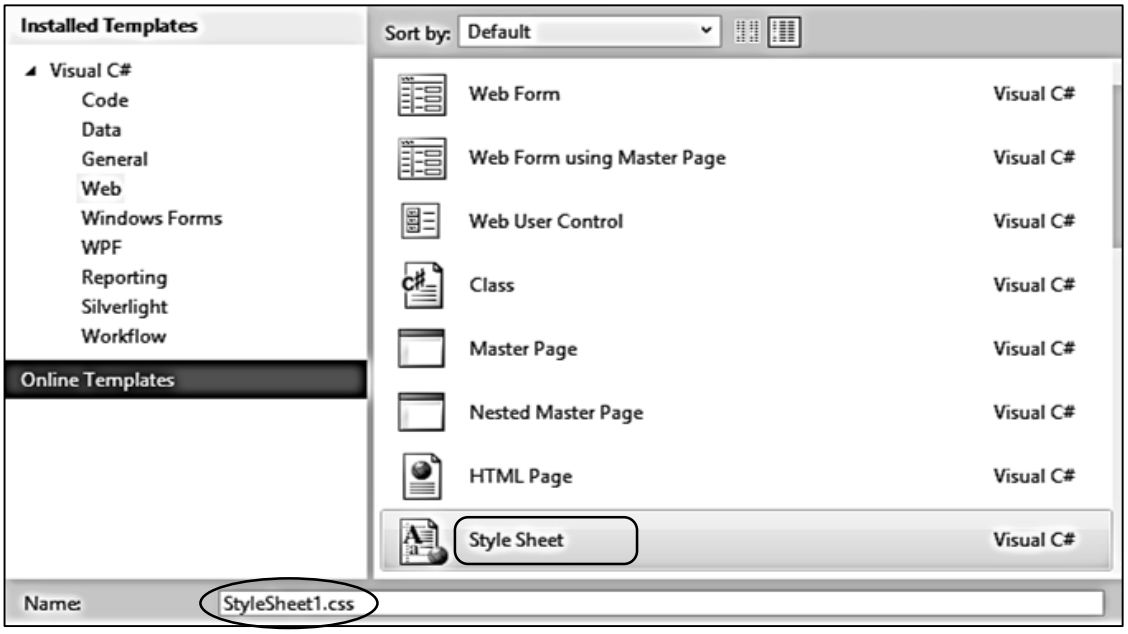
We will create a homepage which uses the site master. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give the name '**home**'.



Accept ***Site.Master*** as the master page and click the '***OK***' button.



We will also create a style sheet. Return to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Style Sheet*** . Accept the name '**StyleSheet1**'.



For the current project, the background screen colour will be dark blue with a central content area in white. Go to the ***StyleSheet1*** file and add sections of formatting code to create the page structure.

```
body
{
    background-color: #103B80;
    font-family: Arial, Helvetica, sans-serif;
    color: black;
    font-size: small;
}

 #header
 {
     background-color: #103B80;
     width: 1000px;
     height: 330px;
 }

 #content
 {
     width: 1000px;
     height: 600px;
     background-color: #FFFFFF;
     padding:20px;
 }
```

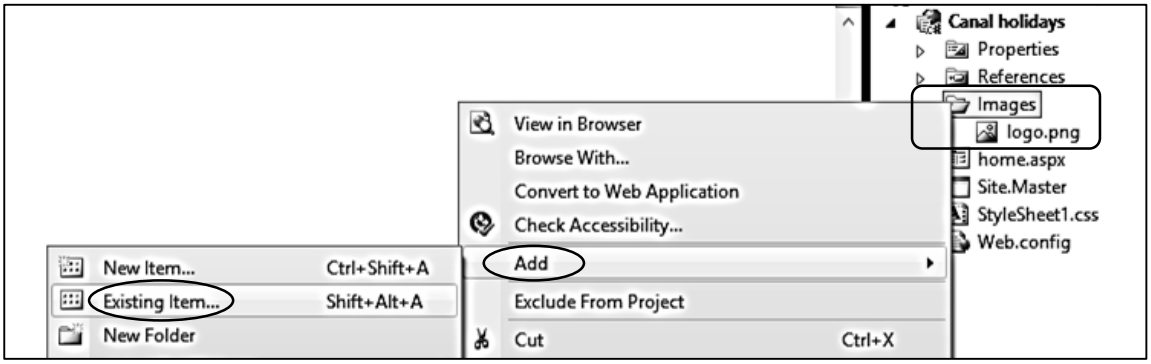
Open the ***Site.Master*** page. Add code to link to the Style Sheet, and to set up divisions.

```
<head runat="server">
  <title>Canal boat holidays</title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
  <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
<body bgcolor="#103B80">
  <center>
    <form id="form1" runat="server">
      <div id="header" >
      </div>
      <div id="content">
          <asp:ContentPlaceHolder ID="ContentPlaceHolder1 " runat="server">
          </asp:ContentPlaceHolder>
      </div>
    </form>
  </center>
</body>
```

Build and run the home page. The basic page structure should appear.

Use a graphics application such as ***Photo Shop*** or ***Paint Shop*** to create a logo and title for the web pages. This should be in white, with a dark blue background to match the web page. A suitable size would be about 500 pixels wide and 75 pixels high. Save the completed image in .PNG or .JPG format with the file name '***logo***' .

Return to Visual Studio, go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / Folder*** and give the name '**Images**'. Click right on the ***Images*** folder icon, select ***Add / Existing item***, and upload the ***logo*** file to the ***Images*** folder.



Return to the ***Site.Master*** file and add a line of code to the 'header' division to display the image.

```
<div id="header" >
  <img src="Images/logo.png" width=500px />
</div>
```

Build and run the home page to check that the logo and title appears correctly.

We will now work on a main menu system for the site. This will be constructed from a series of graphics images which represent tabs for different page options: 'Home', 'Canal route', 'Boats' and 'Booking'. Use any graphics application such as ***Photo Shop*** or ***Paint Shop*** to produce these. Each tab image should be 180 pixels wide by 30 pixels high. Two versions are required for each tab:

- a version in medium blue with white text, to represent the unselected tab
- an orange version with black text, to represent the tab when selected by the mouse pointer.

You may like to use a gradient fill to give a highlight effect. The curved upper corners of the tab images should be in dark blue to match the web page background.

Upload the tab images to the '***Images***' folder in .PNG or .JPG format, using the names:

Bhome, Broute, Bboats, Bbooking for the blue verions,

Ohome,Oroute,Oboats, Obooking for the orange versions.

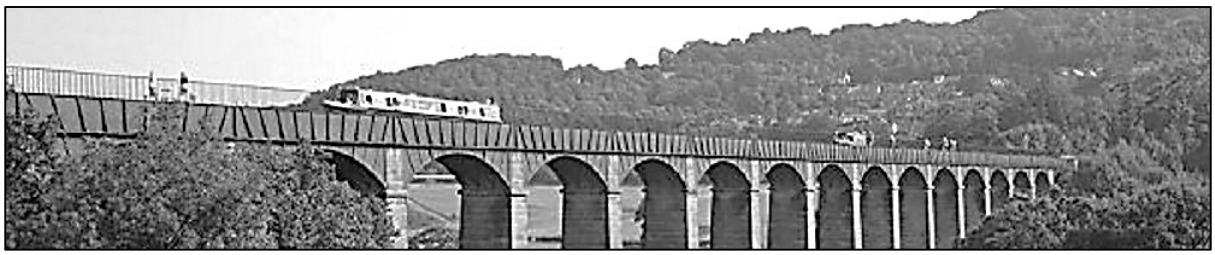
Go to the ***Site.Master*** file and add lines of code to the '***header***' division.

```
<div id="header" >
    <img src="Images/logo.png" width=500px />
    <br />
    <a href="home.aspx">
    <img  src="/Images/Bhome.png" onmouseover="this.src='/Images/Ohome.png'"
                              onmouseout="this.src='/Images/Bhome.png'" /></a>
    <a href="route.aspx" >
    <img  src="/Images/Broute.png" onmouseover="this.src='/Images/Oroute.png'"
                             onmouseout="this.src='/Images/Broute.png'" /></a>
    <a href="boats.aspx" >
    <img  src="/Images/Bboats.png" onmouseover="this.src='/Images/Oboats.png'"
                             onmouseout="this.src='/Images/Bboats.png'" /></a>
    <a href="booking.aspx" >
    <img  src="/Images/Bbooking.png" onmouseover="this.src='/Images/Obooking.png'"
                             onmouseout="this.src='/Images/Bbooking.png'" /></a>
    <br />
</div>
```

Notice that each image command specifies the blue version of the tab when unselected, but reloads the orange version when the mouse pointer moves over the tab area.

Build and run the homepage. Check that the tabs are displayed, and that each tab changes colour as the mouse pointer moves over it.

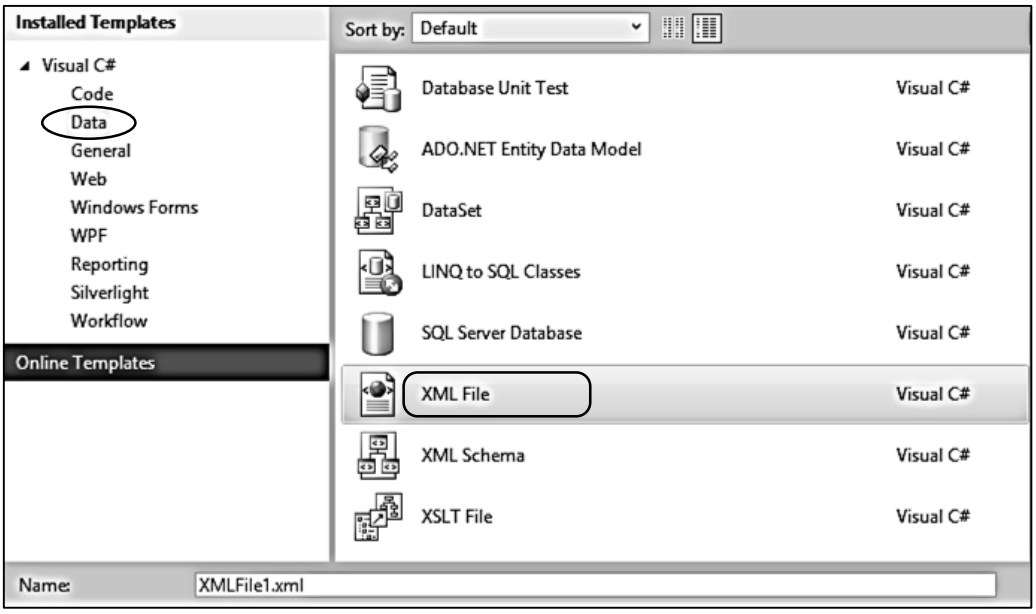
The next feature we will add to our site is a banner with changing images. Canal Boat Holidays operates on the Llangollen Canal in North Wales. Use the Internet to obtain about four suitable images of the Llangollen Canal, and use a graphics application to crop and adjust the image sizes so that each is 1000 pixels wide by 200 pixels high:



Upload your photographs to the '***Images***' folder in .JPG format using the names:

pic0.jpg, pic1.jpg, pic2.jpg, pic3.jpg

The first step in creating an animated banner is to specify the image file names in an XTML document. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item***. Change to the '***Data***' options and choose ***XML File***. Accept the name '**XMLFile1**'.



Open the XML file and add lines of code.

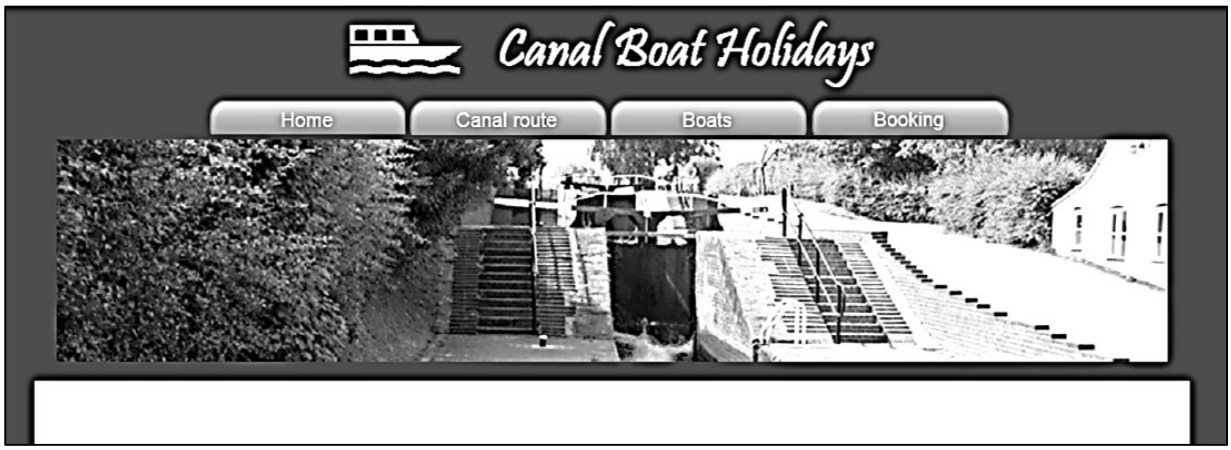
```
<?xml version="1.0" encoding="utf-8" ?>

<Advertisements>
  <Ad>
    <ImageUrl>/Images/pic0.jpg</ImageUrl>
    <Impressions>40</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>/Images/pic1.jpg</ImageUrl>
    <Impressions>40</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>/Images/pic2.jpg</ImageUrl>
    <Impressions>40</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>/Images/pic3.jpg</ImageUrl>
    <Impressions>40</Impressions>
  </Ad>
</Advertisements>
```

The XML file we have created will provide input to an '***AdRotator***' component. This is similar to an image box, but will change the image displayed each time that a web page is reloaded within the master page. The picture image will be randomly selected from the set specified in the XML file list.

Go to the Site.Maser file and add a line of code to create the AdRotator component.

```
    <a href="booking.aspx" >
    <img  src="/Images/Bbooking.png" onmouseover="this.src='/Images/Obooking.png'"
                              onmouseout="this.src='/Images/Bbooking.png'" /></a>
    <br />
     <asp:AdRotator ID="AdRotator1" AdvertisementFile="XMLFile1.xml"
                runat="server" target="_blank" Height="200px" Width="1000px" />
</div>
```

Build and run the home page. One of the banner images will be displayed. Click the '***Home***' tab to reload the page. The image should randomly change.



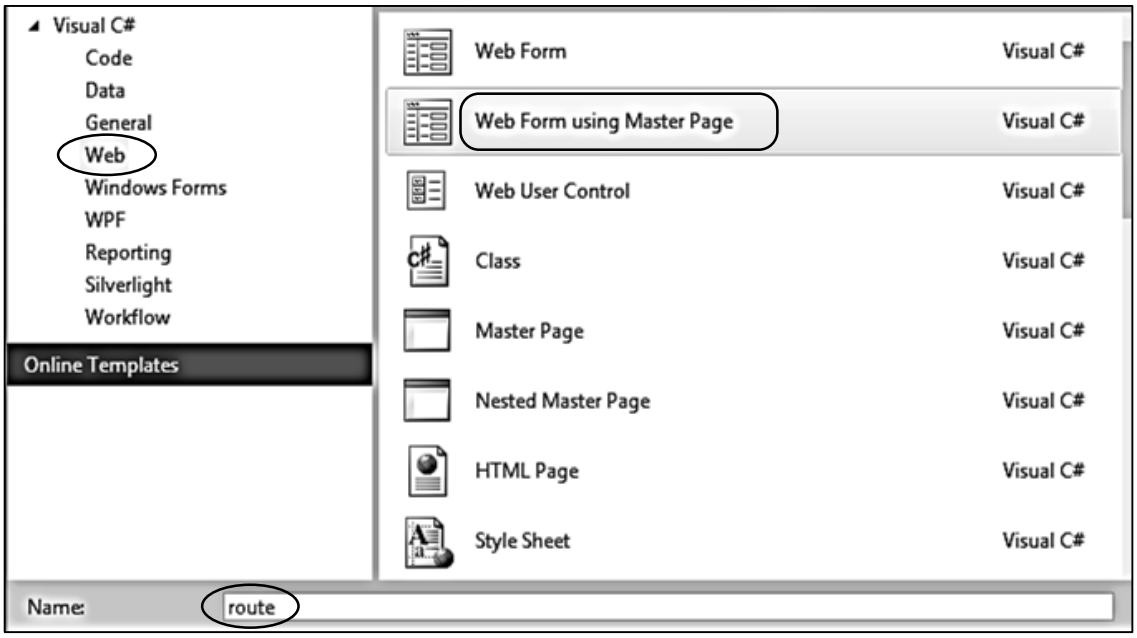
This works, but it would be better if the images changed automatically, for example: every eight seconds. We can easily do this. Return to the Site.Master file, and add lines of code to embed the ***AdRotator*** within an ***UpdatePanel*** component. This forces an update of the page at intervals set by the timer. We have selected a timer interval of 8 seconds, which must be specified in thousands of a second.

```
  <img  src="/Images/Bbooking.png" onmouseover="this.src='/Images/Obooking.png'"
                            onmouseout="this.src='/Images/Bbooking.png'" /></a>
  <br />
  <asp:ScriptManager ID="ScriptManager1" runat="server" />
  <asp:Timer ID="Timer1" Interval="8000" runat="server" />
  <asp:UpdatePanel ID="up1" runat="server">
    <Triggers>
       <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
    </Triggers>
    <ContentTemplate>
          <asp:AdRotator ID="AdRotator1" AdvertisementFile="XMLFile1.xml"
               runat="server" target="_blank" Height="200px" Width="1000px" />
    </ContentTemplate>
  </asp:UpdatePanel>
</div>
```

Build and run the homepage. The image should now change automatically.

Close the web browser and stop debugging. You can adjust the timer interval so that the changes occur more frequently or less frequently if you wish.

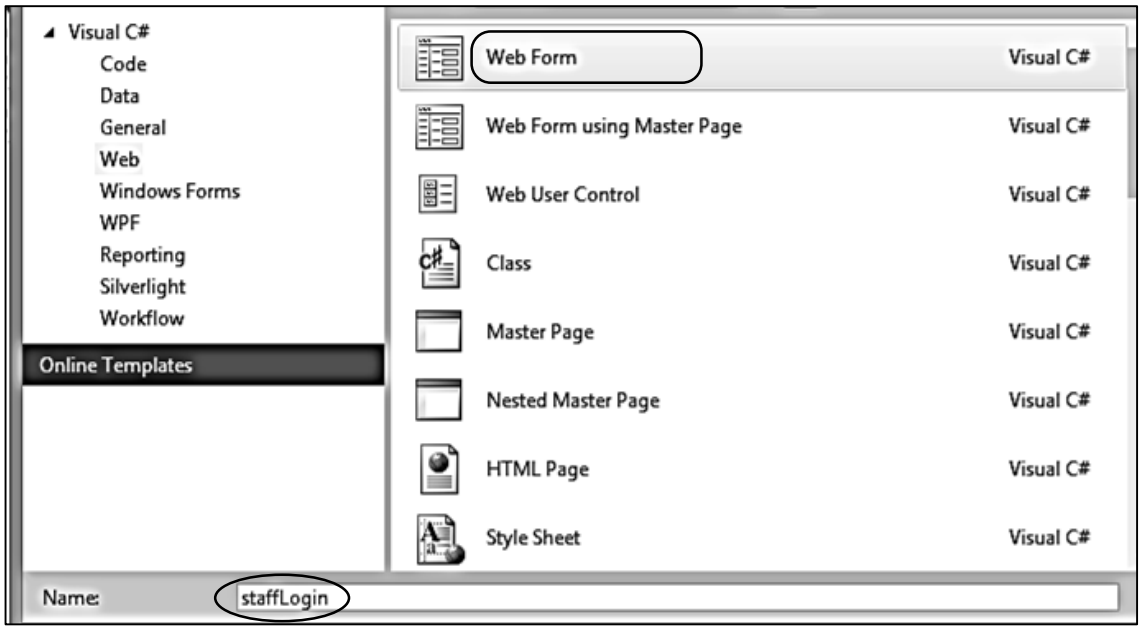
Now that we have a basic structure for the web site, we can turn our attention to the page content. Begin by creating pages for the Canal route, Boats and Bookings options. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Web*** / ***Web Form using Master Page***. Give the name '**route**'. Accept ***Site.Master*** as the master file.



Use the same procedure to create the two remaining pages: ***boats.aspx*** and ***booking.aspx.***

We will be setting up a content management system, similar to the Hardware Store project, which will allow the staff of Canal Boat Holidays to easily edit the page content of the site. In addition, staff will be able to access bookings which are made on-line.

As with the Hardware Store, the staff web site will need to be password protected. We will begin by producing a log-in page. Return to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose a simple ***Web Form***. Give the name '**staffLogin**'.



Open the ***staffLogin.aspx*** file and add lines of code to input the user name and password.

```
<head runat="server">
    <title>Staff</title>
    <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
<body>
<center>
    <form id="form1" runat="server">
    <div id="wrapper">
      <div id="staffHeader">
         <img src="Images/logo.png" width=500px />
      </div>
      <div id="login">
         <h3>Staff Log-in</h3>
      <table border="0" cellpadding="10">
      <tr>
         <td>
           User name
         </td>
       <td>
          <asp:TextBox ID="txtUser" runat="server"></asp:TextBox>
       </td>
     </tr>
     <tr>
       <td>
          Password
       </td>
       <td>
          <asp:TextBox ID="txtPassword" runat="server" TextMode="Password">
          </asp:TextBox>
       </td>
      </tr>
      <tr>
        <td>
        </td>
        <td>
           <asp:Button ID="enter" runat="server" Text="Enter" />
        </td>
       </tr>
       </table>
     </div>
   </div>
    </form>
</center>
  </body>
</html>
```

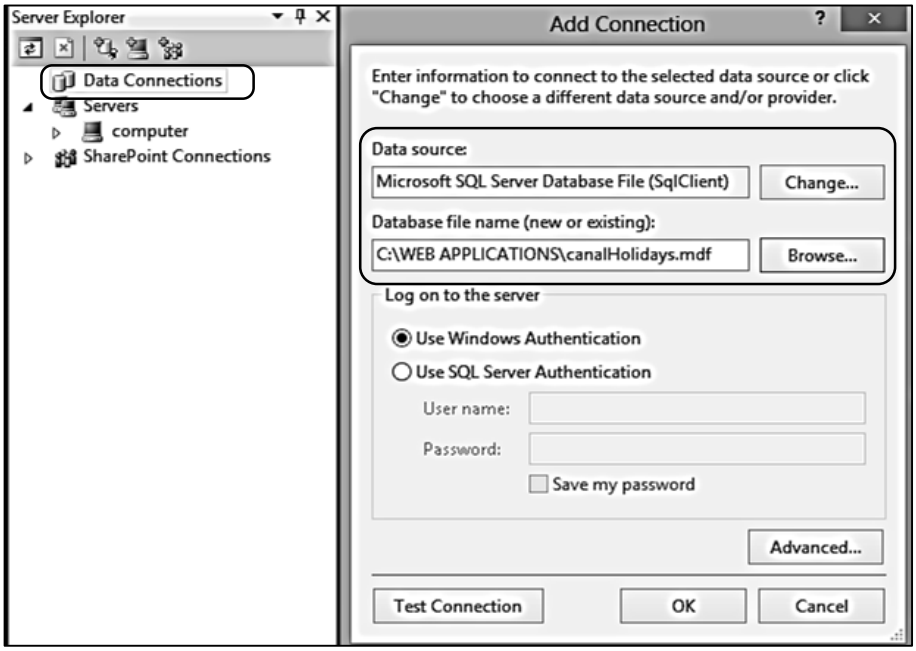
We have used two new divisions on the staff log-in page. Open the ***Style Sheet*** file and add sections of formatting code for these divisions.

```
#staffHeader
 {
     background-color: #103B80;
     width: 1000px;
     height: 112px;
}

 #login
 {
     width: 1000px;
     height: 300px;
     background-color: #FFFFFF;
     padding:20px;
 }
```

Build and run the ***staffLogin*** page. Check that the input boxes and '***Enter***' button are correctly displayed. Close the browser and stop debugging.

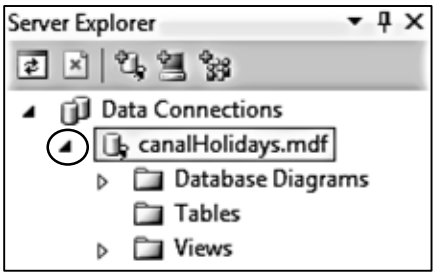
Before proceeding further with the web site, we will set up a database table to store staff log-in data.

Select '***Server Explorer***' from the '***View***' drop down list on the top menu bar of Visual Studio. Right click the ***Data Connections*** icon and select '***Add Connection***'. Check that the ***Data source*** is set to '***Microsoft SQL Server Database File (SqlClient)***'. Use the ***Browse*** button to choose a suitable location to store the database on your computer, and give the name '***canalHolidays.mdf***' for the database file, as shown below.

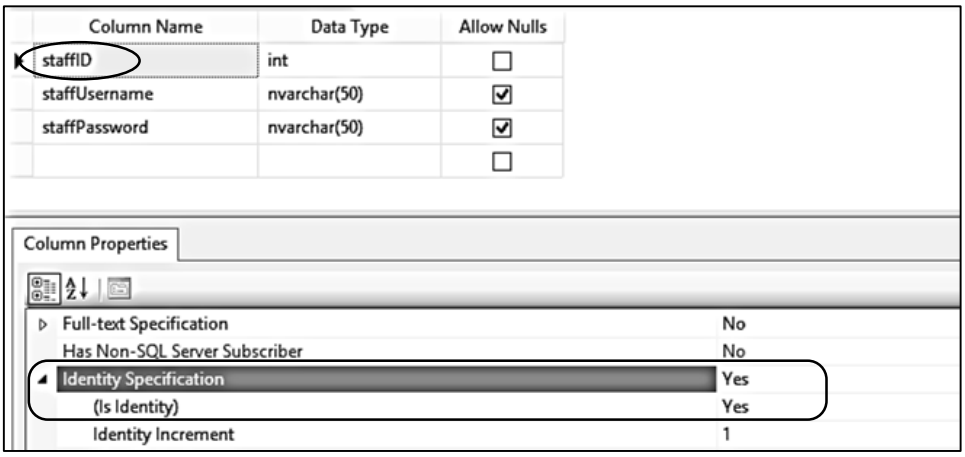


Click OK, then confirm that you wish to create a new database file.

An icon for the newly created canalHolidays database should appear in the Server Explorer window. Click the small arrow to the left of the ***canalHolidays.mdf*** icon to open the list of database components.

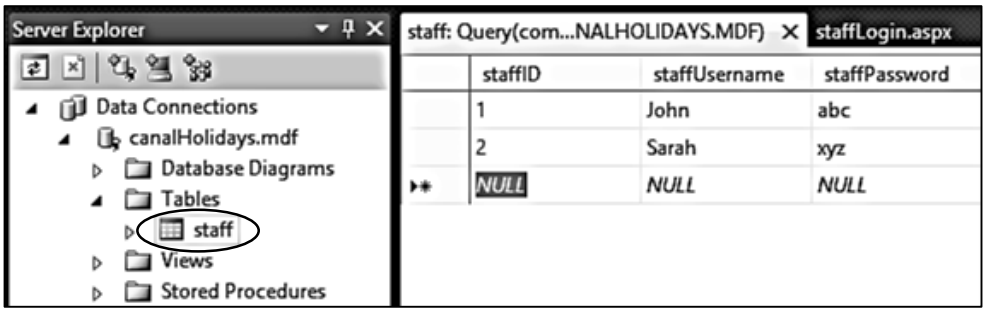


Right click the ***Tables*** icon and select '***Add New Table***'. Enter the fields shown below. The ***staffID*** field should be set to an auto-number by selecting ***Identity Specification***, opening the additional options by means of the small arrow, then selecting '**Yes**' for the ***(Is Identity)*** property:



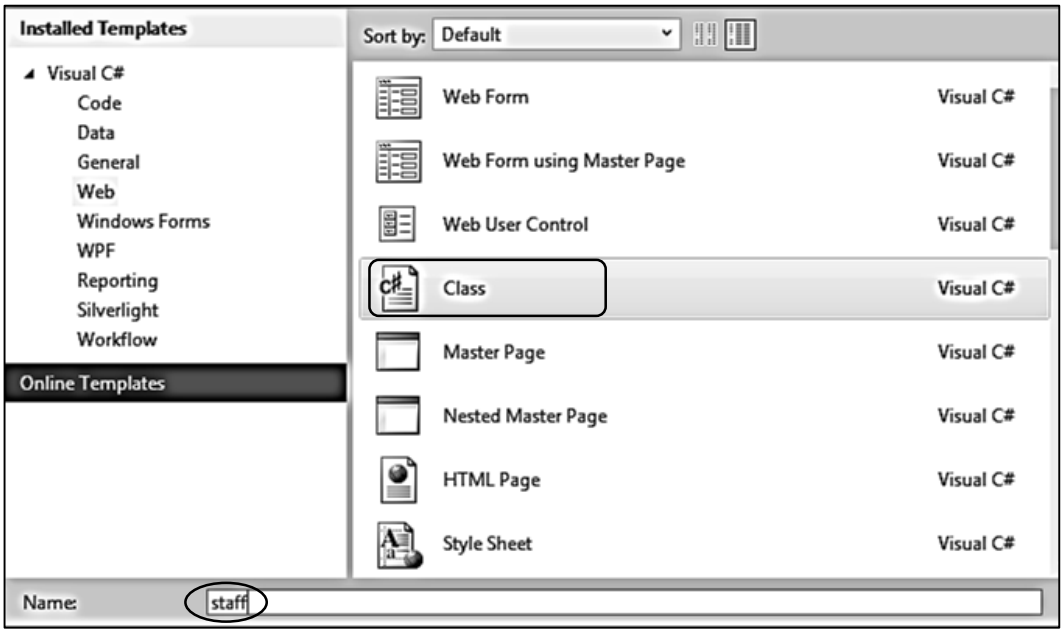
Close the table using the small '**X**' icon at the top right of the form, then save this with the name '**staff**'.

Right click the ***staff*** table icon and select '***Show Table Data***'. Add usernames and passwords for a couple of staff members. Note that there is no need to enter the **staffID** numbers in the first column of the table; these will be added automatically by the computer. Close the table when data entry is completed.



As in the Hardware Store project, we will use an object oriented approach for data handling.

Begin by setting up a ***class*** of ***staff*** objects. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name '**staff**'.



Open the ***staff.cs*** class file and add lines of code as shown below.

We begin by setting up a variable ***staffCount*** to count the number of staff objects created, and an array to provide links to these staff objects. We then create object properties corresponding to each of the fields of a staff record.

The ***loadStaff( )*** method loads each of the staff records from the database. When loaded, the records are stored temporarily in a data set called ***dsStaff***:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

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

namespace Canal_holidays
{
    public class staff
    {
        public static int staffCount = 0;
        public static staff[] staffObject = new staff[10];
        public static string databaseLocation =
                              "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";
        public int staffID { get; set; }
        public string staffUsername { get; set; }
        public string staffPassword { get; set; }
        public static bool login = false;

        public static void loadStaff()
        {
           DataSet dsStaff = new DataSet();
           SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
              AttachDbFilename="+ databaseLocation + "Integrated Security=True;
              Connect Timeout=30; User Instance=True");
           try
           {
               cnTB.Open();
               SqlCommand cmStaff = new SqlCommand();
               cmStaff.Connection = cnTB;
               cmStaff.CommandType = CommandType.Text;
               cmStaff.CommandText = "SELECT * FROM staff";
               SqlDataAdapter daStaff = new SqlDataAdapter(cmStaff);
               daStaff.Fill(dsStaff);
               cnTB.Close();
           }
           catch
           {
           }
        }

    }
```

The records which have been loaded are now used to create ***staff objects***. Add lines of code to the ***loadStaff( )*** method to do this:

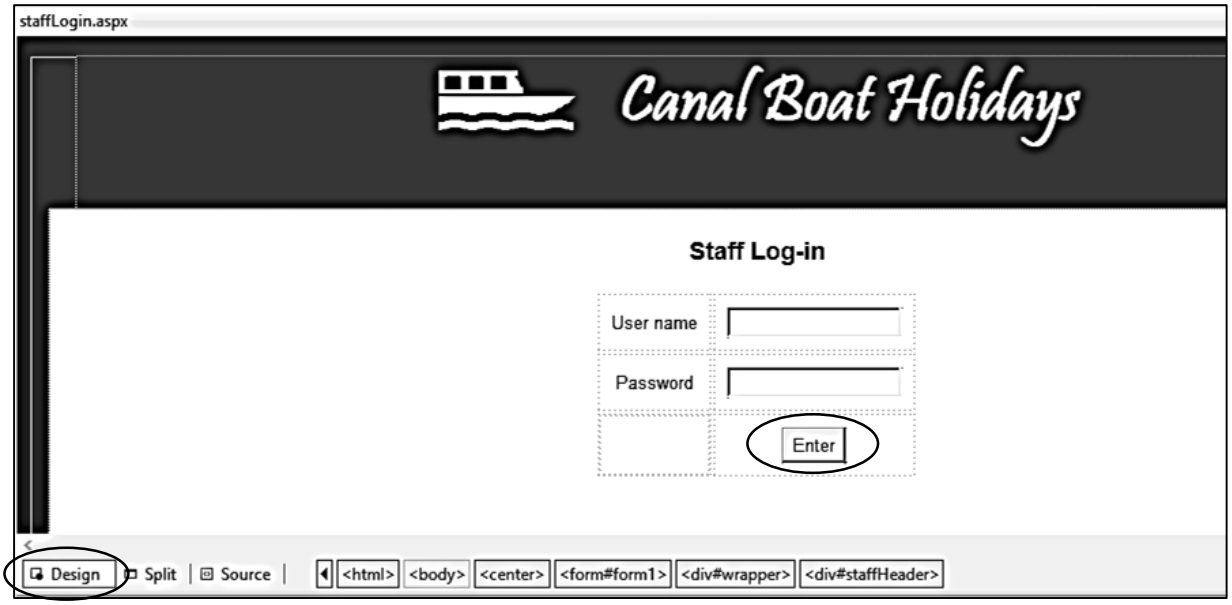
```
try
{
    cnTB.Open();
    SqlCommand cmStaff = new SqlCommand();
    cmStaff.Connection = cnTB;
    cmStaff.CommandType = CommandType.Text;
    cmStaff.CommandText = "SELECT * FROM staff";
    SqlDataAdapter daStaff = new SqlDataAdapter(cmStaff);
    daStaff.Fill(dsStaff);
    cnTB.Close();

    int countRecords = dsStaff.Tables[0].Rows.Count;
    staff.staffCount = 0;
    for (int i = 0; i < countRecords; i++)
    {
        DataRow drStaff = dsStaff.Tables[0].Rows[i];
        int staffID = (int)drStaff[0];
        string username = Convert.ToString(drStaff[1]);
        string password = Convert.ToString(drStaff[2]);

        staff.staffObject[staff.staffCount] = new staff();
        staff.staffObject[staff.staffCount].staffID = staffID;
        staff.staffObject[staff.staffCount].staffUsername = username;
        staff.staffObject[staff.staffCount].staffPassword = password;
        staff.staffCount++;
    }
}
catch
{

}
```

Return to the ***staffLogin.aspx*** page and open the C# code file which accompanies this page. This can be done by right clicking and selecting the '***View Code***' option.

Add lines to the ***Page_Load*** method to call the ***loadStaff( )*** method in the ***staff*** class file. Notice that we only need the staff objects to be created once, so the ***loadStaff*** method only operates if no staff objects are currently in the RAM memory..

```
protected void Page_Load(object sender, EventArgs e)
{
    if (staff.staffCount == 0)
    {
        staff.loadStaff();
    }
}
```

Return to the ***staffLogin.aspx*** page and click the ***Design*** button in the bottom left corner to show the design view. Double click the ***Enter*** button to create a ***Button_click*** method.



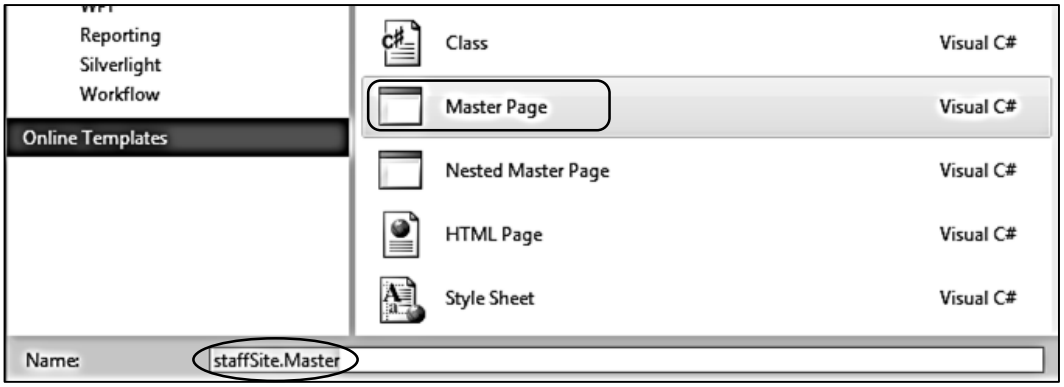
The staff usernames and passwords are held in the set of ***staff objects***. We can add a section of code to check through each of the objects to see if there is a match with the data entered on the log-in screen.

```
protected void enter_Click(object sender, EventArgs e)
{
    string userWanted = txtUser.Text;
    string passWanted = txtPassword.Text;

    bool found = false;
    for (int i = 0; i < staff.staffCount; i++)
    {
        if (userWanted == staff.staffObject[i].staffUsername &&
                          passWanted == staff.staffObject[i].staffPassword)
        {
            found = true;
        }
    }
    if (found == true)
    {
        staff.login = true;
        Response.Redirect("staffHome.aspx");
    }
}
```

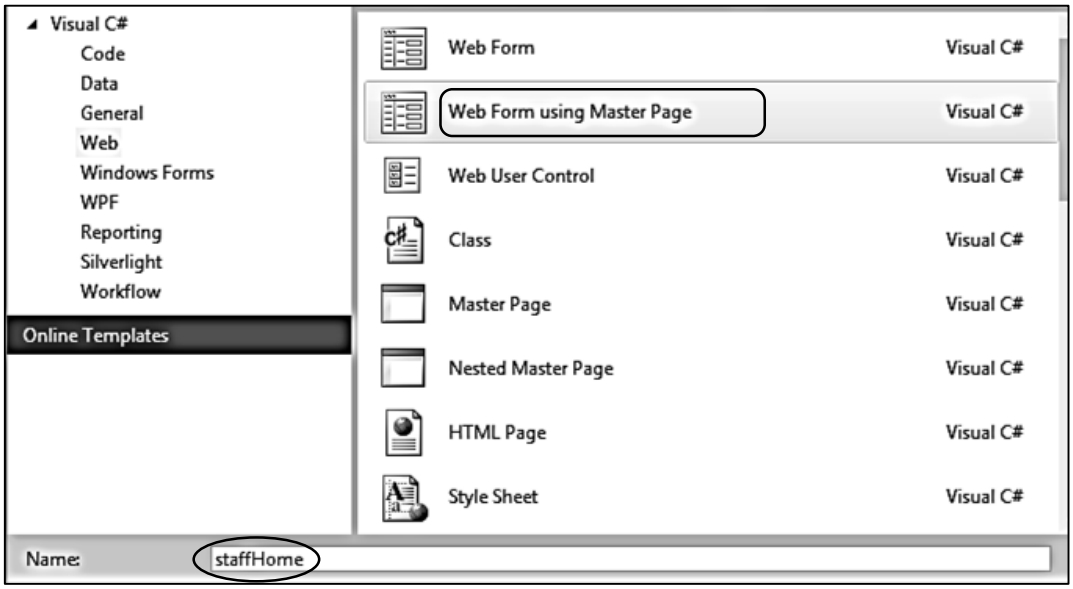
Build and run the ***staffLogin*** page. Enter an incorrect username/password then click the ***Enter*** button. The computer should ignore the entry and remain on the login page. Now enter a correct username/password. In this case, the computer should attempt to load another page called ***staffHome.aspx.*** We will work on this page next. Close the browser and stop debugging.

The functions required by the staff system are: to edit the home page content, edit the information about locations along the canal route, edit details of canal boats for hire, and to handle bookings made on-line.

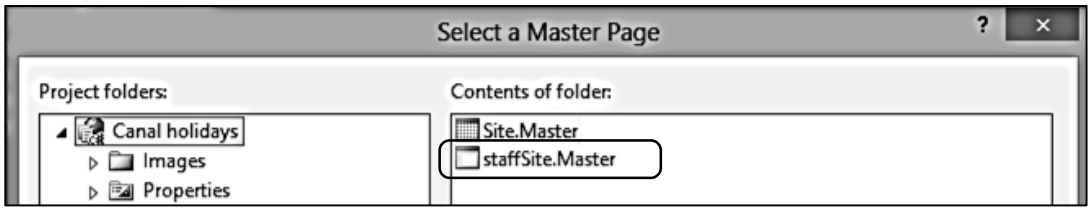
We will set up a menu for the staff system in a similar way to the main web site by using the set of tab images. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Master Page***. Give the name '**staffSite.Master**'.



We will create a set of pages which use the staffSite master file. Return to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give the name '**staffHome**'.



Select ***staffSite.Master*** as the master page and click the '***OK***' button.



Repeat these steps to create three further pages: ***staffRoute***, ***staffBoats*** and ***staffBooking***. In each case, use ***staffSite.Master*** as the master page.

Open the ***staffSite.Master*** file. Add lines of code to link to the style sheet and create a menu bar. Note that the ***ContentPlaceHolder1*** section should be moved to after the closing div tag *** ***.

```
<head runat="server">
  <title>Staff</title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
  <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
<body bgcolor="#103B80">
  <center>
    <form id="form1" runat="server">
  <div id="staffHeader" >
     <img src="Images/logo.png" width="500px" />
     <br />
     <a href="staffHome.aspx">
     <img  src="/Images/Bhome.png" onmouseover="this.src='/Images/Ohome.png'"
                          onmouseout="this.src='/Images/Bhome.png'" /></a>
     <a href="staffRoute.aspx" >
      <img  src="/Images/Broute.png" onmouseover="this.src='/Images/Oroute.png'"
                         onmouseout="this.src='/Images/Broute.png'" /></a>
    <a href="staffBoats.aspx" >
    <img  src="/Images/Bboats.png" onmouseover="this.src='/Images/Oboats.png'"
                         onmouseout="this.src='/Images/Bboats.png'" /></a>
    <a href="staffBooking.aspx" >
    <img  src="/Images/Bbooking.png" onmouseover="this.src='/Images/Obooking.png'"
                        onmouseout="this.src='/Images/Bbooking.png'" /></a>
    <br />
  </div>
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>
 </form>
</center>
</body>
```

Build and run the ***staffHome.aspx*** web page. Check that the menu is displayed, and correct pages are loaded when the different tabs are clicked. Close the browser and stop debugging.

We will work first on the home page. Staff of Canal Boat Holidays will use this to provide up-to-date information and news for customers, so the page content will probably be updated regularly.

The homepage can have a magazine-style layout with two columns, each displaying a title, information text and a picture image.

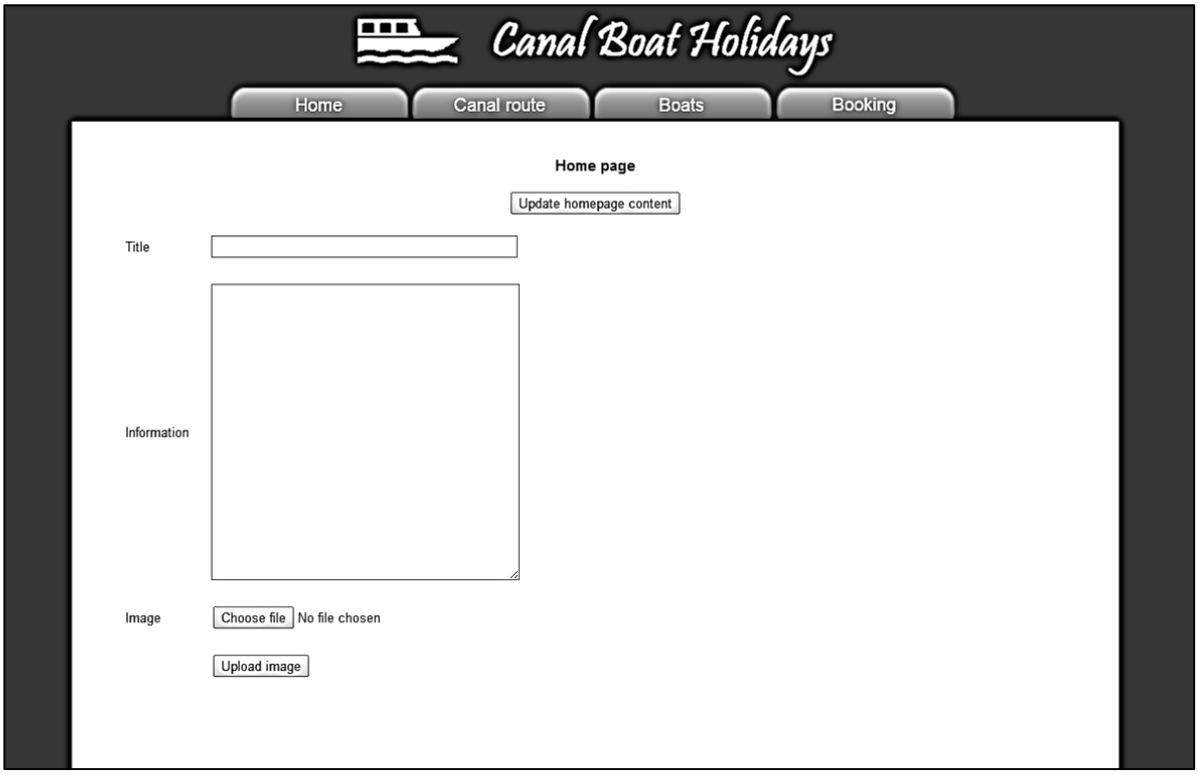
Open the ***staffHome.aspx*** file and add lines of code to the '***Content2***' section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
 <div id="content2">
   <h3>Home page</h3>
   <asp:Button ID="btnUpdate" runat="server" Text="Update homepage content" />
   <br />
   <div id="leftColumn">
      <table border="0" cellpadding="10">
        <tr>
           <td>Title</td>
           <td>
             <asp:TextBox ID="txtTitle1" runat="server" Width="300px"></asp:TextBox>
           </td>
        </tr>
        <tr>
           <td>Information</td>
           <td>
             <asp:TextBox ID="txtInformation1" runat="server" Width="300px"
                       TextMode="MultiLine" Rows="18"></asp:TextBox>
           </td>
        </tr>
        <tr>
           <td>Image</td>
           <td>
              <asp:FileUpload ID="FileUpload1" runat="server" />
           </td>
        </tr>
        <tr>
           <td></td>
           <td>
                <asp:Button ID="btnUpload1" runat="server" Text="Upload image"  />
           </td>
        </tr>
     </table>
     <br />
     <asp:Label ID="Label1" runat="server"></asp:Label>
   </div>
 </div>
</asp:Content>
```

We have created two new divisions, '***content2***' and '***leftColumn***'. Open the ***Style Sheet*** and add formatting code for these.

```
#content2
{
     width: 1000px;
     height: 1000px;
     background-color: #FFFFFF;
     padding:20px;
}

#leftColumn
{
    float:left;
    padding:20px;
    width: 450px;
    text-align:left
}
```

Build and run the staff website and select the Home page. We have set up entry boxes for one column of the display.



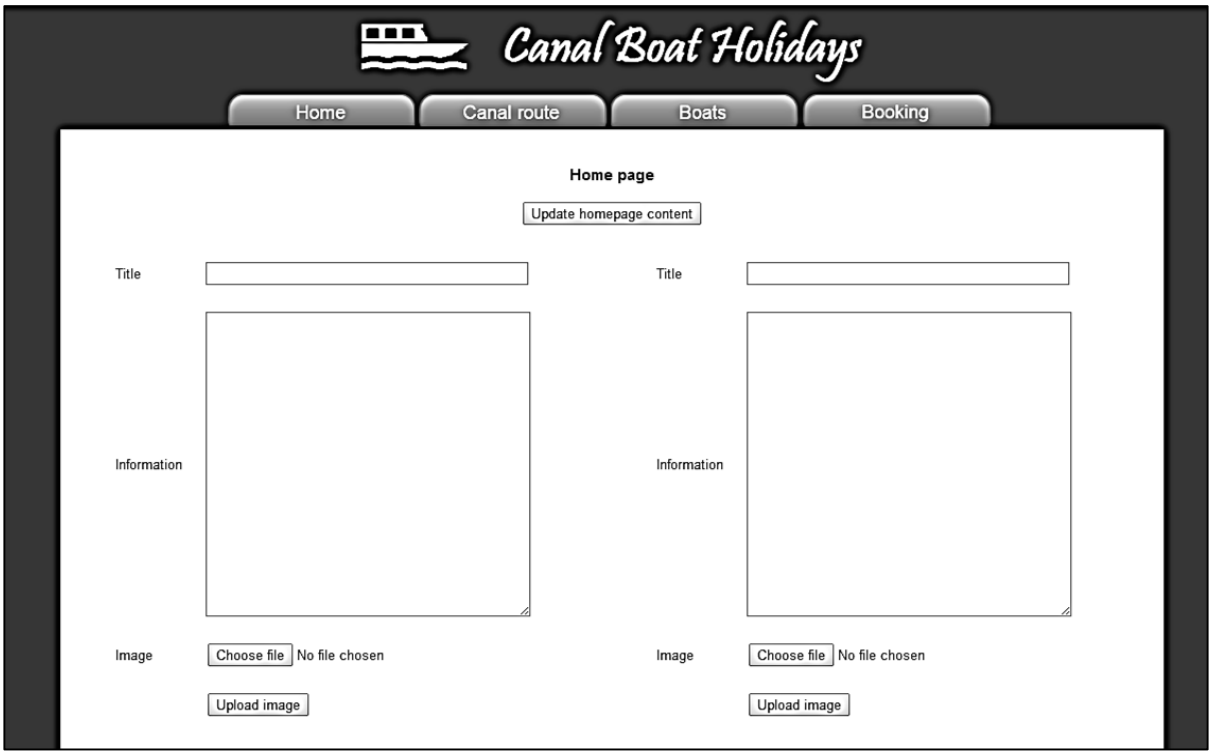
Close the browser and stop debugging. Return to the ***staffHome.aspx*** file and add the lines of code shown on the next page to produce a second data entry column.

```
        <br />
        <asp:Image ID="Image1" runat="server"  width='300px' />
    </div>
    <div id=rightColumn>
     <table border=0 cellpadding=10>
       <tr>
          <td>Title</td>
           <td>
             <asp:TextBox ID="txtTitle2" runat="server" Width="300px"></asp:TextBox>
           </td>
       </tr>
       <tr>
          <td>Information</td>
          <td>
             <asp:TextBox ID="txtInformation2" runat="server" Width="300px"
                   TextMode="MultiLine" Rows="18"></asp:TextBox>
          </td>
       </tr>
       <tr>
          <td>Image</td>
          <td>
             <asp:FileUpload ID="FileUpload2" runat="server" />
          </td>
        </tr>
        <tr>
          <td></td>
          <td>
             <asp:Button ID="btnUpload2" runat="server" Text="Upload image" />
          </td>
        </tr>
      </table>
      <br />
      <asp:Label ID="Label2" runat="server"></asp:Label>
    </div>
  </div>
</asp:Content>
```

Add code to the ***Style Sheet*** file for formatting the division '***rightColumn***'.

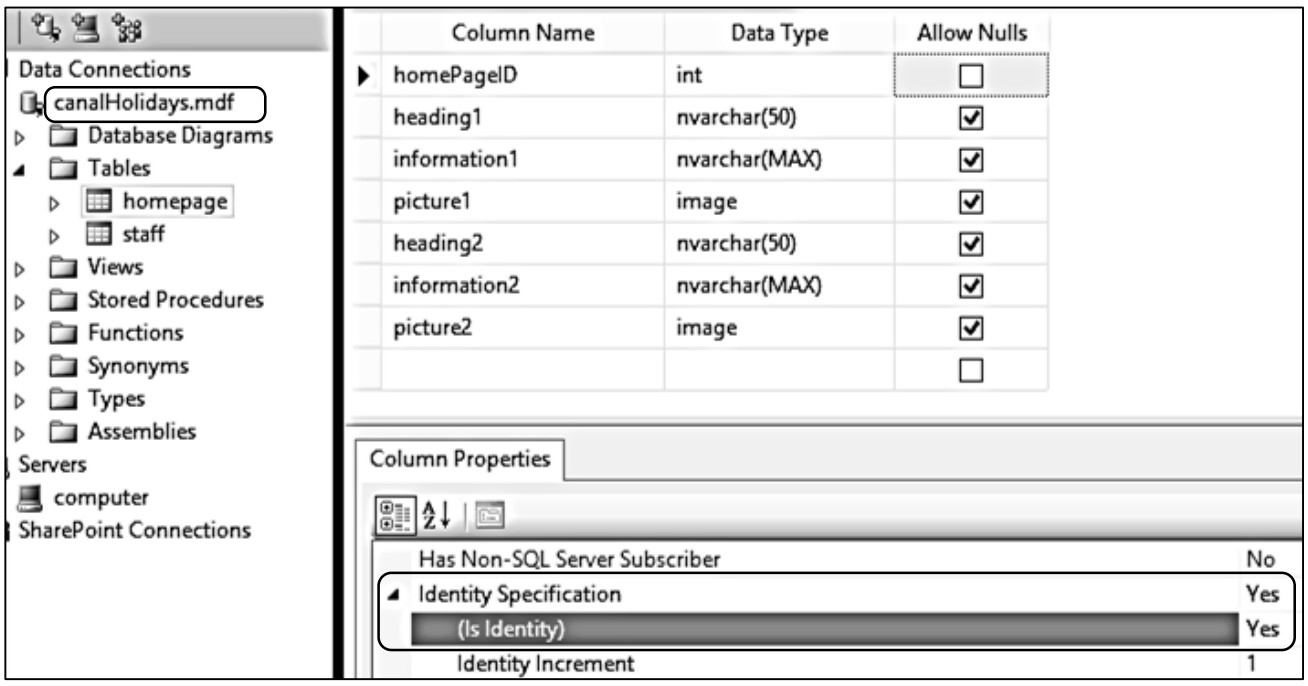
```
#rightColumn
{
    float:right;
    padding:20px;
    width: 450px;
    text-align:left
}
```

Build and run the staff website. The Home page should now display two columns for data entry.



We will now add a table to the database to store the home page content . Go to the Server Explorer window and open the ***canalHolidays*** database. Right click the ***Tables*** icon and select '***Add New Table***'.

Insert the fields shown below. The '***homePageID***' field should be set as an auto-number by opening the '***Identity Specification***' property and changing the value of '***(Is identity)***' to '***Yes***'.



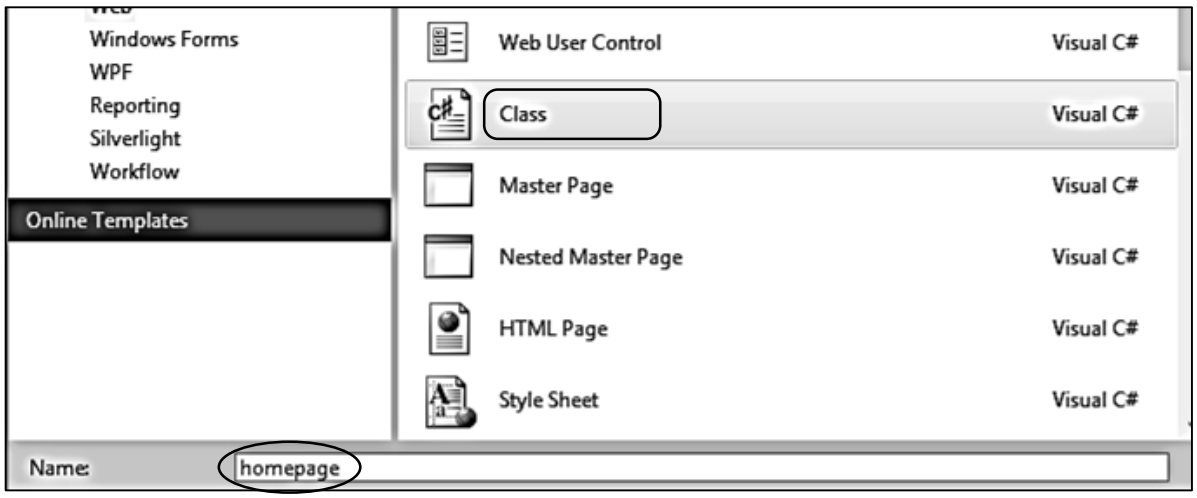
Close the table and save it with the name '***homepage***'.

Right click the homepage table icon and select ***'Show Table Data***'. Create one record by entering test data in the Heading and Information fields of the first record. The ***homePageID*** auto-number value will be added automatically by the program.

homePageID	heading1	information1	picture1	heading2	information2	picture2
1	Test Heading 1	Test Information 1	*NULL*	Test Heading 2	Test Information 2	*NULL*
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Close the ***homepage*** table. We will now set up a ***class*** to handle the homepage data.

Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name '**homepage**'.



Open the ***homepage.cs*** class file and add lines of code.

```
using System.Linq;
using System.Web;

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

namespace Canal_holidays
{
    public class homepage
    {
        public static string databaseLocation =
                            "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";
        public string title1 { get; set; }
        public string title2 { get; set; }
        public string information1 { get; set; }
        public string information2 { get; set; }
        public static homepage homeObject;

    }
}
```

We will now add a method to load the homepage record from the database table and create a ***homepage object***. Notice that, unlike the staff login system, no loop is needed when loading the data. This is because only one home page record is ever created, and this record is simply updated if the content of the page changes.

```
    public string information2 { get; set; }
    public static homepage homeObject;

    public static void loadHomepage()
    {
        DataSet dsHome = new DataSet();
        SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
           AttachDbFilename=" + databaseLocation + "Integrated Security=True;
           Connect Timeout=30; User Instance=True");
        try
        {
            cnTB.Open();
            SqlCommand cmHome = new SqlCommand();
            cmHome.Connection = cnTB;
            cmHome.CommandType = CommandType.Text;
            cmHome.CommandText = "SELECT * FROM homepage";
            SqlDataAdapter daHome = new SqlDataAdapter(cmHome);
            daHome.Fill(dsHome);
            cnTB.Close();
            DataRow drHome = dsHome.Tables[0].Rows[0];
            string title1 = Convert.ToString(drHome[1]);
            string information1 = Convert.ToString(drHome[2]);
            string title2 = Convert.ToString(drHome[4]);
            string information2 = Convert.ToString(drHome[5]);
            homepage.homeObject = new homepage();
            homepage.homeObject.title1 = title1;
            homepage.homeObject.information1 = information1;
            homepage.homeObject.title2 = title2;
            homepage.homeObject.information2 = information2;
        }
        catch
        {
        }
    }
}
```

Go now to the ***staffHome.aspx*** C# page and add variables at the start of the class.

```
    public partial class staffHome : System.Web.UI.Page
    {
        byte[] picbyte1;
        byte[] picbyte2;
        string title1;
        string title2;
        string information1;
        string information2;

        protected void Page_Load(object sender, EventArgs e)
        {
```

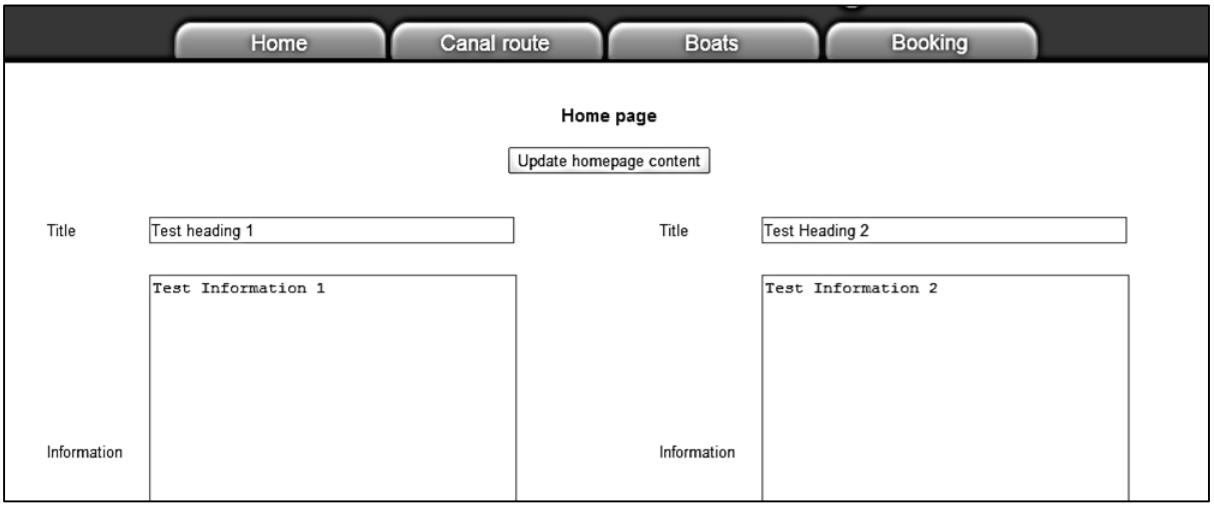
We will now add lines of code to the Page_Load method. This code carries out two functions:

- The program firstly checks that a valid staff login has been made. If not, the user is redirected to the log-in page. This avoids the risk of an unauthorised user by-passing the security system through loading the page directly.
- The program checks whether data is already displayed, and only loads the homepage from the database if the page entries are currently blank. It is a feature of ASP.NET that a webpage is reloaded each time a button is clicked; this allows for the page to be updated in response to commands from the user. ***Page_Load*** is always the first method to run each time the webpage reloads, and this could result in data being reloaded multiple times from the database unless a check is made.

We will also add the ***loadData( )*** method, which collects the homepage object and transfers the data items to the correct text boxes

```
protected void Page_Load(object sender, EventArgs e)
{
    if (staff.login == false)
    {
        Response.Redirect("staffLogin.aspx");
    }
    if (txtTitle1.Text == "")
    {
        loadData();
    }
}

protected void loadData()
{
     homepage.loadHomepage();
     txtTitle1.Text = homepage.homeObject.title1;
     txtTitle2.Text = homepage.homeObject.title2;
     txtInformation1.Text = homepage.homeObject.information1;
     txtInformation2.Text = homepage.homeObject.information2;
}
```

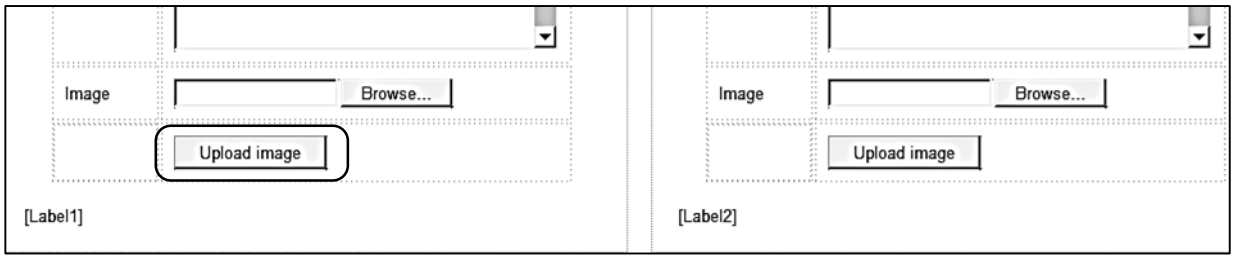
Build and run the staff website. Select the Home page, and check that your test data is displayed in the correct text boxes. Close the browser and stop debugging.



We will now work on the uploading of photographs to the database record. Go to the ***hompage.cs*** class file and add a method to upload the picture images. There are two pictures to be uploaded, and these must be stored in the correct fields of the database record. We are using a variable 'picturePosition' which will be set to '1' for the left column picture, or '2' for the right column picure. Different SQL update queries are carried out according to this value.

```
public string information2 { get; set; }
public static homepage homeObject;

public static void updateImage(int picturePosition, byte[] picbyte)
{
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmHome = new SqlCommand();
        cmHome.Connection = cnTB;
        cmHome.CommandType = CommandType.Text;
        string query;
        if (picturePosition == 1)
        {
            query = "UPDATE homepage SET picture1 = @pic WHERE homePageID='1'";
        }
        else
        {
            query = "UPDATE homepage SET picture2 = @pic WHERE homePageID='1'";
        }
        SqlCommand cmd = new SqlCommand(query, cnTB);
        SqlParameter picparameter = new SqlParameter();
        picparameter.SqlDbType = SqlDbType.Image;
        picparameter.ParameterName = "pic";
        picparameter.Value = picbyte;
        cmd.Parameters.Add(picparameter);
        cmd.ExecuteNonQuery();
        cnTB.Close();
    }
    catch
    {
    }
}
```

Return to the ***staffHome.aspx*** file and change to the design view. Double click the '***Upload image***' button in the left hand column to create a button_click method.

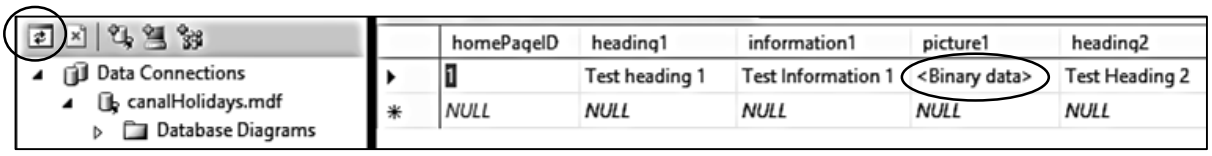


Add lines of code to the button_click method.

```
protected void btnUpload1_Click(object sender, EventArgs e)
{
    try
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(Server.MapPath("Images").ToString() + @"\" +
                                                    FileUpload1.FileName);
            string s = "<img src='Images/" + FileUpload1.FileName +
                                                "' width='400' border='0'>";
            Label1.Text = s;
            picbyte1 = FileUpload1.FileBytes;
            homepage.updateImage(1,picbyte1);
        }
    }
    catch
    {
    }
}
```

Build and run the staff website, and go to the Home page. Use the '***Choose file***' button in the left column to select a photograph, then click '***Upload image***'. The photograph should be displayed.

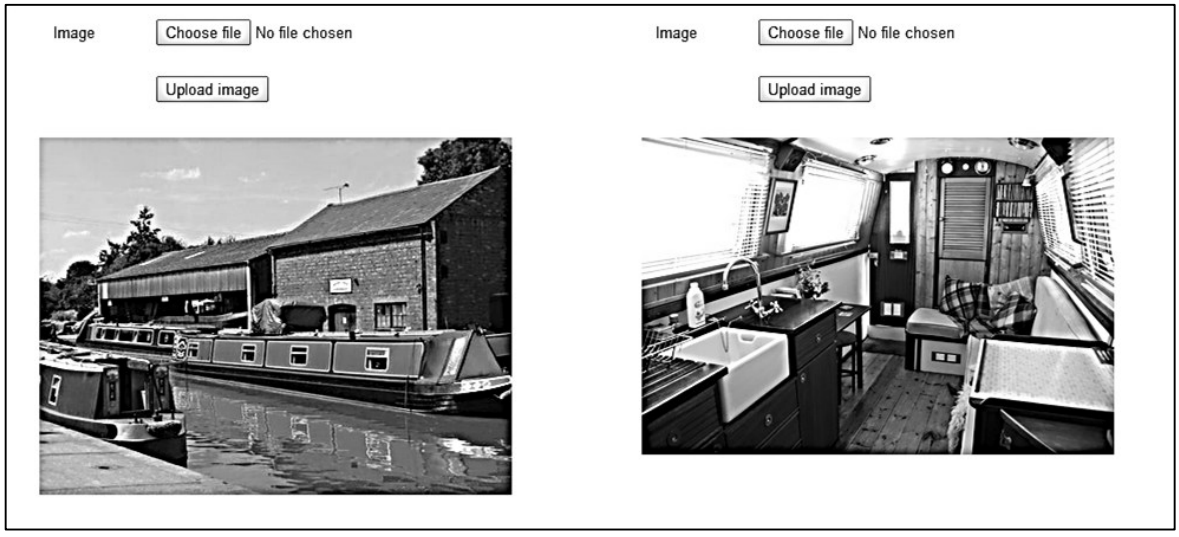
Close the browser and stop debugging. Go to the Server Explorer window and click the ***Refresh*** button. Open the ***homepage*** table and check that the message ***<binary data>*** now appears in the ***picture1*** field.



Repeat the previous steps to create a button_click method for the '***Upload image***' button in the right hand column, and add lines of code.

```
protected void btnUpload2_Click(object sender, EventArgs e)
{
    try
    {
        if (FileUpload2.HasFile)
        {
            FileUpload2.SaveAs(Server.MapPath("Images").ToString() + @"\" +
                                                        FileUpload2.FileName);
            string s = "<img src='Images/" + FileUpload2.FileName +
                                                "' width='400' border='0'>";
            Label2.Text = s;
            picbyte2 = FileUpload2.FileBytes;
            homepage.updateImage(2,picbyte2);
        }
    }
    catch
    {
    }
}
```

Build and run the staff website. Check that images can now be uploaded in both columns, and that these are represented in the database table by ***<binary data>*** messages.



homePageID	heading1	information1	picture1	heading2	information2	picture2
1	Test heading 1	Test Information 1	<Binary data>	Test Heading 2	Test Information 2	<Binary data>
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

The next step is to allow the picture images to be loaded and displayed when the staff homepage screen is opened. You may remember from previous projects that a Handler file will be necessary. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Generic Handler***. Accept the name '**Handler1**'.

Open the Handler file and replace lines of code in the ***ProcessRequest( )*** method. Note that the commands beginning:

conn = new System.Data.SqlClient.SqlConnection(...
sqlcmd = new System.Data.SqlClient.SqlCommand(...

should be entered as single lines without line breaks. Notice that we use a variable '***imgid***' to determine whether the photograph for the left or righ column is required, so that the data can be collected from the correct field of the database table.

```
public class Handler1 : IHttpHandler
{
    string databaseLocation = "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";

    public void ProcessRequest(HttpContext context)
    {
        System.Data.SqlClient.SqlDataReader rdr = null;
        System.Data.SqlClient.SqlConnection conn = null;
        System.Data.SqlClient.SqlCommand sqlcmd = null;
        try
        {
            conn = new System.Data.SqlClient.SqlConnection(@"Data Source=.\SQLEXPRESS;
                AttachDbFilename=" + databaseLocation + "Integrated Security=True;
                Connect Timeout=30; User Instance=True");
            string pictureID = Convert.ToString(context.Request.QueryString["imgid"]);
            if (pictureID == "1")
            {
                sqlcmd = new System.Data.SqlClient.SqlCommand("SELECT picture1
                       FROM homepage WHERE homePageID='1'", conn);
            }
            if (pictureID == "2")
            {
                sqlcmd = new System.Data.SqlClient.SqlCommand("SELECT picture2
                       FROM homepage WHERE homePageID='1'", conn);
            }
            conn.Open();
            rdr = sqlcmd.ExecuteReader();
```

```
        while (rdr.Read())
        {
            context.Response.ContentType = "image/jpg";
            if (pictureID == "1")

            {
                context.Response.BinaryWrite((byte[])rdr["picture1"]);
            }
            if (pictureID == "2")
            {
                context.Response.BinaryWrite((byte[])rdr["picture2"]);
            }
        }
        if (rdr != null)
            rdr.Close();
    }
    finally
    {
        if (conn != null)
            conn.Close();
    }
}
```

Return to the ***staffHome.aspx*** C# page. Add lines of code to the ***loadData( )*** method to load and display the two photographs.

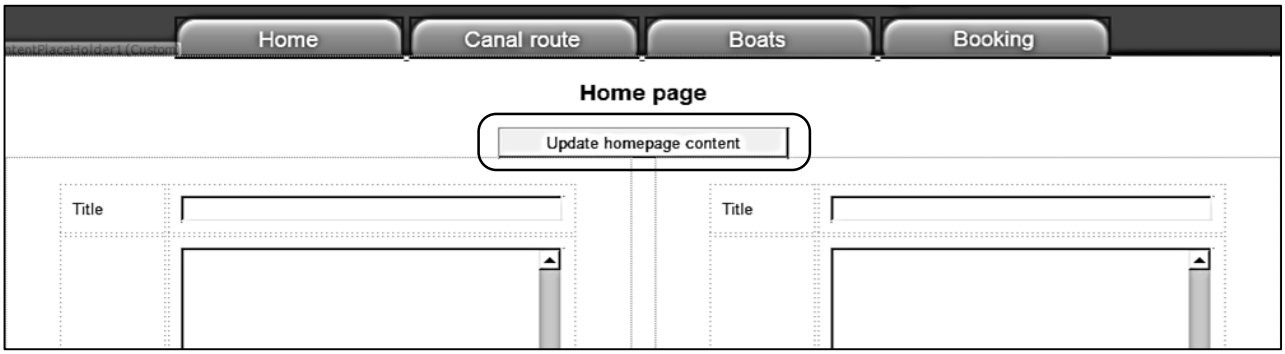
```
protected void loadData()
{
     homepage.loadHomepage();
     txtTitle1.Text = homepage.homeObject.title1;
     txtTitle2.Text = homepage.homeObject.title2;
     txtInformation1.Text = homepage.homeObject.information1;
     txtInformation2.Text = homepage.homeObject.information2;
     string s = "";
     s += "<img src='Handler1.ashx?imgid=1' width='400' border='0' >";
     Label1.Text = s;
     s = "";
     s += "<img src='Handler1.ashx?imgid=2' width='400' border='0' >";
     Label2.Text = s;
}
```

Build and run the staff website, and select the Home page. Both the text entries and the images should now be displayed. Check that you are able to change an image, and that the new image appears correctly when the web page is reopened.

It just remains to complete the updating methods for text entries on the home page. Go to the ***homepage.cs*** class file and add an ***updateHomepage( )*** method.

```
public string information2 { get; set; }
public static homepage homeObject;

public static void updateHomepage(string title1, string title2,
                                  string information1, string information2)
{
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
           AttachDbFilename=" + databaseLocation + "Integrated Security=True;
           Connect Timeout=30; User Instance=True");
     try
     {
          cnTB.Open();
          SqlCommand cmHome = new SqlCommand();
          cmHome.Connection = cnTB;
          cmHome.CommandType = CommandType.Text;
          string query;

          query = "UPDATE homepage SET heading1='" + title1 + "', information1='"
                  + information1 + "', heading2='" + title2 + "', information2='"
                  + information2 + "' WHERE homePageID='1'";
          SqlCommand cmd = new SqlCommand(query, cnTB);
          cmd.ExecuteNonQuery();
          cnTB.Close();
     }
     catch
     {
     }
 }
```

Return to the ***staffHome.aspx*** page and change to the design view. Double click the '***Update homepage content***' button to produce a button_click method.

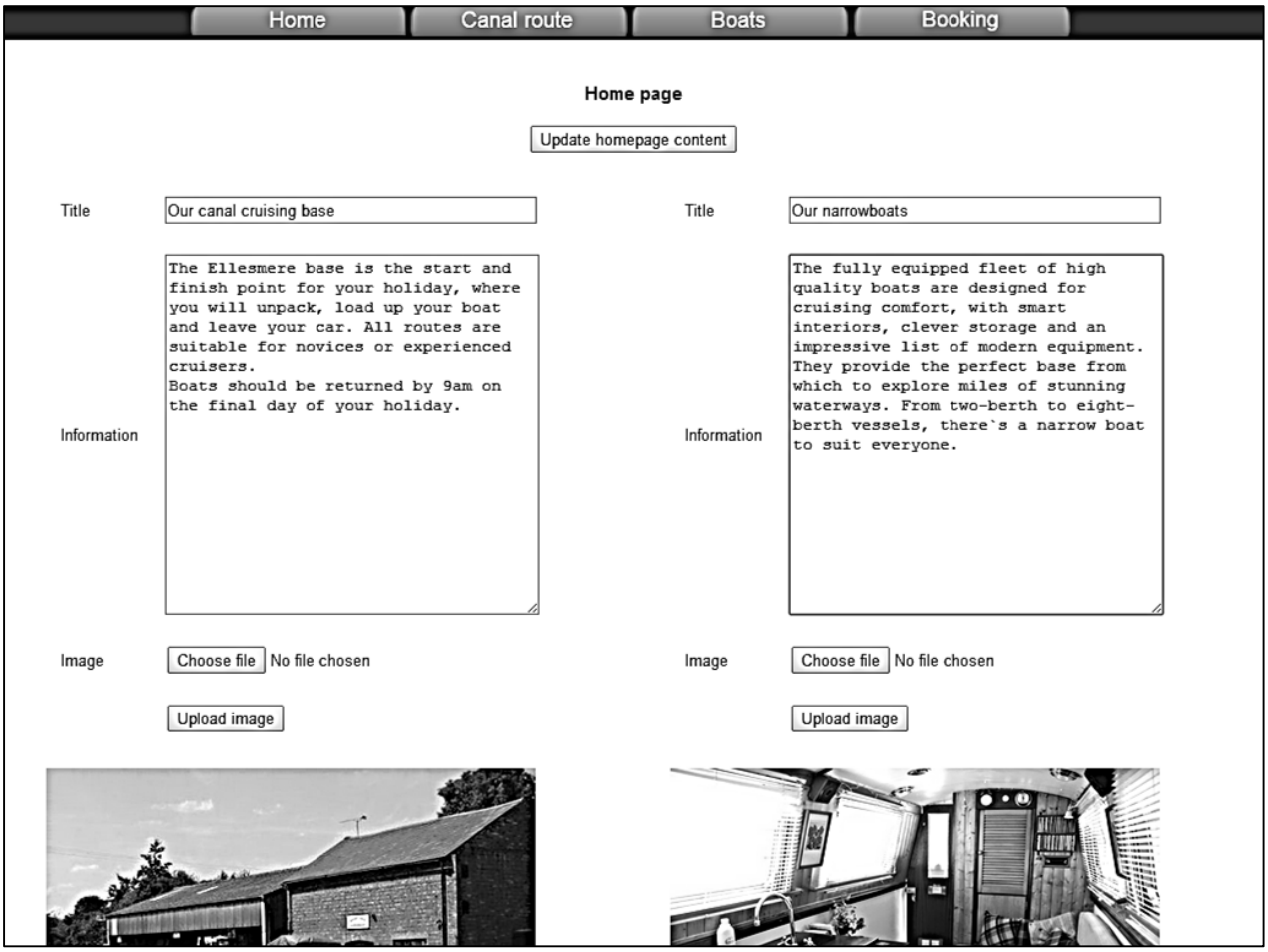


Add lines of code to the ***btnUpdate_Click( )*** method.

```
protected void btnUpdate_Click(object sender, EventArgs e)
{
    title1 = txtTitle1.Text;
    title2 = txtTitle2.Text;
    information1 = txtInformation1.Text;
    information2 = txtInformation2.Text;
    homepage.updateHomepage(title1, title2, information1, information2);
    homepage.loadHomepage();
}
```

Build and run the staff website. Open the Home page and enter a realistic amount of test data for the Title and Information fields of each column, but please avoid using apostrophy (‘) characters. These can cause an error, but we will deal with this problem shortly. Click the ‘***Update homepage content***’ button, then close the web page.

Check that the data is reloaded correctly when the web page is re-run.



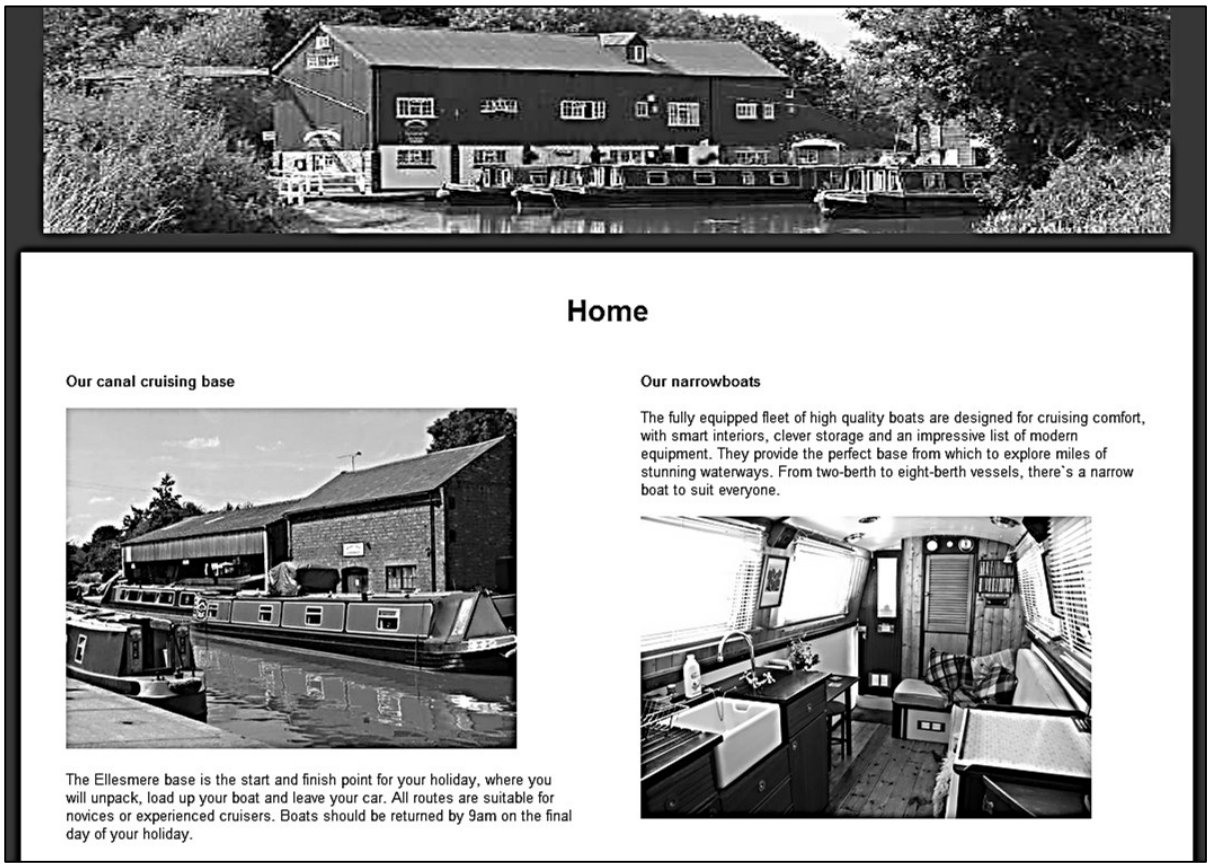
We will now return to the public homepage and set up the screen display. Open the ***home.aspx*** file and add lines of code to the ***content2*** section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
 <h1>Home</h1>
 <div id="leftColumn">
    <asp:Label ID="Label1" runat="server"></asp:Label>
 </div>
 <div id="rightColumn">
    <asp:Label ID="Label2" runat="server"></asp:Label>
 </div>
</asp:Content>
```

Right click on the ***home.aspx*** page and select the option to 'View Code'. Add a line of code to call a ***loadData( )*** method. Add the ***loadData( )*** method underneath. This will collect the home page object, then display the text and images in the two columns of the page by inserting HTML code into the ***label*** components.

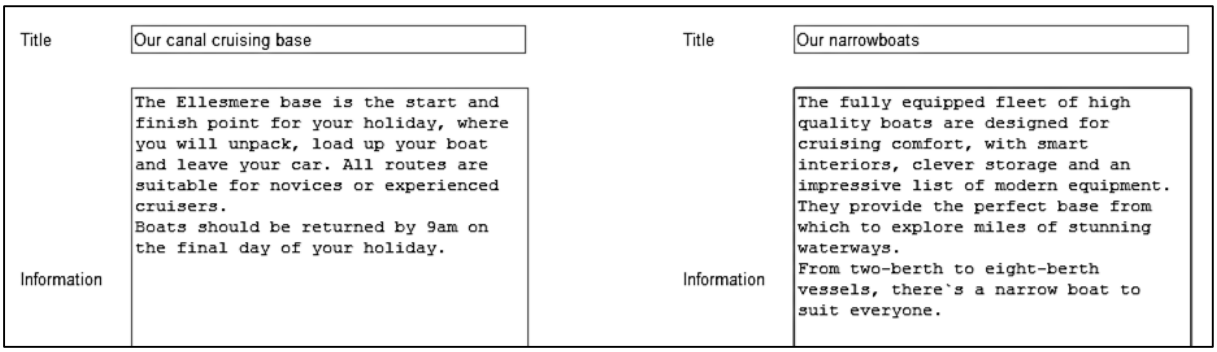
```
public partial class home : System.Web.UI.Page
  {
      protected void Page_Load(object sender, EventArgs e)
      {
          loadData();
      }

      protected void loadData()
      {
          homepage.loadHomepage();
          string s = "";
          s += "<b>"+homepage.homeObject.title1+"</b>";
          s += " <br /><br />";
          s += "<img src='Handler1.ashx?imgid=1' width='400' border='0' >";
          s += " <br /><br />";
          s += homepage.homeObject.information1;
          Label1.Text = s;
          s = "";
          s += "<b>" + homepage.homeObject.title2 + "</b>";
          s += " <br /><br />";
          s += homepage.homeObject.information2;
          s += " <br /><br />";
          s += "<img src='Handler1.ashx?imgid=2' width='400' border='0' >";
          Label2.Text = s;
      }
  }
```

Build and run the homepage. Check that the information and photographs are displayed correctly.



A couple of problems may occur with the text display, which we can now solve:

When the text was entered on the ***staffHome*** page, paragraph breaks were present. These have not appeared on the public home page.



This problem occurs because of the different ways in which line breaks are represented: by a ***'\n'*** control character in the text box, and by a ***
*** command in the HTML display. We simply need to convert from '***\n***' to '***
' before saving the text, so that the format is correct for the public homepage. We can convert back before redisplaying text in edit boxes on the ***staffHome.aspx page.

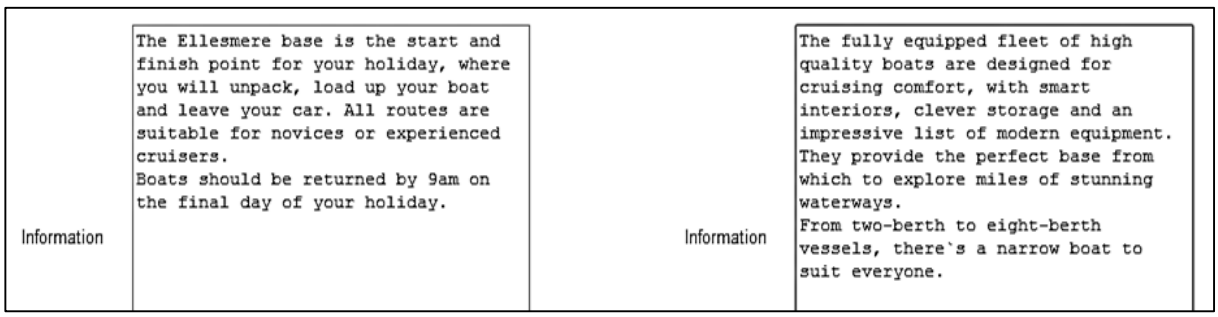
Go to the ***staffHome.aspx*** C# code page and add lines to carry out replacements in the ***btnUpdate_Click*** method .

```
protected void btnUpdate_Click(object sender, EventArgs e)
{

    title1 = txtTitle1.Text;
    title2 = txtTitle2.Text;
    information1 = txtInformation1.Text;
    information2 = txtInformation2.Text;

    information1 = information1.Replace("\n", "<br />");
    information2 = information2.Replace("\n", "<br />");

    homepage.updateHomepage(title1, title2, information1, information2);
    homepage.loadHomepage();
}
```

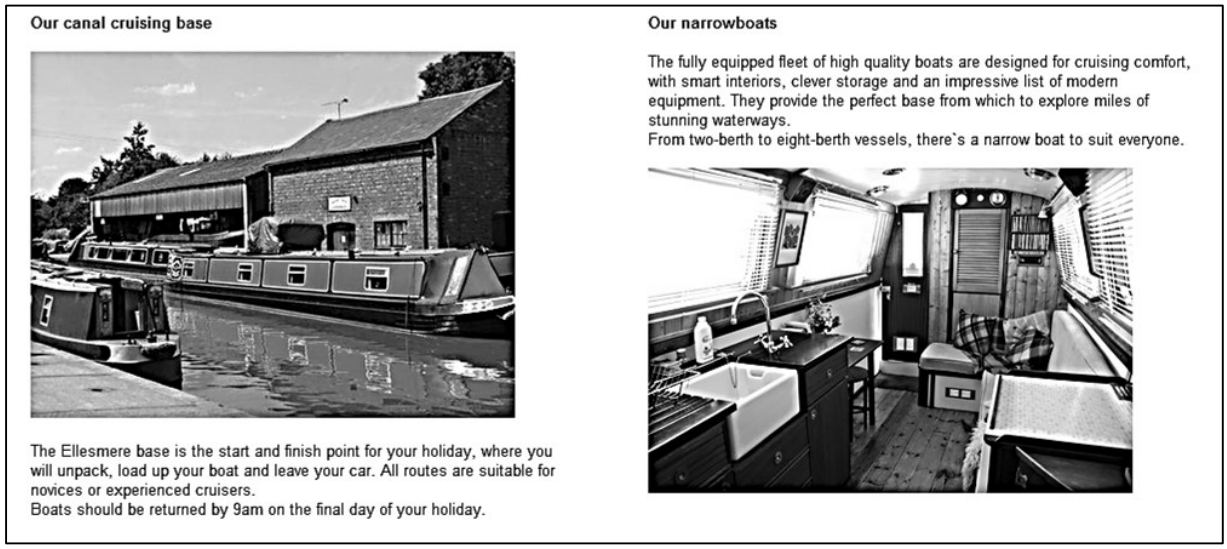
Make reverse replacements in the ***loadData( )*** method. Some existing lines of code need to be changed.

```
protected void loadData()
{
     homepage.loadHomepage();
     txtTitle1.Text = homepage.homeObject.title1;
     txtTitle2.Text = homepage.homeObject.title2;

     string information1 = homepage.homeObject.information1;
     string information2 = homepage.homeObject.information2;
     information1 = information1.Replace( "<br />","\n");
     information2 = information2.Replace( "<br />","\n");
     txtInformation1.Text = information1;
     txtInformation2.Text = information2;

     string s = "";
     s += "<img src='Handler1.ashx?imgid=1' width='400' border='0' >";
     Label1.Text = s;
     s = "";
     s += "<img src='Handler1.ashx?imgid=2' width='400' border='0' >";
     Label2.Text = s;
}
```

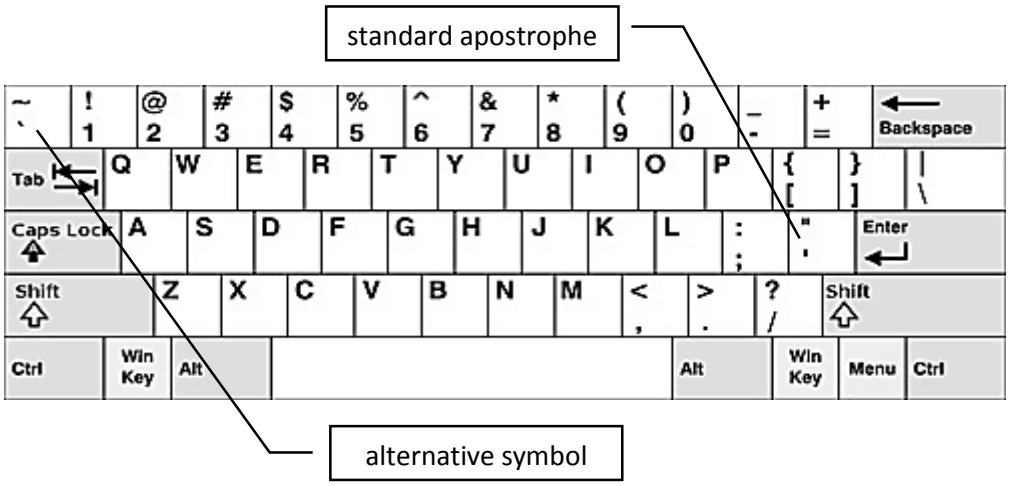
Build and run the staff website. Make paragraph breaks in the blocks of information text, and click the '***Update homepage content***' button to re-save the data.



Go to the public homepage and check that the papragraph breaks now appear correctly.



A second, and perhaps more serious, problem occurs when apostrophe characters (‘) are present in blocks of text, for example: “There’s a narrow boat to suit everyone.” The apostrophe is used as a special control character by the C# language, and can cause an error when data is being uploaded to the database. Fortunately there is a simple solution. The computer has an alternative symbol which looks similar to an apostrophe, but is not recognised as a C# control character. This is located in the upper left hand corner of the keyboard.

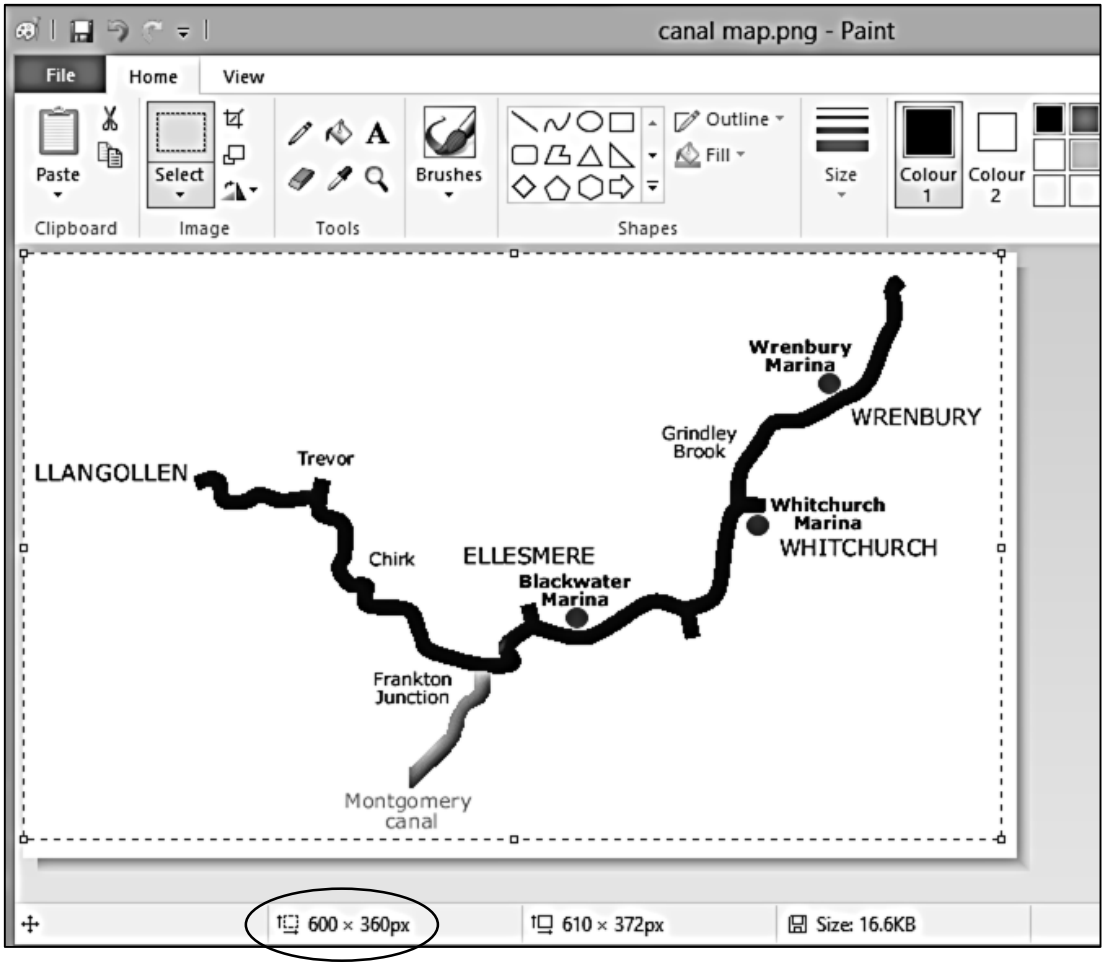


Return to the ***staffHome.aspx*** C# code page and add lines to the ***btnUpdate_Click( )*** method to make the replacement from the standard apostrophe to the alternative symbol.

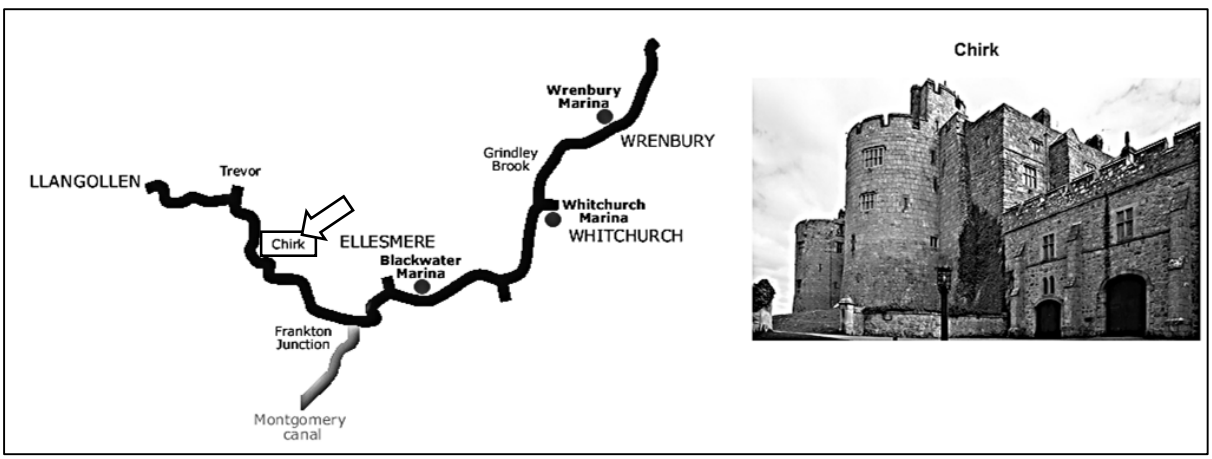
```
protected void btnUpdate_Click(object sender, EventArgs e)
{
                    . . . .
    information1 = information1.Replace("\n", "<br />");
    information2 = information2.Replace("\n", "<br />");
    information1 = information1.Replace("'", "`");
    information2 = information2.Replace("'", "`");
    homepage.updateHomepage(title1, title2, information1, information2);
```

With the homepage completed, we can now move on to the ***Canal route*** page. This will display a map of the canal and allow users to display information and photographs of places along the route.

Canal Boat Holidays operates on the Llangollen canal. Obtain a map of the canal route from the Internet, or produce your own outline map using a graphics application. This should show principal locations as illustrated below.

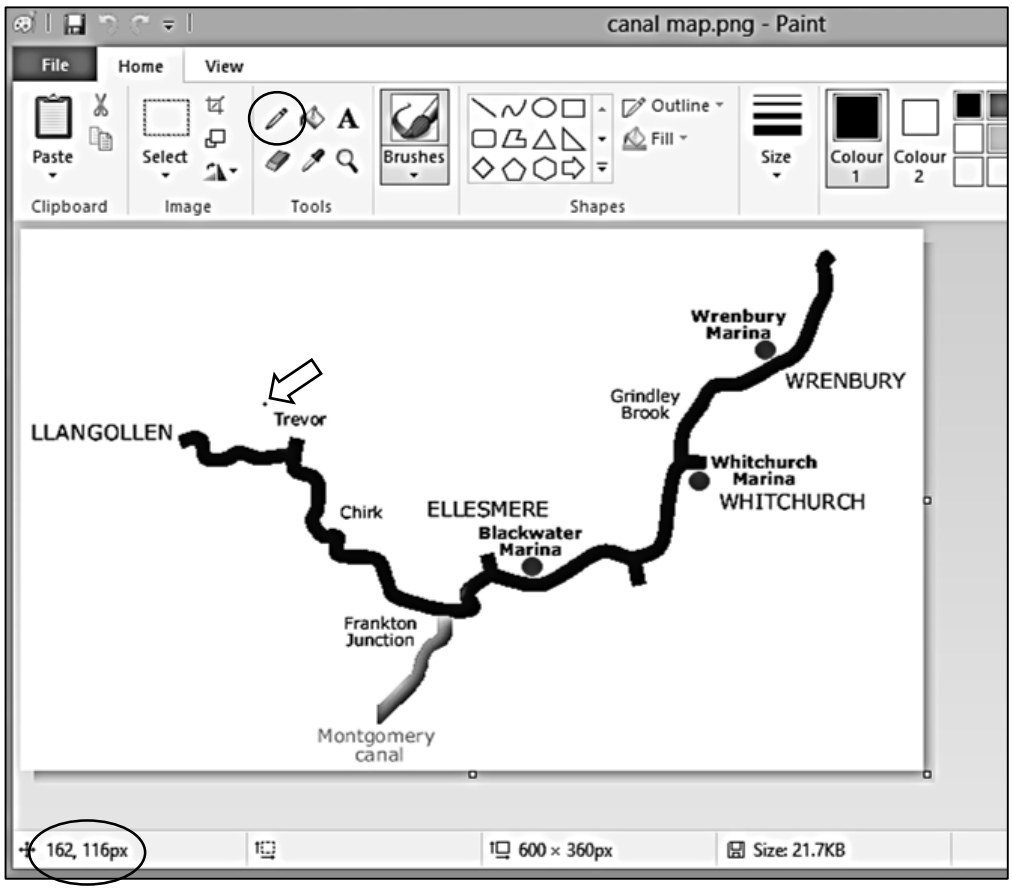


Load the map into a graphics program such as ***Microsoft Paint***. Use the ***resize*** and ***crop*** tools to adjust the size to exactly **600 pixels** wide by **360 pixels** high. Upload the completed image to the ***Images*** folder in the Solution Explorer window using the name '***canal map.png***'. This will be used to create an ***image map***, with active areas around the location names:



When the user clicks the mouse in one of the active areas, information and a photograph of the location will be displayed.

Load the canal map into ***Microsoft Paint***. Select the ***pencil*** tool, then move to one of the places marked. Notice that the current X and Y pixel cooordinates of the pencil position are displayed, and change as you move across the image.



To set up an active mouse-click area on the image map, we will need to know the X and Y coordinates of the two opposite corners of an imaginary rectangle covering the location name.

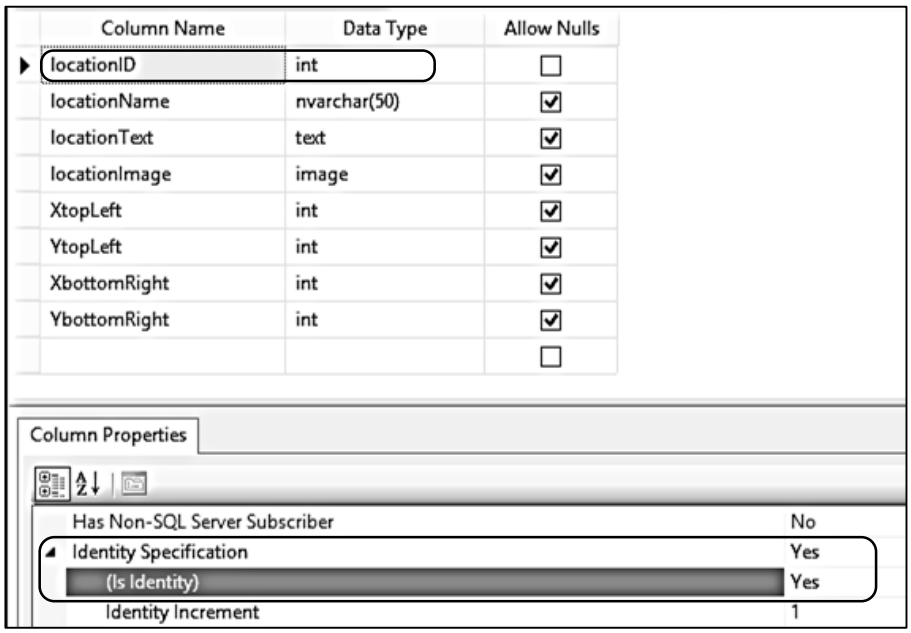
Select two locations on the map, and read-off the sets of corner cooordinates for rectangles around these locations:

top left X, top left Y, bottom right X, bottom right Y

Record the values for use as test data.

We will need to add a table to the database to store information about canal locations. Go to the Server Explorer window and right click the ***Tables*** icon. Select the '***Add New Tabl***e' option.

Add fields to the table as shown below. The ***locationID*** field should be set as an auto-number by selecting ***Identity Specification / (Is Identity)*** and giving the value '***Yes***'.

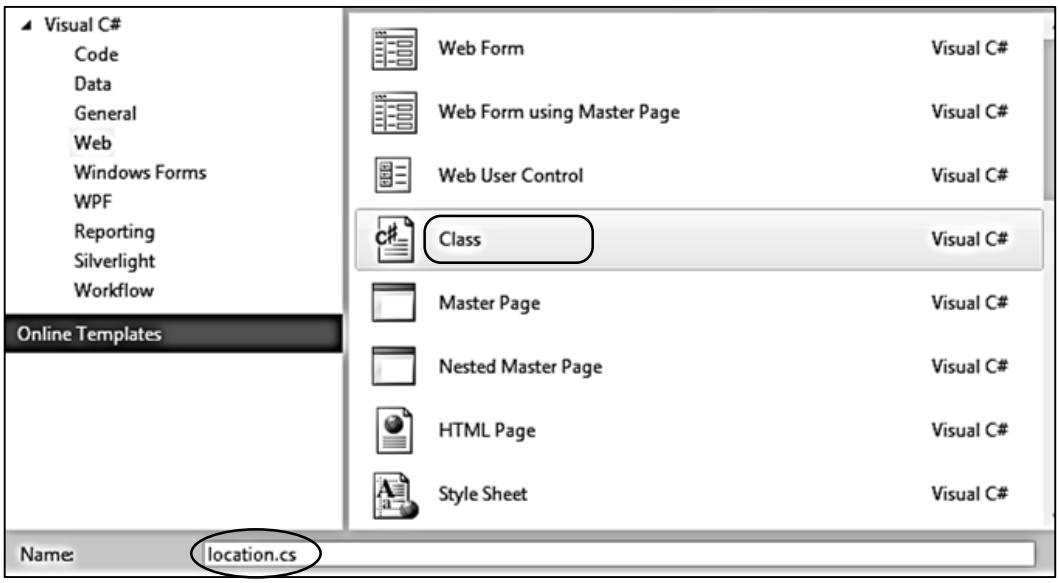


Save the table with the name '***locations***'.

Right click the ***locations*** table icon and select '***Show table data***'. Add test data entries for the two locations which you selected from the map, along with the corner coordinates for the map rectangles. **LocationID** values will be inserted automatically by the computer.

locationID	locationName	locationText	locationImage	XtopLeft	YtopLeft	XbottomRight	YbottomRight
1	Llangollen	Test information Llangollen.	*NULL*	3	124	103	145
2	Trevor	Test information Trevor.	*NULL*	164	118	206	134
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

The next step is to set up a ***class*** file to produce location ***objects*** for use by the web page. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name '**location**'.



Open the ***location*** class file. Add code to load records and create a set of location ***objects***.

```
using System.Web;

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

namespace Canal_holidays
{
   public class location
   {
      public static int locationCount = 0;
      public static location[] locationObject = new location[100];
      public static string databaseLocation =
                                  "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";
      public int locationID { get; set; }
      public string locationName { get; set; }
      public string locationText { get; set; }
      public int XtopLeft { get; set; }
      public int YtopLeft { get; set; }
      public int XbottomRight { get; set; }
      public int YbottomRight { get; set; }
      public bool imageLoaded { get; set; }

      public static void loadLocations()
      {
         DataSet dsLocations = new DataSet();
         SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
               AttachDbFilename=" + databaseLocation + "Integrated Security=True;
               Connect Timeout=30; User Instance=True");
         try
         {
            cnTB.Open();
            SqlCommand cmLocations = new SqlCommand();
            cmLocations.Connection = cnTB;
            cmLocations.CommandType = CommandType.Text;
            cmLocations.CommandText = "SELECT * FROM locations";
            SqlDataAdapter daLocations = new SqlDataAdapter(cmLocations);
            daLocations.Fill(dsLocations);
            cnTB.Close();
            int countRecords = dsLocations.Tables[0].Rows.Count;
            locationCount = 0;
            for (int i = 0; i < countRecords; i++)
            {
               DataRow drLocations = dsLocations.Tables[0].Rows[i];
               int locationID = (int)drLocations[0];
               string locationName = Convert.ToString(drLocations[1]);
               string locationText = Convert.ToString(drLocations[2]);
               int XtopLeft = Convert.ToInt16(drLocations[4]);
               int YtopLeft = Convert.ToInt16(drLocations[5]);
               int XbottomRight = Convert.ToInt16(drLocations[6]);
               int YbottomRight = Convert.ToInt16(drLocations[7]);
```

```
            bool pictureLoaded = true;
            if (drLocations.IsNull("locationImage"))
            {
               pictureLoaded = false;
            }
            locationObject[locationCount] = new location();
            locationObject[locationCount].locationID = locationID;
            locationObject[locationCount].locationName = locationName;
            locationObject[locationCount].locationText = locationText;
            locationObject[locationCount].XtopLeft = XtopLeft;
            locationObject[locationCount].YtopLeft = YtopLeft;
            locationObject[locationCount].XbottomRight = XbottomRight;
            locationObject[locationCount].YbottomRight = YbottomRight;
            locationObject[locationCount].imageLoaded = pictureLoaded;
            locationCount++;
        }
      }
      catch
      {

      }
    }
  }
}
```

Go now to the ***route.aspx*** page and add ***divisions*** and ***labels*** to the **content2** section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="routeMap">
      <asp:Label ID="Label1" runat="server" ></asp:Label>
    </div>
    <div id="locationPhoto">
      <asp:Label ID="Label2" runat="server" Text=""></asp:Label>
    </div>
    <div id="routeInformation">
      <asp:Label ID="Label3" runat="server" Text=""></asp:Label>
    </div>
</asp:Content>
```

Open the ***Style Sheet*** and add a formatting block for these divisions.

```
#routeMap
{
    float:left;
}
#locationPhoto
{
    float: right;
    width: 400;
}
```

```
#routeInformation
{
    float: left;
    width: 900;
    text-align:left;
}
```

We can now produce the HTML code which will be inserted into the page by means of the label component. To set up an image map which can be clicked to select other web pages, we will need the following commands:

<img src='Images/canal map.png' usemap=#canalmap>

specifies the location of the graphics file which we are going to display.

<map name=canalmap>

indicates that an image map with clickable areas is to be defined. Each active rectangle on the map is set up using a line of code similar to:

<area shape=rect coords=164, 118, 206, 134 href="route.aspx?location=Trevor" />

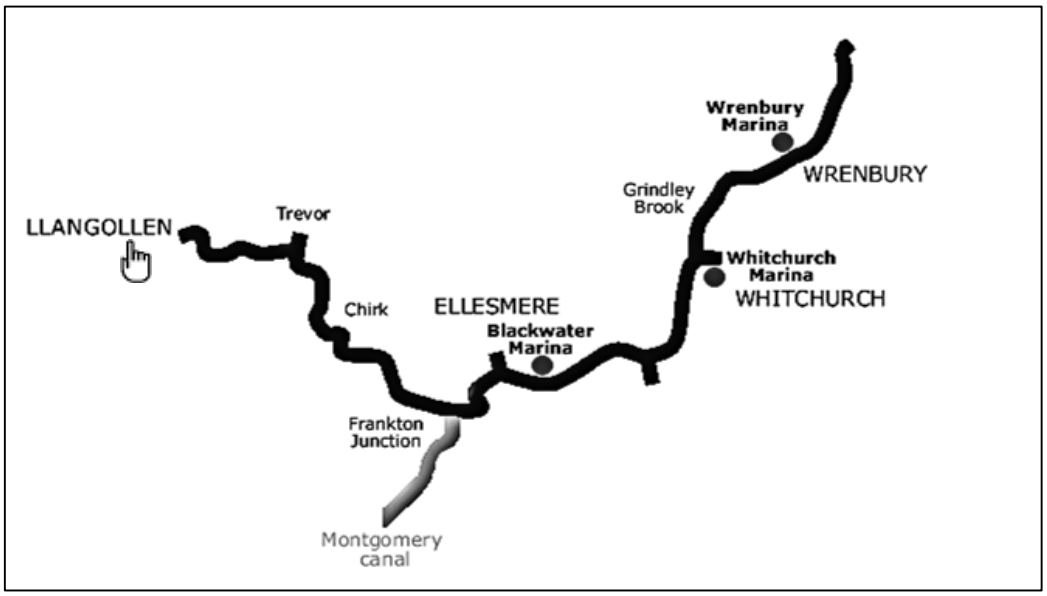
rectangle corner coordinates

information to load

Open the C# code window for the ***route.aspx*** page and add lines of program code to the ***Page_Load*** method. These load the location records from the database, create location ***objects***, then set up the ***image map*** with a clickable area for each location as described above. Note that each command beginning: `s += "...` should be entered as a single line of code with no line breaks.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (location.locationCount < 1)
    {
        location.loadLocations();
    }
    string s ="";
    s +="<img src='Images/canal map.png' usemap=#canalmap border=0 />";
    s += "<map name=canalmap>";
    for (int i = 0; i < location.locationCount; i++)
    {
        s += "<area shape=rect coords=";
        s += location.locationObject[i].XtopLeft + "," +
                location.locationObject[i].YtopLeft + "," +
                location.locationObject[i].XbottomRight + "," +
                location.locationObject[i].YbottomRight;
        s += " href='route.aspx?location=" +
                location.locationObject[i].locationName + "' />";
    }
    s += "</map>";
    Label1.Text = s;
}
```

Build and run the ***route.aspx*** page. The canal route is displayed. Notice that if you move the mouse onto one of the locations which you entered in the database as test data, the normal arrow pointer changes to a '***pointing finger***' icon.



Click the mouse while the 'pointing finger' icon is displayed, and check the page URL in the address bar at the top of the screen. This should show the page name '***route.aspx***' followed by the name of the selected location as the value of the variable '***location***'.

Close the browser and stop debugging. At this stage it will be useful to set up the page in the staff website where employees of Canal Boat Holidays will be able to enter and edit locations.

Open the Style Sheet and add two formatting blocks for divisions.

```
#locationContent
 {
     width: 1040px;
     height: 960px;
     background-color: #FFFFFF;
     padding:20px;
 }

#locationList
{
    height: 140px;
    width: 1000px;
    background-color: white;

}
```

Open the ***staffRoute.aspx*** file and add lines of HTML code to the '***content2***' section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="locationContent">
    <div id="locationList">
      <table border="0" cellpadding="10">
        <tr>
          <td>
            Location name
          </td>
        <td>
           <asp:DropDownList ID="ddlLocations" runat="server" Width="300px">
           </asp:DropDownList>
        </td>
        <td>
           <asp:Button ID="btnDelete" runat="server" Text="Delete location" />
        </td>
        <td>
           <asp:Button ID="btnNew" runat="server" Text="New location" />
        </td>
      </tr>
      <tr>
        <td colspan="4">
          <center>
             <asp:Label ID="lblMessage" runat="server" Font-Bold="True"
                                                  ForeColor="red"></asp:Label>
          </center>
        </td>
       </tr>
     </table>
   </div>
  </div>
</asp:Content>
```

Change to the design view by clicking the ***Design*** button. The screen should show:

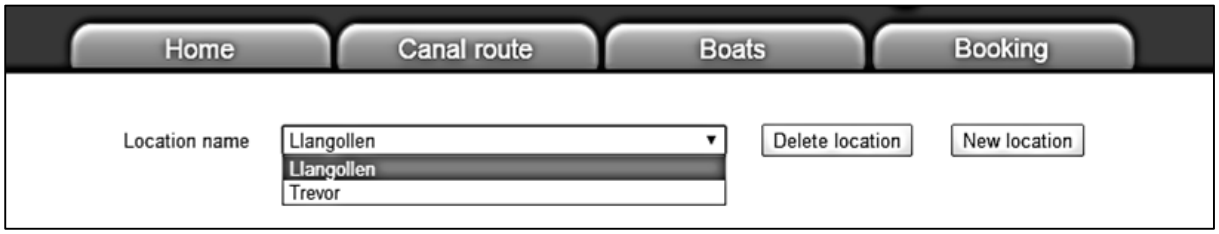
- A drop down list, which can be used to select a location object for editing.
- A button to delete a location object no longer required.
- A button to create a new location object.

A label has also been created, which will be used to display messages for the user.

Open the C# code page for ***staffRoute.aspx***. Add lines of program to ***Page_Load***, and produce a ***loadData( )*** method.

```
public partial class staffRoute : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (ddlLocations.Items.Count == 0)
        {
            location.loadLocations();
            loadData();
        }
    }

    protected void loadData()
    {
        ddlLocations.Items.Clear();
        for (int i = 0; i < location.locationCount; i++)
        {
            ddlLocations.Items.Add(location.locationObject[i].locationName);
        }
    }
```

Build and run the ***staffRoute*** web page. Check that the test locations are displayed in the drop down list.

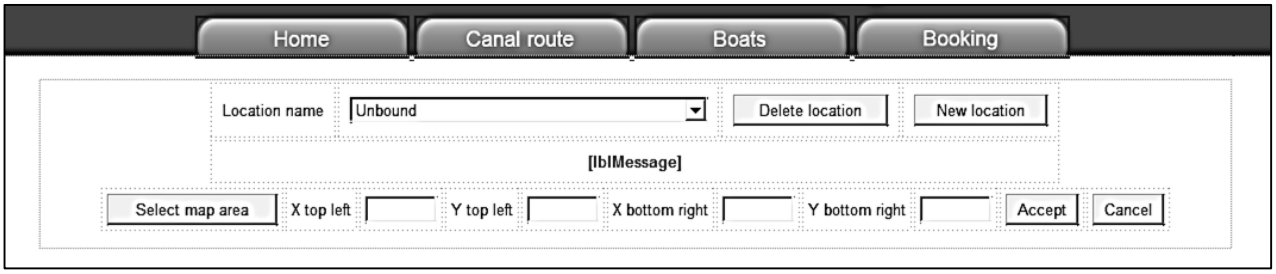


Close the browser, return to the ***staffRoute.aspx*** page and stop debugging. Add lines of HTML code to the ***content2*** section.

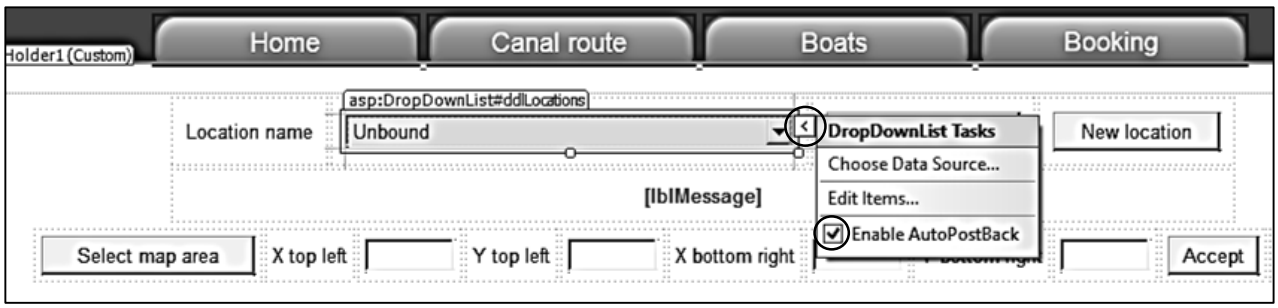
```
    <td colspan="4">
     <center>
         <asp:Label ID="lblMessage" runat="server" Font-Bold="True"
                                                 ForeColor="red"></asp:Label>
     </center>
    </td>
   </tr>
</table>

<table cellpadding="5">
 <tr>
  <td><asp:Button ID="btnMapArea" runat="server" Text="Select map area" /></td>
  <td>X top left</td>
  <td>
     <asp:TextBox ID="txtXtopLeft" runat="server" Width="60px"></asp:TextBox>
  </td>
  <td>Y top left</td>
```

```
      <td>
         <asp:TextBox ID="txtYtopLeft" runat="server" Width="60px"></asp:TextBox>
      </td>
      <td>X bottom right</td>
      <td>
        <asp:TextBox ID="txtXbottomRight" runat="server" Width="60px"></asp:TextBox>
      </td>
      <td>Y bottom right</td>
      <td>
        <asp:TextBox ID="txtYbottomRight" runat="server" Width="60px"></asp:TextBox>
      </td>
      <td>
        <asp:Button ID="btnAccept" runat="server" Text="Accept" Visible="False" />
      </td>
      <td>
        <asp:Button ID="btnCancel" runat="server" Text="Cancel" Visible="False" />
      </td>
     </tr>
    </table>
   </div>
  </div>
</asp:Content>
```

Change to the design view by clicking the Design button. A set of text boxes has been created for displaying the corner coordinates of the active clickable area of the image map.



Use the mouse to select the ***drop down list*** component, then click the small arrow icon in the top right corner to open the task menu. Click the mouse to tick the box for '***Enable AutoPostBack***'.



Double click the ***drop down list*** component to create a ***Selected Index Changed*** method. This method will operate whenever the user choses an item from the drop down list. Add lines of code to the method.

```
protected void ddlLocations_SelectedIndexChanged(object sender, EventArgs e)
{
    int recordIDwanted = ddlLocations.SelectedIndex;
    displayRecord(recordIDwanted);
    lblMessage.Text = "";
}
```

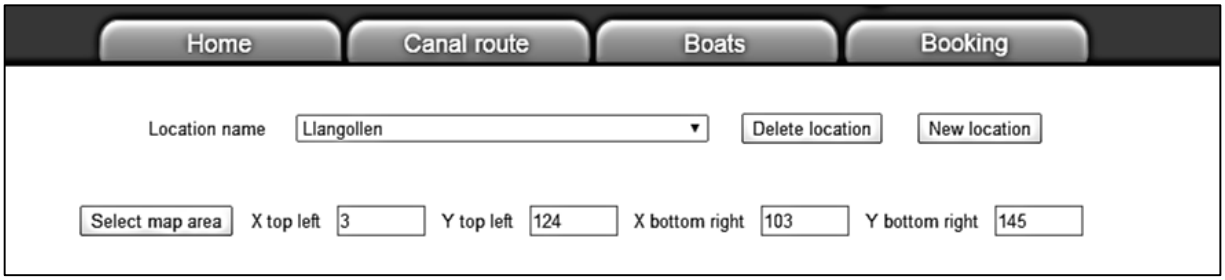
Move up to the top of the C# code page. Add variables at the start of the ***staffRoute*** class, and a line of code in the ***Page_Load*** method.

```
public partial class staffRoute : System.Web.UI.Page
{
    public static int topLeftX = 0;
    public static int topLeftY = 0;
    public static int bottomRightX = 0;
    public static int bottomRightY = 0;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (ddlLocations.Items.Count == 0)
        {
            location.loadLocations();
            loadData();
            displayRecord(0);
        }
    }
```

Insert a ***displayRecord( )*** method immediately after the ***Page_Load( )*** method.

```
protected void displayRecord(int recordID)
{
        topLeftX = location.locationObject[recordID].XtopLeft;
        topLeftY = location.locationObject[recordID].YtopLeft;
        bottomRightX = location.locationObject[recordID].XbottomRight;
        bottomRightY = location.locationObject[recordID].YbottomRight;
        txtXtopLeft.Text = Convert.ToString(topLeftX);
        txtYtopLeft.Text = Convert.ToString(topLeftY);
        txtXbottomRight.Text = Convert.ToString(bottomRightX);
        txtYbottomRight.Text = Convert.ToString(bottomRightY);
}
```

Build and run the ***staffRoute*** web page. Check that the image map coordinates are displayed when each of the test locations are selected from the drop down list.



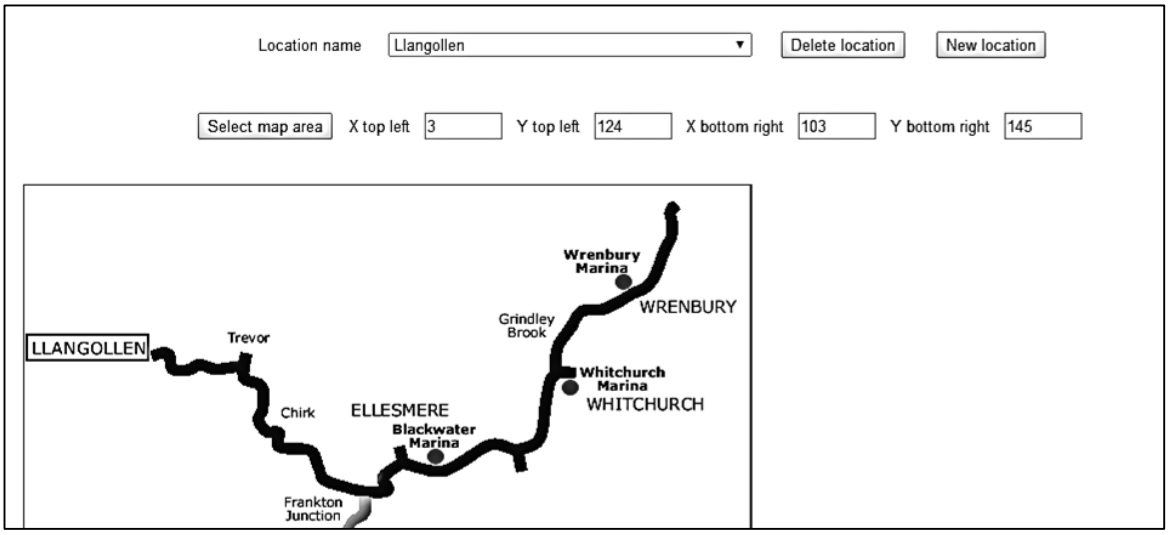
Close the browser, return to the ***staffRoute.aspx*** page and stop debugging. Add a ***staffRouteMap*** division and create a ***Canvas*** drawing area. We will also add some Javascript code.

```
        <td>
         <asp:Button ID="btnCancel" runat="server" Text="Cancel" Visible="False" />
        </td>
       </tr>
      </table>
    </div>

    <div id="staffRouteMap">
        <canvas id="myCanvas" width="1000" height="360"></canvas>
        <script>
            var canvas = document.getElementById('myCanvas');
            var context = canvas.getContext('2d');
            var Xtop;
            var Ytop;
            var Xbottom;
            var Ybottom;
            var Rwidth;
            var Rheight;
            var corner = 1;

            function setup() {
                var Xtop = '<%= topLeftX %>';
                var Ytop= '<%= topLeftY %>';
                var Xbottom = '<%= bottomRightX %>';
                var Ybottom = '<%= bottomRightY %>';
                var Rwidth = Xbottom - Xtop;
                var Rheight = Ybottom - Ytop;
                context.fillStyle = 'black';
                context.strokeRect(0, 0, 600, 360);
                context.strokeRect(Xtop, Ytop, Rwidth, Rheight);
            }

           setup();
        </script>
    </div>
  </div>
</asp:Content>
```

Open the Style Sheet and add formatting code for the staffRouteMap division.

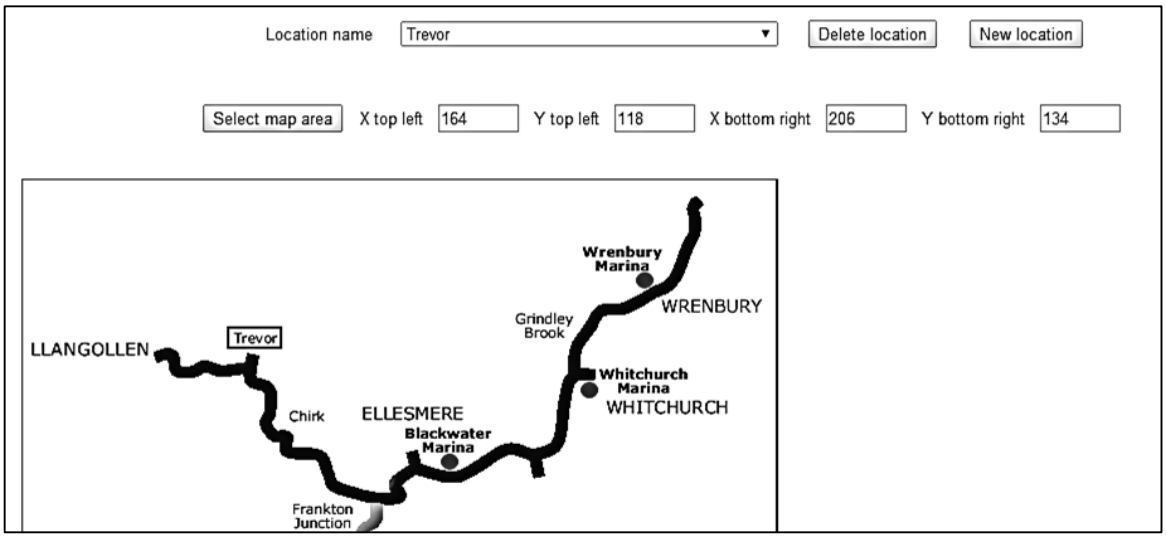
```
#staffRouteMap
{
    height: 360px;
    text-align: left;
    position:relative;
    background-image: url('Images/canal map.png');
    background-repeat: no-repeat;
    width: 1040px;
}
```

Build and run the ***staffRoute*** web page.



The canal route map has been displayed as a background image for the ***staffRouteMap*** division, and a drawing canvas placed over it.

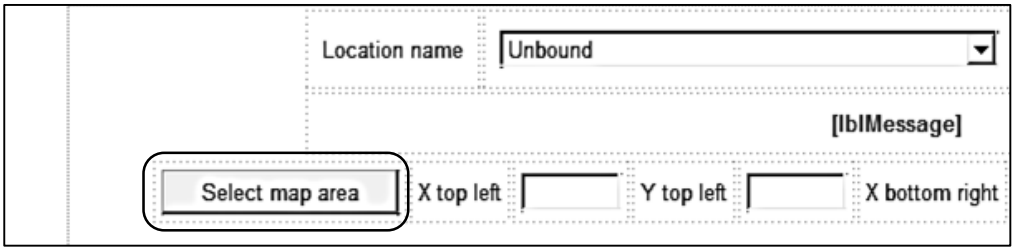
When the page opens, the first ***location*** object is loaded and its image map coordinates are displayed in the text boxes. The Javascript code takes these values and draws a rectangle on the map to show the position of the active area.Use the drop down list to select the second test location, and the rectangle should now move to the new coordinates.



Close the browser and stop debugging.

We will now add Javascript functions to allow the user to easily set the image map coordinates for a location by clicking the mouse at the required positions on the canal route map.

Go to the ***staffRoute.aspx*** page and select the ***Design*** view. Double click the '***Select map area***' button to create a button_click method.



Add lines of code to the ***btnMapArea_Click*** method. Note that the line beginning:

lblMessage.Text = " . . .

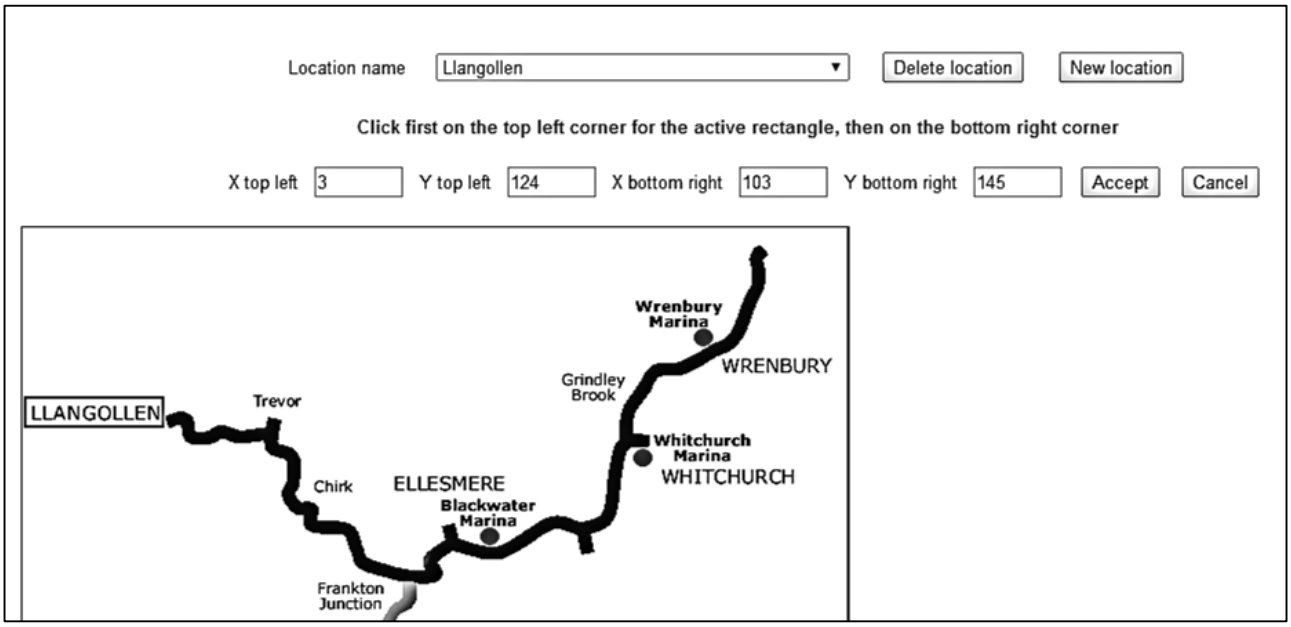
should be entered without a single command without a line break.

```
protected void btnMapArea_Click(object sender, EventArgs e)
{
    lblMessage.Text = "Click first on the top left corner for the active
                                rectangle, then on the bottom right corner";
    btnAccept.Visible = true;
    btnCancel.Visible = true;
    btnMapArea.Visible = false;
    mapSelecting = "yes";
}
```

Add the variable '***mapSelecting***' at the start of the class.

```
public partial class staffRoute : System.Web.UI.Page
{
    public static int topLeftX = 0;
    public static int topLeftY = 0;
    public static int bottomRightX = 0;
    public static int bottomRightY = 0;
    public static string mapSelecting = "no";
```

Build and run the ***staffRoute.aspx*** web page. Click the '***Select map area***' button. The screen display changes to show an instruction message to the user. Check that '***Accept***' and '***Cancel***' buttons appear correctly.



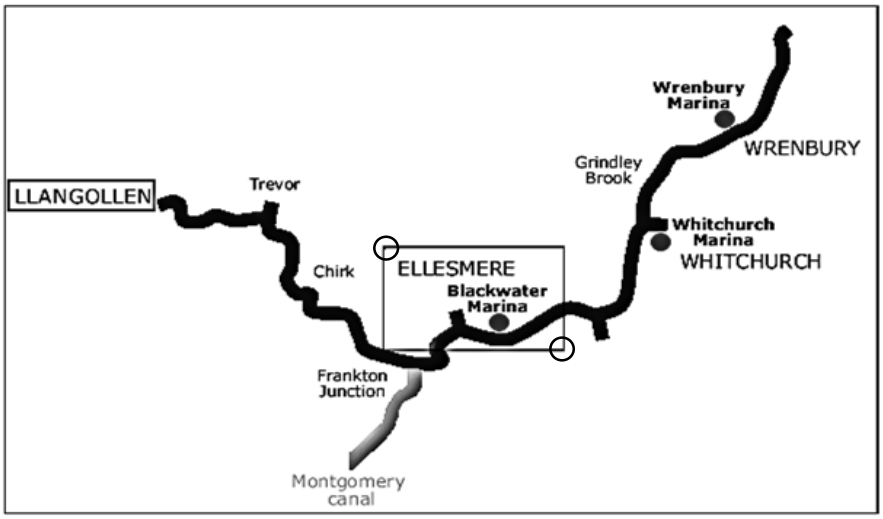
Close the web browser and stop debugging. Return to the ***staffRoute.aspx*** page and insert further Javascript code in the page Source view to allow staff to enter the position of the interactive rectangle.

```
function setup() {
    var Xtop = '<%= topLeftX %>';
    var Ytop= '<%= topLeftY %>';
    var Xbottom = '<%= bottomRightX %>';
    var Ybottom = '<%= bottomRightY %>';
    var Rwidth = Xbottom - Xtop;
    var Rheight = Ybottom - Ytop;
    context.fillStyle = 'black';
    context.strokeRect(0, 0, 600, 360);
    context.strokeRect(Xtop, Ytop, Rwidth, Rheight);
}
```

```
function getMousePos(canvas, evt) {
    var rect = canvas.getBoundingClientRect();
    return {
        x: evt.clientX - rect.left,
        y: evt.clientY - rect.top
    };
}

canvas.addEventListener('mousedown', function (evt) {
    var mousePos = getMousePos(canvas, evt);
    var selecting = '<%= mapSelecting %>';
    var context = canvas.getContext('2d');
    context.fillStyle = 'black';
```

```
        if (selecting == "yes") {
           if (corner == 1) {
               Xtop = mousePos.x;
               Ytop = mousePos.y;
               context.strokeRect(Xtop, Ytop, 1, 1);
           }
           else {
               Rwidth = mousePos.x - Xtop;
               Rheight = mousePos.y - Ytop;
               context.strokeRect(Xtop, Ytop, Rwidth, Rheight);
           }
           corner++;
           if (corner > 2)
               corner = 1;
       }
   }, false);

   setup();
</script>
```

Build and run the ***staffRoute*** web page. Choose one of the test locations and click the '***Select map area***' button. Move the mouse to the map image and click twice to position the upper left and lower right corners of a rectangle.



Close the browser, return to the ***staffRoute.aspx*** page and stop debugging.

Examine the way in which the Javascript code is working:

- When the mouse is clicked, the ***getMousePos( )*** function obtains the **x** and **y** coordinates of the mouse pointer on the map.
- The ***corner*** variable was initialised to '1'. The ***mousedown*** event function realises that the top left corner of the required rectangle has been selected, so puts a dot at this position. The value of ***corner*** is then incremented to '2'.

- When the mouse is clicked for a second time, the ***mousedown*** event function knows that the bottom right corner of the rectangle has been clicked, so adds a border line in the correct position.

The final step is to transfer the corner coordinates of the rectangle back to the C# code page so that the location object can be updated. To do this, we will need to add four '***hidden fields***' to the page.

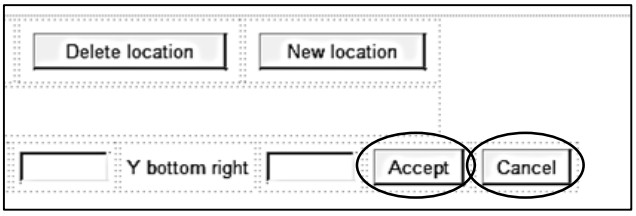
Go to the end of the ***staffRoute.aspx*** page and add lines of code.

```
            setup();
          </script>
        </div>
      </div>

    <asp:HiddenField id="hfXtop" runat="server" />
    <asp:HiddenField ID="hfYtop" runat="server" />
    <asp:HiddenField ID="hfXbottom" runat="server" />
    <asp:HiddenField ID="hfYbottom" runat="server" />

    </asp:Content>
```

Add lines to the Javascript ***mousedown*** event function to store the corner coordinates of the selected rectangle in the ***hidden fields***.

```
canvas.addEventListener('mousedown', function (evt) {
    var mousePos = getMousePos(canvas, evt);
    var selecting = '<%= mapSelecting %>';
    var context = canvas.getContext('2d');
    context.fillStyle = 'black';
    if (selecting == "yes") {
        if (corner == 1) {
            Xtop = mousePos.x;
            Ytop = mousePos.y;
            context.strokeRect(Xtop, Ytop, 1, 1);
            document.getElementById('<%= hfXtop.ClientID %>').value = Xtop;
            document.getElementById('<%= hfYtop.ClientID %>').value = Ytop;
        }
        else {

            Rwidth = mousePos.x - Xtop;
            Rheight = mousePos.y - Ytop;
            context.strokeRect(Xtop, Ytop, Rwidth, Rheight);
            document.getElementById('<%= hfXbottom.ClientID %>').value = mousePos.x;
            document.getElementById('<%= hfYbottom.ClientID %>').value = mousePos.y;
        }
        corner++;
        if (corner > 2)
            corner = 1;
    }
}, false);
```

Return to the Design view and double click the '***Accept***' and '***Cancel***' buttons to create button_click methods.

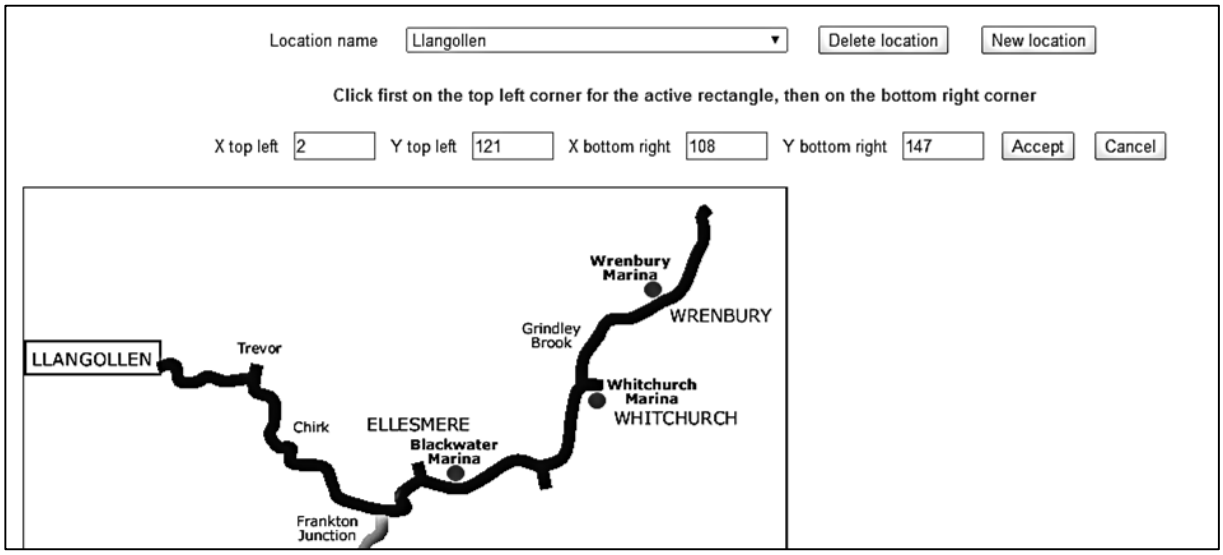


Add code to the ***btnAccept_Click( )*** and ***btnCancel_Click( )*** methods, then create a ***resetMap( )*** method.

```
protected void btnAccept_Click(object sender, EventArgs e)
{
    double tlX = Convert.ToDouble(hfXtop.Value);
    double tlY = Convert.ToDouble(hfYtop.Value);
    double brX = Convert.ToDouble(hfXbottom.Value);
    double brY = Convert.ToDouble(hfYbottom.Value);

    topLeftX = Convert.ToInt16(tlX);
    topLeftY = Convert.ToInt16(tlY);
    bottomRightX = Convert.ToInt16(brX);
    bottomRightY = Convert.ToInt16(brY);

    txtXtopLeft.Text = Convert.ToString(topLeftX);
    txtYtopLeft.Text = Convert.ToString(topLeftY);
    txtXbottomRight.Text = Convert.ToString(bottomRightX);
    txtYbottomRight.Text = Convert.ToString(bottomRightY);

    resetMap();
}

protected void btnCancel_Click(object sender, EventArgs e)
{
    resetMap();
}

protected void resetMap()
{
    btnAccept.Visible = false;
    btnCancel.Visible = false;
    btnMapArea.Visible = true;
    lblMessage.Text = "";
    mapSelecting = "no";
}
```

The ***btn_Accept*** method collects the **x** and **y** coorodinate values from the hidden fields, but at this stage they may contain decimal fractions (e.g. 126.5). We convert these to integers before transferring them to the text box display.

Build and run the staffRoute web page.



Carefully check the functionality of the map by carrying out a series of tests:

- Canal locations can be selected by means of the drop down list. Their active rectangles are shown in the correct positions on the route map.
- Clicking the mouse on the map has no effect unless the '***Select map area***' button has first been activated.
- A rectangle can be positioned by means of mouse clicks.
- The '***Accept***' button updates the coorodinates and shows the new rectangle in the selected position.
- The '***Cancel***' button does not update the coorodinates and the rectangle remains in its original location.

If all is well, we can now move on to input the information and photograph for the canal location. We will use similar methods to the staff Home page.

Close the web browser and stop debugging. Go to the ***staffRoute.aspx*** page and add lines of code after the block of Javascript.

```
      setup();
    </script>
  </div>
```

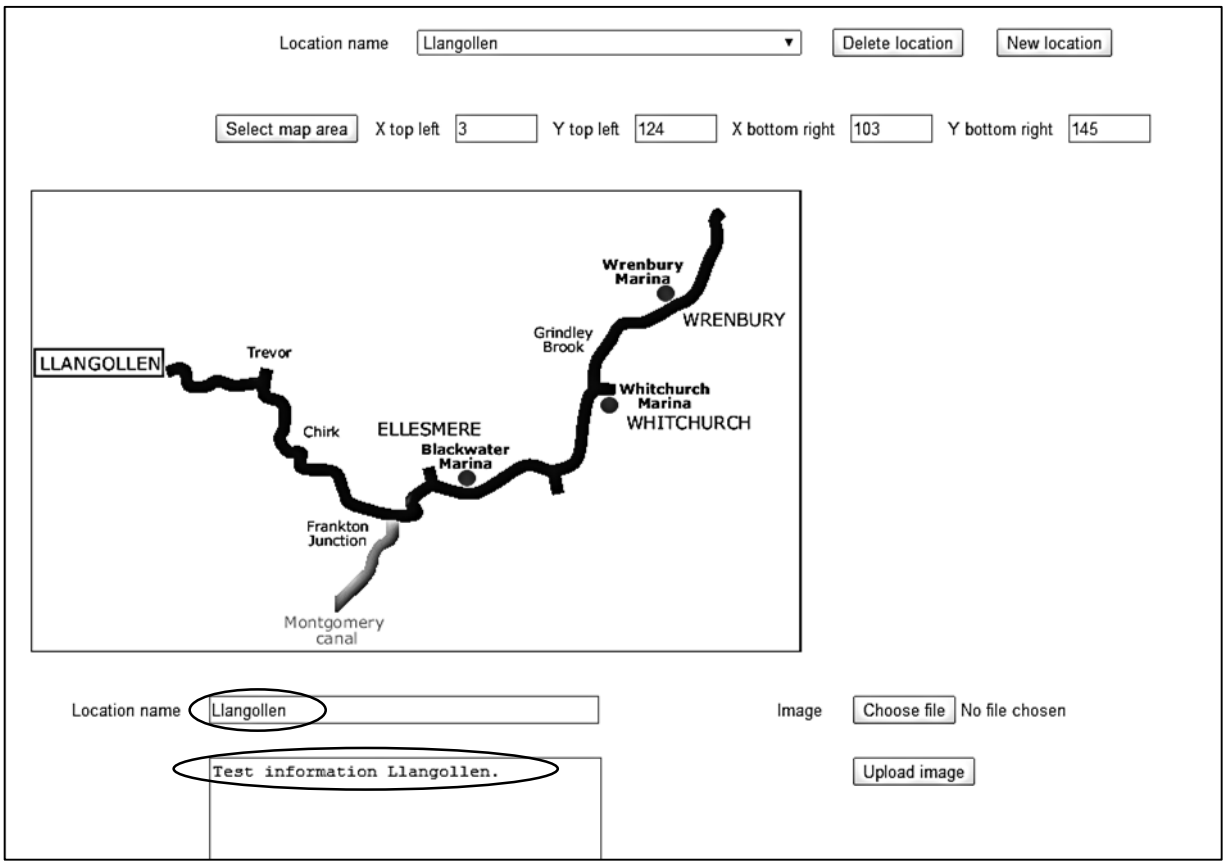
```
<div id="leftColumn">
  <table border="0" cellpadding="10">
    <tr>
      <td>Location name</td>
      <td>
        <asp:TextBox ID="txtLocationName" runat="server" Width="300px">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>Information</td>
```

```
        <td>
          <asp:TextBox ID="txtInformation" runat="server" Width="300px"
                               TextMode="MultiLine" Rows="18"></asp:TextBox>
        </td>

      </tr>
    </table>
  </div>

  <div id="rightColumn">
    <table border="0" cellpadding="10">
      <tr>
        <td>Image</td>
        <td>
            <asp:FileUpload ID="FileUpload1" runat="server" />
        </td>
      </tr>
      <tr>
        <td></td>
        <td>
           <asp:Button ID="upload" runat="server" Text="Upload image"  />
        </td>
      </tr>
    </table>
    <br />
    <asp:Image ID="Image1" runat="server"  width='300px' />
   </div>
  </div>
  <asp:HiddenField id="hfXtop" runat="server" />
  <asp:HiddenField ID="hfYtop" runat="server" />
```

Move to the C# code page for ***staffRoute.aspx*** and add lines to the ***displayRecord( )*** method.

```
protected void displayRecord(int recordID)
{
        txtLocationName.Text = location.locationObject[recordID].locationName;
        txtInformation.Text = location.locationObject[recordID].locationText;
        topLeftX = location.locationObject[recordID].XtopLeft;
        topLeftY = location.locationObject[recordID].YtopLeft;
        bottomRightX = location.locationObject[recordID].XbottomRight;
        bottomRightY = location.locationObject[recordID].YbottomRight;
```

Build and run the staffRoute web page. Select the example canal locations and check that location names and information are displayed in the text boxes below the map



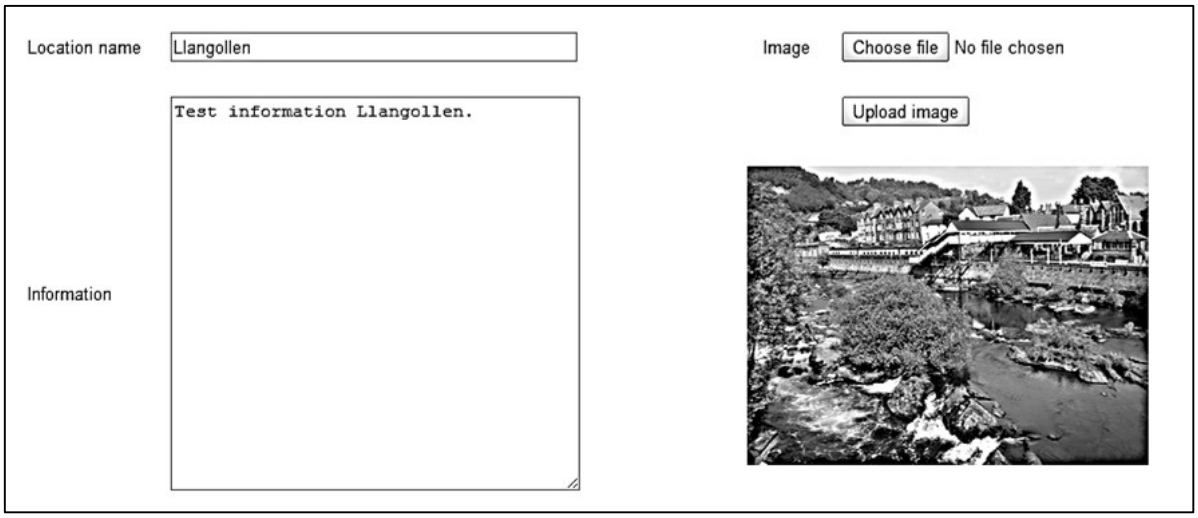
Close the browser and stop debugging. Go to the Design view for the ***staffRoute.aspx*** page and double click the '***Upload image***' button. Add lines of code to the ***upload_Click( )*** method.

```
protected void upload_Click(object sender, EventArgs e)
{
    try
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(Server.MapPath("Images").ToString() + @"\" +
            FileUpload1.FileName);
            Image1.ImageUrl = @"Images\" + FileUpload1.FileName;
            picbyte = FileUpload1.FileBytes;
        }
        else
        {
            lblMessage.Text = "Please load an image";
        }
    }
    catch
    {
    }
}
```

Scroll up to the top of the C# code page and add the variable '***picbyte***' at the start of the class.

```
public partial class staffRoute : System.Web.UI.Page
{
    public static byte[] picbyte;

    public static int topLeftX = 0;
    public static int topLeftY = 0;
```

Build and run the staffRoute web page to check that a photograph can be selected and displayed. Note that this may not work in a ***Firefox*** browser due to security restrictions on handling images on local systems.



Close the browser and stop debugging. We now need to create a method to save the updated ***location*** object into the database table. Begin by adding a button to the ***staffRoute.aspx*** page at the end of the '***leftColumn***' division.

```
        <asp:TextBox ID="txtInformation" runat="server" Width="300px"
                 TextMode="MultiLine" Rows="18"></asp:TextBox>
         </td>
      </tr>
    </table>

    <center>
       <asp:Button ID="btnSave" runat="server" Text="Save record" />
    </center>

  </div>

  <div id="rightColumn" style="float: right">
   <table border="0" cellpadding="10">
     <tr>
        <td>
           Image
```

Go now to the ***location.cs*** class file and add an ***updateLocation( )*** method near the start of the class. Note that the lines beginning:

public static string updateLocation(... , SqlConnection cnTB = new SqlConnection(...

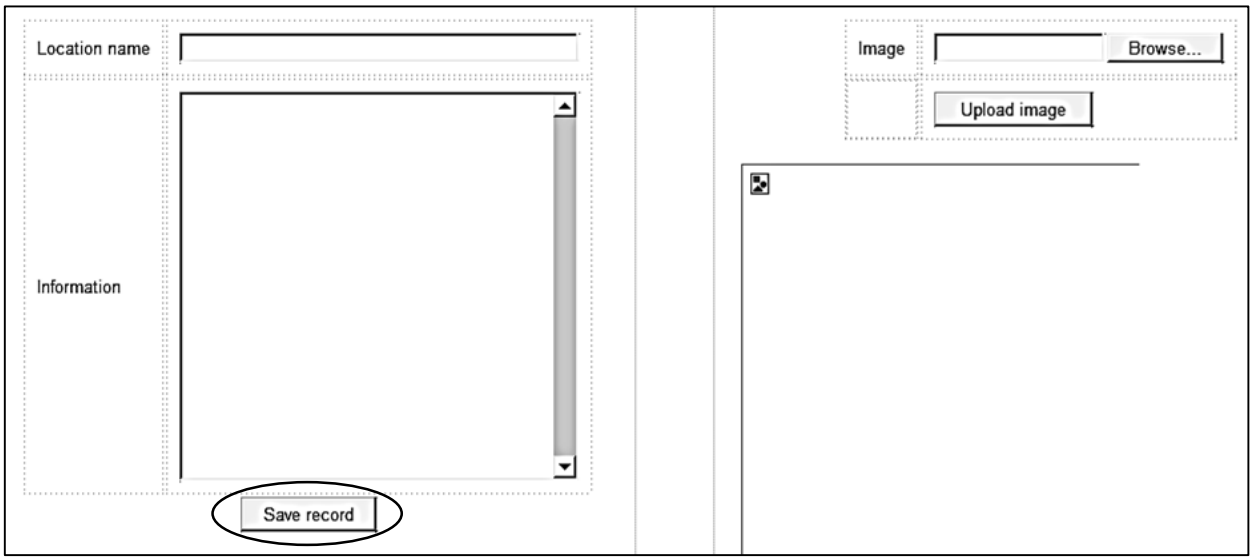
query = "UPDATE locations ...

should be entered as single commands with no line breaks.

```
public int XbottomRight { get; set; }
public int YbottomRight { get; set; }
```

```
public static string updateLocation(byte[] picbyte, string locationName,
         string locationText, int XtopLeft, int YtopLeft, int XbottomRight,
         int YbottomRight, int locationSelected)
{
    string message = "";
    string query = "";
    SqlParameter picparameter = new SqlParameter();
    picparameter.SqlDbType = SqlDbType.Image;
    picparameter.ParameterName = "pic";
    picparameter.Value = picbyte;
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
       AttachDbFilename=" + databaseLocation +"Integrated Security=True;
       Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmStock = new SqlCommand();
        cmStock.Connection = cnTB;
        cmStock.CommandType = CommandType.Text;
        if (picbyte == null)
        {
           query = "UPDATE locations SET locationName='" + locationName +
               "', locationText='" + locationText + "', XtopLeft='" + XtopLeft +
               "', YtopLeft='" + YtopLeft + "', XbottomRight='" + XbottomRight +
               "', YbottomRight='" + YbottomRight + "' WHERE locationID = '" +
               locationSelected + "'";
        }
        else
        {
           query = "UPDATE locations SET locationName='" + locationName +
               "', locationText='" + locationText + "', XtopLeft='" + XtopLeft +
               "', YtopLeft='" + YtopLeft + "', XbottomRight='" + XbottomRight +
               "', YbottomRight='" + YbottomRight + "',locationImage= @pic
               WHERE locationID = '" + locationSelected + "'";
        }
        SqlCommand cmd = new SqlCommand(query, cnTB);
        if (!(picbyte == null))
        {
            cmd.Parameters.Add(picparameter);
        }
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record saved";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

Return now to the ***staffRoute.aspx*** page and change to the Design view. Double click the '***Save record***' button to create a button_click method.



Add a lines of code to the ***btnSave_Click( )*** method. The line beginning: '***lblMessage.Text = …***' is a single command, which should be entered with no line breaks.

```
protected void btnSave_Click(object sender, EventArgs e)
{
    string information = txtInformation.Text;
    information = information.Replace("'", "`");
    lblMessage.Text=location.updateLocation(picbyte,txtLocationName.Text,
            txtInformation.Text,topLeftX,topLeftY, bottomRightX,
            bottomRightY,locationSelected);
    location.loadLocations();
    loadData();
}
```

Move to the top of the C# code page, and add an additional variable at the start of the ***staffRoute*** class.

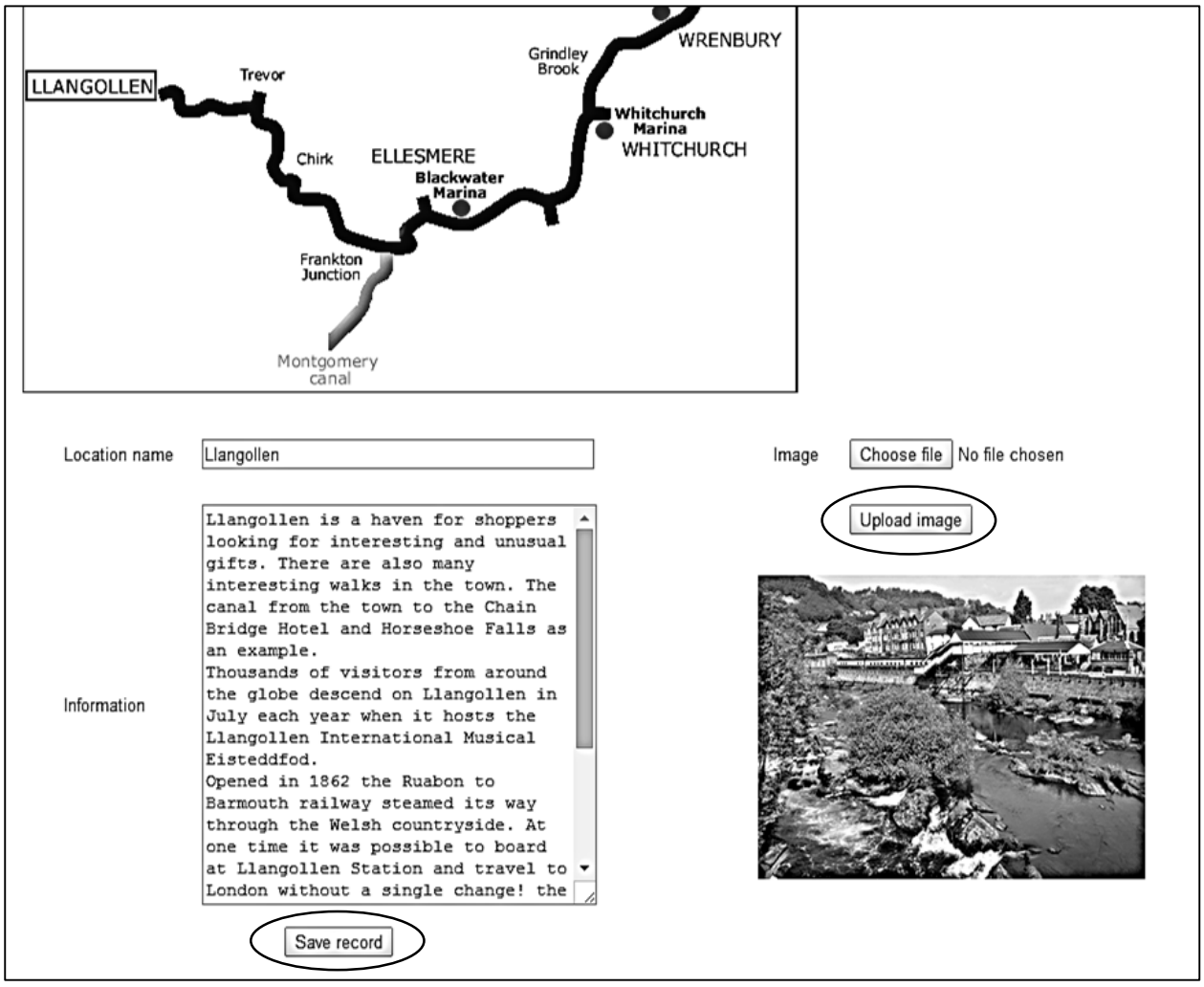
```
public partial class staffRoute : System.Web.UI.Page
{
    public static byte[] picbyte;
    public static int locationSelected;
    public static int topLeftX = 0;
    public static int topLeftY = 0;
    public static int bottomRightX = 0;
```

The '***picbyte***' variable is an array which will hold the digitised picture image. The variable '***locationSelected***' will store the ID number of the location being edited, so that the correct record can be updated in the database.

We finally have to complete the ***displayRecord( )*** method by adding a line of code to record the ID number of the selected location.

```
protected void displayRecord(int recordID)
{
    for (int i = 0; i < location.locationCount; i++)
    {
        txtLocationName.Text = location.locationObject[recordID].locationName;
        txtInformation.Text = location.locationObject[recordID].locationText;

        locationSelected = location.locationObject[recordID].locationID;

        topLeftX = location.locationObject[recordID].XtopLeft;
        topLeftY = location.locationObject[recordID].YtopLeft;
```

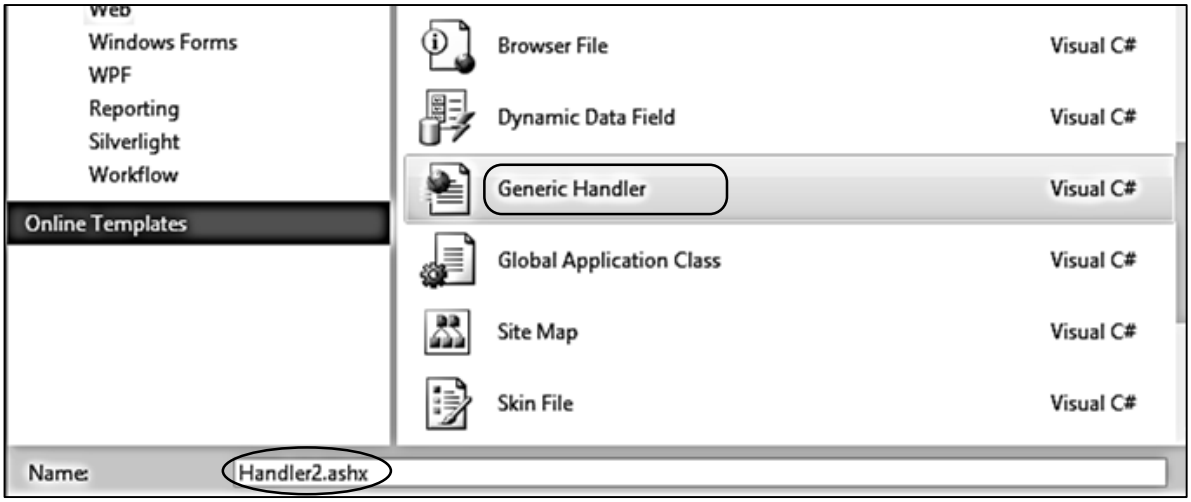
Build and run the staffRoute web page. Select one of the test locations. Edit the position of the image map active area if you wish. You may also edit the location name and text information. Select and upload a photograph of the location, then click the '***Save record***' button.



Close the browser and stop debugging. Go to the Server Explorer window and refresh the data. Open the ***locations*** table and check that the record has been updated. The uploaded image should appear as ***<Binary data>***.

locationID	locationName	locationText	locationImage	XtopLeft	YtopLeft	XbottomRight	YbottomRight
1	Llangollen	Llangollen is a haven for shoppers...	<Binary data>	3	124	103	145
2	Trevor	Test information Trevor.	*NULL*	164	118	206	134
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

To redisplay the photograph image, we require a Handler. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Generic Handler***. Accept the name '**Handler2**'.



Open the handler file and replace lines of code in ***ProcessRequest( )***. Note that the lines beginning:

conn = new System.Data.SqlClient.SqlConnection(...

sqlcmd = new System.Data.SqlClient.SqlCommand(...

should be entered as single commands with no line breaks.

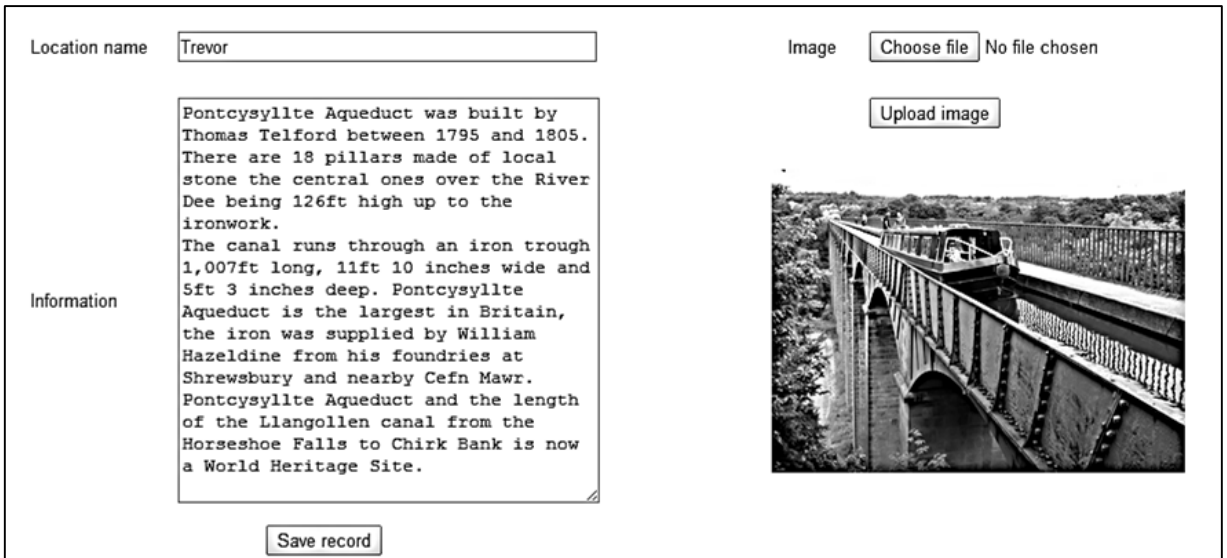
```
public class Handler2 : IHttpHandler
{
  string databaseLocation = "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";
  public void ProcessRequest(HttpContext context)
  {
    System.Data.SqlClient.SqlDataReader rdr = null;
    System.Data.SqlClient.SqlConnection conn = null;
    System.Data.SqlClient.SqlCommand sqlcmd = null;
    try
    {
       conn = new System.Data.SqlClient.SqlConnection(@"Data Source=.\SQLEXPRESS;
            AttachDbFilename=" + databaseLocation +"Integrated Security=True;
            Connect Timeout=30; User Instance=True");
       sqlcmd = new System.Data.SqlClient.SqlCommand("SELECT locationImage
            FROM locations WHERE locationID=" +
            context.Request.QueryString["imgid"], conn);
```

```
            conn.Open();
            rdr = sqlcmd.ExecuteReader();
            while (rdr.Read())
            {
               context.Response.ContentType = "image/jpg";
               context.Response.BinaryWrite((byte[])rdr["locationImage"]);
            }
            if (rdr != null)
              rdr.Close();
        }
        finally
        {
            if (conn != null)
                conn.Close();
        }
    }
```

Return to the ***staffRoute.aspx*** C# code page and add a line of code to the ***displayRecord( )*** method to load and display the photograph.

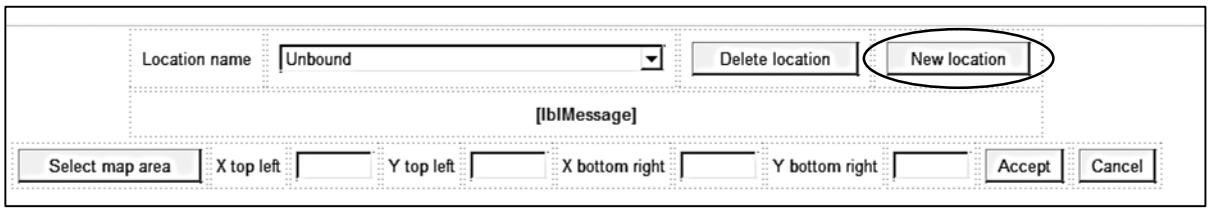
```
protected void displayRecord(int recordID)
{
   txtLocationName.Text = location.locationObject[recordID].locationName;
   txtInformation.Text = location.locationObject[recordID].locationText;
   locationSelected = location.locationObject[recordID].locationID;
   if (location.locationObject[recordID].imageLoaded == true)
   {
     Image1.ImageUrl = "Handler2.ashx?imgid="+ Convert.ToString(locationSelected);
   }
   else
   {
     Image1.ImageUrl = "";
   }
   topLeftX = location.locationObject[recordID].XtopLeft;
   topLeftY = location.locationObject[recordID].YtopLeft;
```

Build and run the ***staffRoute*** web page. Add information and a photograph for the second test location. Check that photographs are now displayed when either test record is selected.



Close the web browser and stop debugging. We can successfully edit existing canal locations. We now need to add new locations.

Go to the ***staffRoute*** design view and double click the '***New location***' button to create a button_click method.



Add lines of code the the ***btnNew_Click( )*** method to blank out the current data and leave the screen ready for entry of the new record.

```
protected void btnNew_Click(object sender, EventArgs e)
{
    txtXtopLeft.Text = "";
    txtYtopLeft.Text = "";
    txtXbottomRight.Text = "";
    txtYbottomRight.Text = "";
    txtLocationName.Text = "";
    txtInformation.Text = "";
    picbyte = null;
    topLeftX = 0;
    topLeftY = 0;
    bottomRightX = 0;
    bottomRightY =0;
    Image1.ImageUrl = "";
    newRecord = true;
}
```

Move to the top of the C# page and add a newRecord variable at the start of the class. This variable will be used to specify whether a new record is being added or whether an existing record is being edited.

```
public partial class staffRoute : System.Web.UI.Page
{
    public static byte[] picbyte;
    public static int locationSelected;
    public static bool newRecord = false;
    public static int topLeftX = 0;
    public static int topLeftY = 0;
```

Open the ***location.cs*** class file and insert an ***addLocation( )*** method. Note that the lines beginning:

public static string addLocation(...

SqlConnection cnTB = new SqlConnection(...

string query = "INSERT INTO locations(...

are single commands and should be entered without line breaks.

```
public int XbottomRight { get; set; }
public int YbottomRight { get; set; }
```

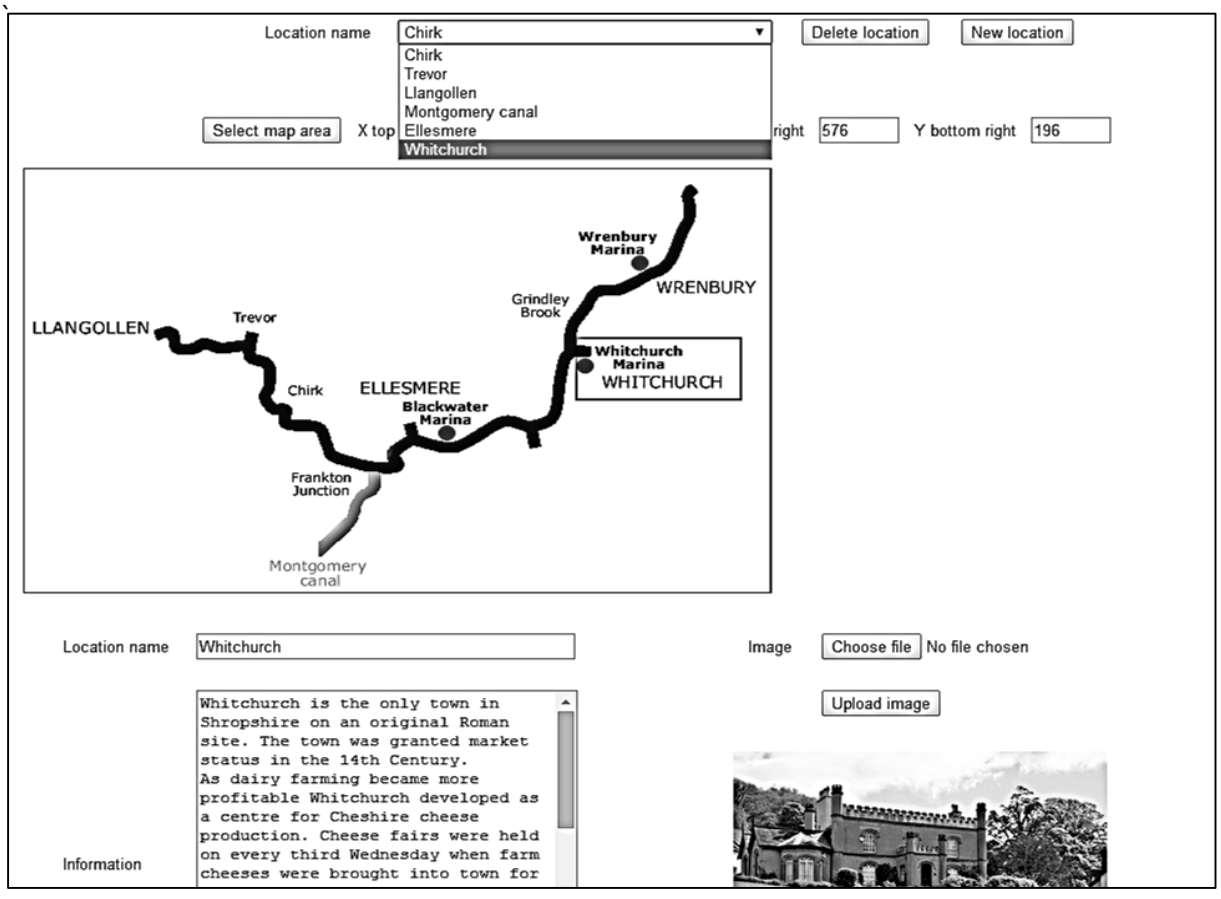
```
public static string addLocation(byte[] picbyte, string locationName,
      string locationText, int XtopLeft, int YtopLeft, int XbottomRight,
      int YbottomRight, int locationSelected)
{
    string message = "";
    SqlParameter picparameter = new SqlParameter();
    picparameter.SqlDbType = SqlDbType.Image;
    picparameter.ParameterName = "pic";
    picparameter.Value = picbyte;
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmStock = new SqlCommand();
        cmStock.Connection = cnTB;
        cmStock.CommandType = CommandType.Text;
        string query = "INSERT INTO locations(locationName,locationText,XtopLeft,
            YtopLeft, XbottomRight, YbottomRight, locationImage) VALUES ('" +
            locationName + "','" + locationText + "','" + XtopLeft + "','" +
            YtopLeft + "','" + XbottomRight + "','" + YbottomRight + "', @pic )";
        SqlCommand cmd = new SqlCommand(query, cnTB);
        cmd.Parameters.Add(picparameter);
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record saved";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

Return to the ***staffRoute.aspx*** C# code page. Make changes to ***btnSave_Click( )*** method so that the correct file handling method is selected in the ***location*** class file, depending on whether the current record is being added or updated.

```
protected void btnSave_Click(object sender, EventArgs e)
{
    string information = txtInformation.Text;
    information = information.Replace("'", "`");
    if (newRecord == false)
    {
        lblMessage.Text = location.updateLocation(picbyte,
             txtLocationName.Text, information, topLeftX, topLeftY,
             bottomRightX, bottomRightY, locationSelected);
    }
    else
    {
        lblMessage.Text = location.addLocation(picbyte, txtLocationName.Text,
            information, topLeftX, topLeftY, bottomRightX, bottomRightY,
            locationSelected);
        newRecord = false;
    }
    location.loadLocations();
    loadData();
}
```

Build and run the staffRoute web page. Use the '***New location***' button to enter details of other canal locations. Each time, select an active rectangle on the image map and enter a photograph. After saving the record, check that the new location now appears on the drop down list.

`

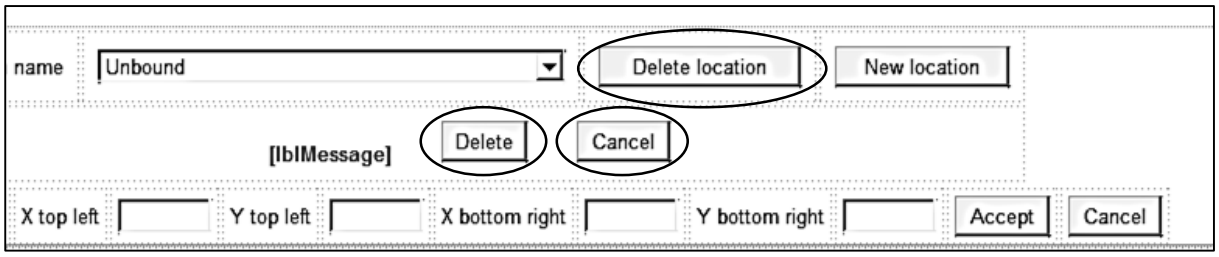


Close the web browser and stop debugging. The final function to be added to the ***staffRoute*** page is the '***Delete location***' option.

Open the ***staffRoute.aspx*** page and add lines of code to create two additional buttons.

```
<div id="locationContent">
  <div id="locationList">
     <table border=0 cellpadding=10>
             . . . . . .
     <td colspan=4>
     <center>
     <asp:Label ID="lblMessage" runat="server" Font-Bold="True" ForeColor="red">
     </asp:Label>

            
     <asp:Button ID="btnConfirmDelete" runat="server" Text="Delete" Visible="False"/>
            
     <asp:Button ID="btnConfirmCancel" runat="server" Text="Cancel" Visible="False"/>

    </center>
    </td>
    </tr>
   </table>
```

Go to the Design view and create three new button_click methods by double clicking the '***Delete location***', '***Delete***' and '***Cancel***' buttons.



Open the ***location.cs*** class file and add a ***deleteLocation( )*** method.

```
   public int XbottomRight { get; set; }
   public int YbottomRight { get; set; }

   public static string deleteLocation(int locationSelected)
   {
     string message = "";
     SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
            AttachDbFilename=" + databaseLocation + "Integrated Security=True;
            Connect Timeout=30; User Instance=True");
     try
     {
        cnTB.Open();
       SqlCommand cmStock = new SqlCommand();
       cmStock.Connection = cnTB;
       cmStock.CommandType = CommandType.Text;
```

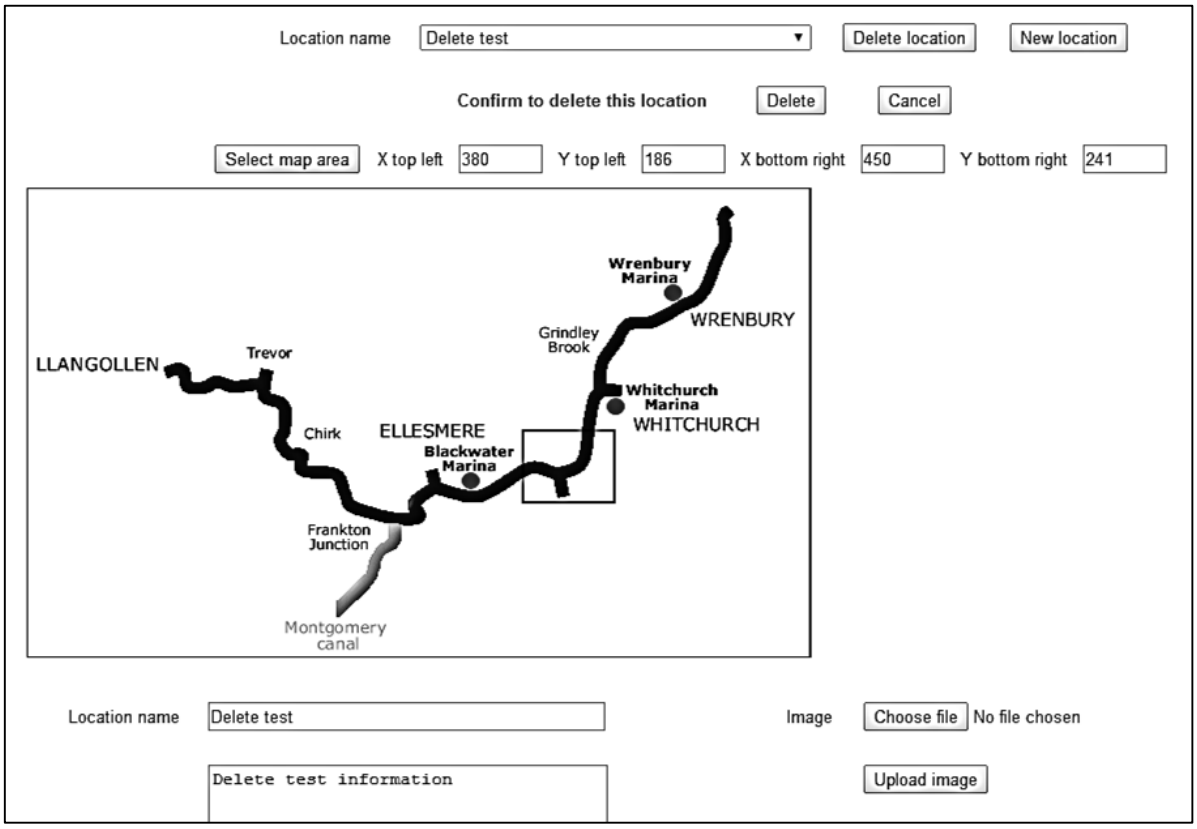
```
        string query;
        query = "DELETE FROM locations WHERE locationID='" + locationSelected + "'";
        message = query;
        SqlCommand cmd = new SqlCommand(query, cnTB);
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record deleted";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

Return to the ***staffRoute.aspx*** page and add code to the three empty button_click methods created earlier.

```
protected void btnDeleteLocation_Click(object sender, EventArgs e)
{
    lblMessage.Text = "Confirm to delete this location";
    btnConfirmDelete.Visible = true;
    btnConfirmCancel.Visible = true;
}

protected void btnConfirmDelete_Click(object sender, EventArgs e)
{
    string message = location.deleteLocation(locationSelected);
    btnConfirmDelete.Visible = false;
    btnConfirmCancel.Visible = false;
    lblMessage.Text = message;
    location.loadLocations();
    loadData();
    displayRecord(0);
}

protected void btnConfirmCancel_Click(object sender, EventArgs e)
{
    lblMessage.Text = "";
    btnConfirmDelete.Visible = false;
    btnConfirmCancel.Visible = false;
}
```

Build and run the ***staffRoute*** web page. Enter another test record.

Select the test record from the drop down list, then click the '***Delete record***' button. The confirm message should appear, along with the '***Delete***' and '***Cancel***' buttons.



Check first that the '***Cancel***' button leaves the test record unaffected.

Select the '***Delete location***' option again, but this time click the '***Delete***' button. The test record should now have been deleted from the database table and removed from the drop down list.

We have completed the staff page for entry, editing and deleting canal locations. We can now work on the public display page.

Open the C# code for ***route.aspx***. Add lines of code to the ***Page_Load( )*** method, as shown on the next page. Note that the line beginning:

Label2.Text = "Click on the names ...

should be entered as a single command with no line break.

The program collects the name of the location included in the page URL, for example:

route.aspx?location=Llangollen

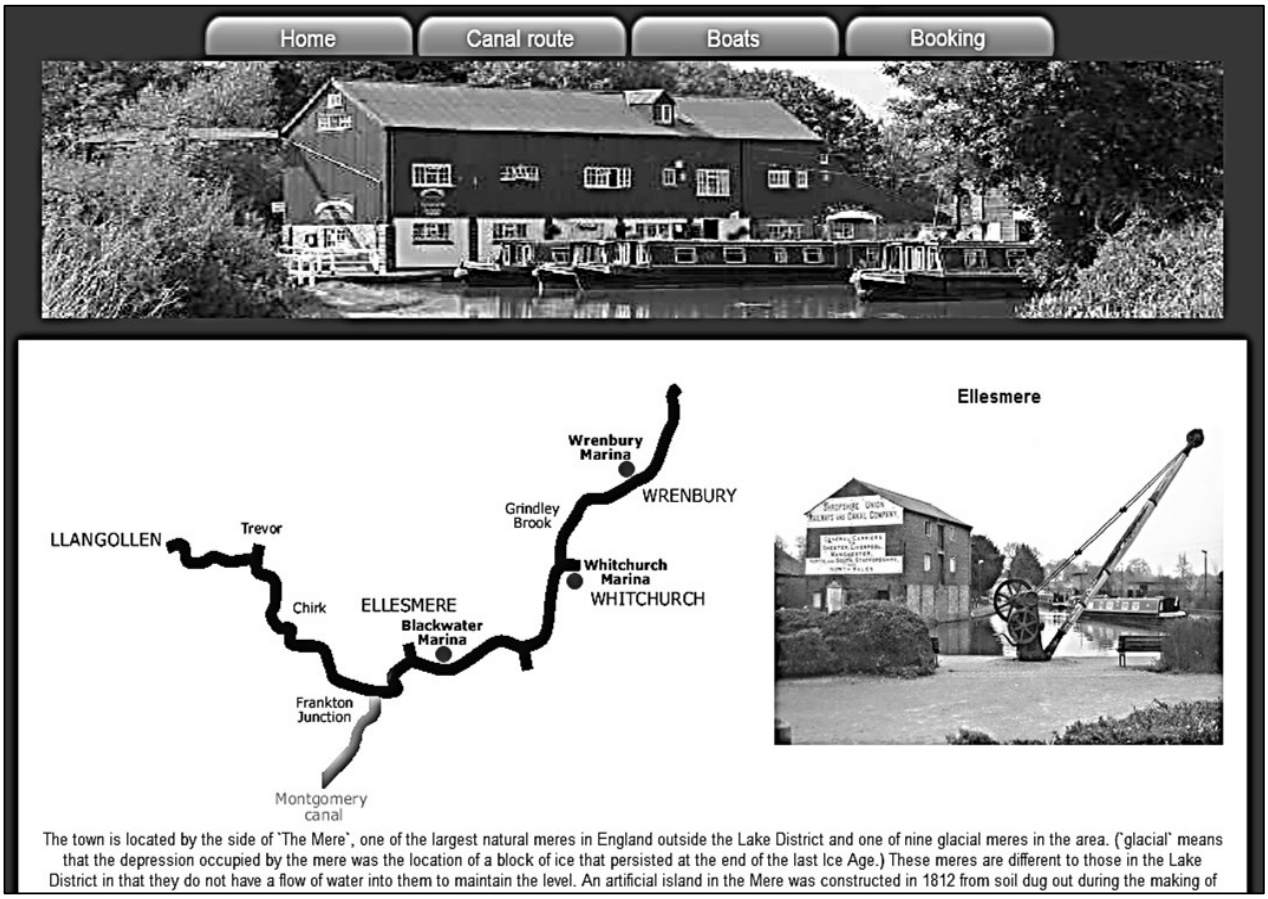
by means of the command

locationWanted = Request.QueryString["location"];

The ***locationWanted*** variable is then used to select the correct information and photograph to display.

```
protected void Page_Load(object sender, EventArgs e)
{
    string locationWanted = Request.QueryString["location"];
    string s2 = "";
    string s3 = "";
    int photoID;

    if (location.locationCount < 1)
    {
        location.loadLocations();
    }

    string s ="";
    s +="<img src='Images/canal map.png' usemap=#canalmap border=0 />";
    s += "<map name=canalmap>";
    for (int i = 0; i < location.locationCount; i++)
    {
        s += "<area shape=rect coords=";
        s += location.locationObject[i].XtopLeft + "," +
                  location.locationObject[i].YtopLeft + "," +
                  location.locationObject[i].XbottomRight + "," +
                  location.locationObject[i].YbottomRight;
        s += " href='route.aspx?location=" +
                  location.locationObject[i].locationName + "' />";

        if (locationWanted == location.locationObject[i].locationName)
        {
            s2 += "<h3>" + location.locationObject[i].locationName + "</h3>";
            s3 += location.locationObject[i].locationText;
            photoID = location.locationObject[i].locationID;
            s2 += "<img src='Handler2.ashx?imgid=" + photoID +
            "' width='380' border='0' >";
        }

    }
    s += "</map>";
    Label1.Text = s;

    Label2.Text = s2;
    Label3.Text = s3;
    if (Label2.Text == "")
    {
        Label2.Text = "Click on the names of locations on the map to display
                       information";
    }
}
```

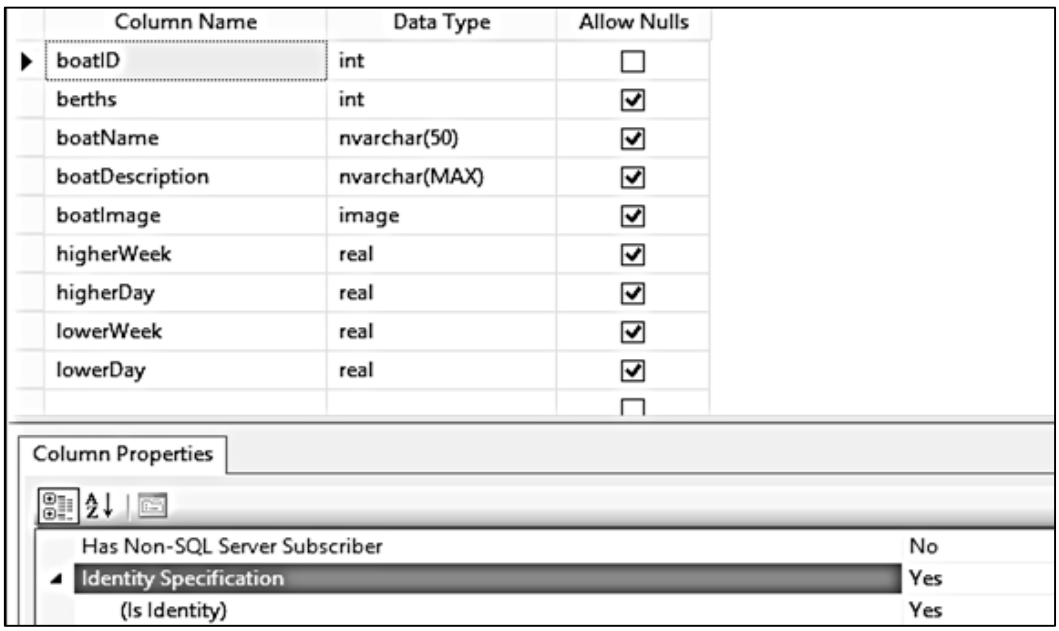
Build and run the web site. Go to the '***Canal Route***' page and check that information and photographs are displayed correctly when locations are selected from the map.



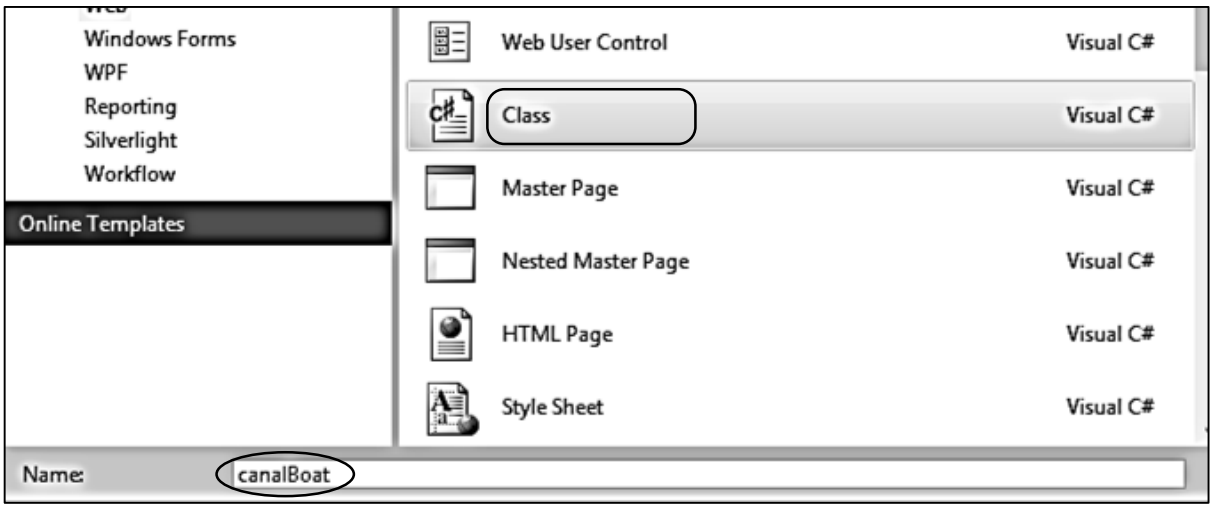
This completes the '***Canal Route***' page. We will now move on to the '***Boats***' option.

We will begin by setting up a database table to store boat details. Go to the Server Explorer window and open the ***canalHolidays*** database. Right click the ***Tables*** icon and select '***Add New Table***'.

Insert the fields shown below. The '***homePageID***' field should be set as an auto-number by opening the '***Identity Specification***' property and changing the value of '***(Is identity)***' to '***Yes***'. Save the completed table with the name '***boats***'.



We will now create a class file. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name '**canalBoat**'.



Open the ***canalBoat.cs*** file and add lines of code to set up the properties for the boat objects.

```
public class canalBoat
{
    public static int boatCount = 0;
    public static canalBoat[] boatObject = new canalBoat[100];
    public static string databaseLocation =
                          "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";

    public int boatID { get; set; }
    public int berths { get; set; }
    public string boatName { get; set; }
    public string boatDescription { get; set; }
    public double higherWeek { get; set; }
    public double higherDay { get; set; }
    public double lowerWeek { get; set; }
    public double lowerDay { get; set; }
    public bool pictureLoaded { get; set; }
}
```

Move to the ***staffBoats.aspx*** page and add code to the '***content2***' section to produce a series of menu options.

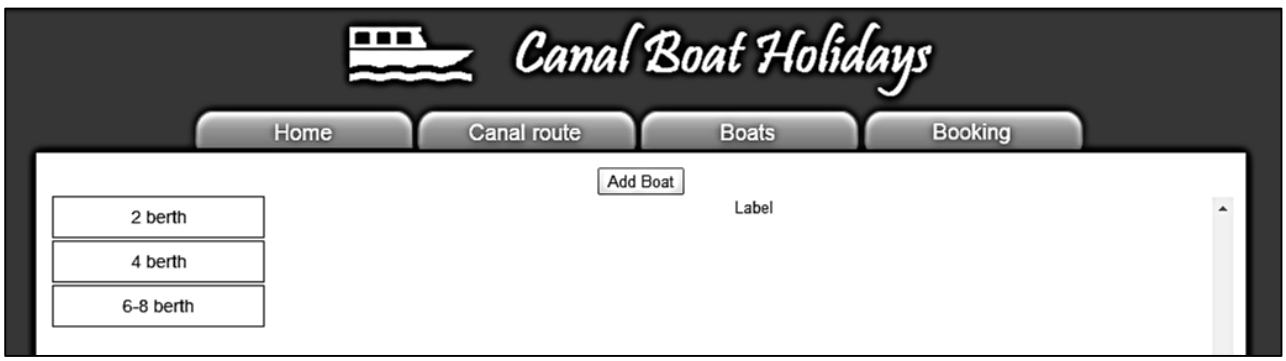
```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="boatContent">
    <center>
        <asp:Button ID="btnAddBoat" runat="server" Text="Add Boat" />
    </center>
    <div id="boats">
          <ul id="cat">
              <li><a href="staffBoats.aspx?berthsWanted=2">2 berth</a></li>
              <li><a href="staffBoats.aspx?berthsWanted=4">4 berth</a></li>
              <li><a href="staffBoats.aspx?berthsWanted=6">6-8 berth</a></li>
          </ul>
    </div>
    <div id="boatDisplay">
       <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </div>
    <br />
    <br />
  </div>
</asp:Content>
```

Open the ***Style Sheet*** and add formatting blocks for the divisions and menu.

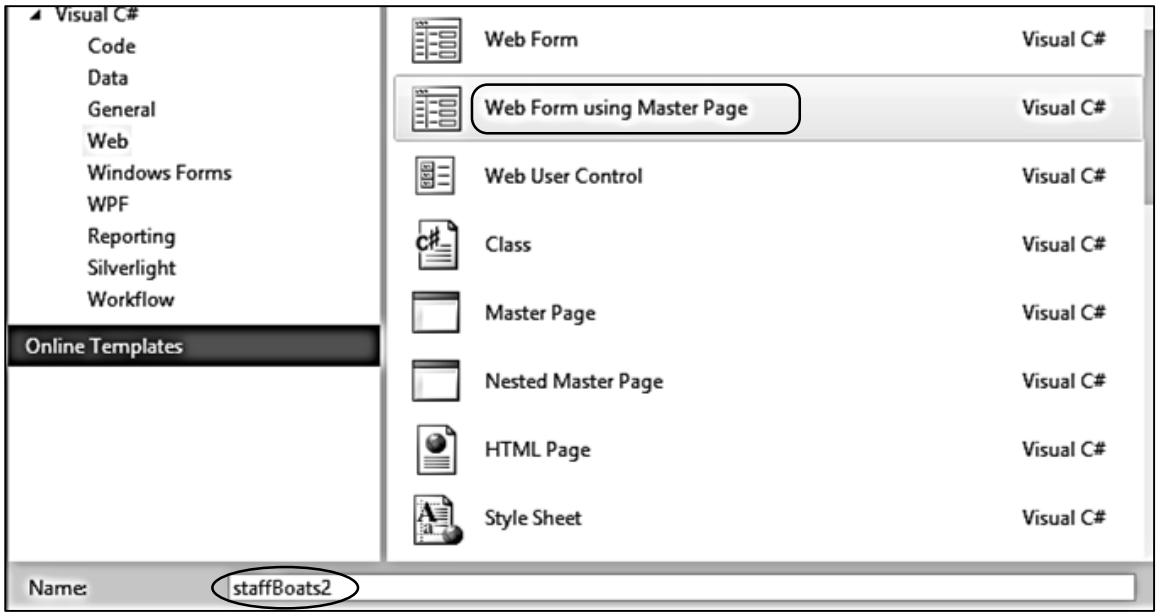
```
#boatContent
 {
     width: 960px;
     background-color: #FFFFFF;
     padding: 10px;
 }
#boatDisplay
{
    height:560px;
    overflow-y: scroll;
    overflow-x: hidden;
}
#boats
{
    background-color: white;
    float: left;
    width: 200px;
    list-style: none;
}
#cat
{
    list-style: none;
}
```

```
#cat ul
{
    list-style: none;
    display: none;
}
#cat li
{
    background-color: #FFFFFF;
    margin: 1px;
    font-size: 14px;
    float: left;
    position: relative;
    width: 170px;
    height: 32px;
    left: -38px;
    top: -15px;
    border: 1px solid #000000;
}
#cat a:link, #nav a:active, #nav a:visited
{
    display:block;
    color: #000000;
    text-decoration: none;
    padding: 8px;
    text-align: center;
}
#cat a:hover
{
    font-weight:bold;
}
```

Build and run the ***staffBoats*** web page. Check that menu options are displayed. There should be a scroll window to display boat details, and a button option to add another boat.



Close the browser and stop debugging. We will set up another page where details of boats can be entered. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Web*** / ***Web Form using Master Page***. Give the name '**staffBoats2**'. Select ***staffSite.Master*** as the master file.



Return to the ***staffBoats.aspx*** file and change to the ***Design*** view. Double click the '***Add Boat***' button to produce an on_click( ) method.

Add a line of code to to the ***staffBoats.aspx*** file which will load the ***staffBoats2.aspx*** page.

```
protected void btnAddBoat_Click(object sender, EventArgs e)
{
    Response.Redirect("staffBoats2.aspx?action=new");
}
```

Go now to the ***staffBoats2.aspx*** file and add code to the '***content2***' section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

  <div id="boatEdit">
     <asp:Label ID="lblMessage" runat="server" Font-Bold="True" ForeColor="red">
     </asp:Label>
     <div id="leftColumn">
        <table border="0" cellpadding="10" bgcolor="white">
            <tr>
              <td>Boat name</td>
```

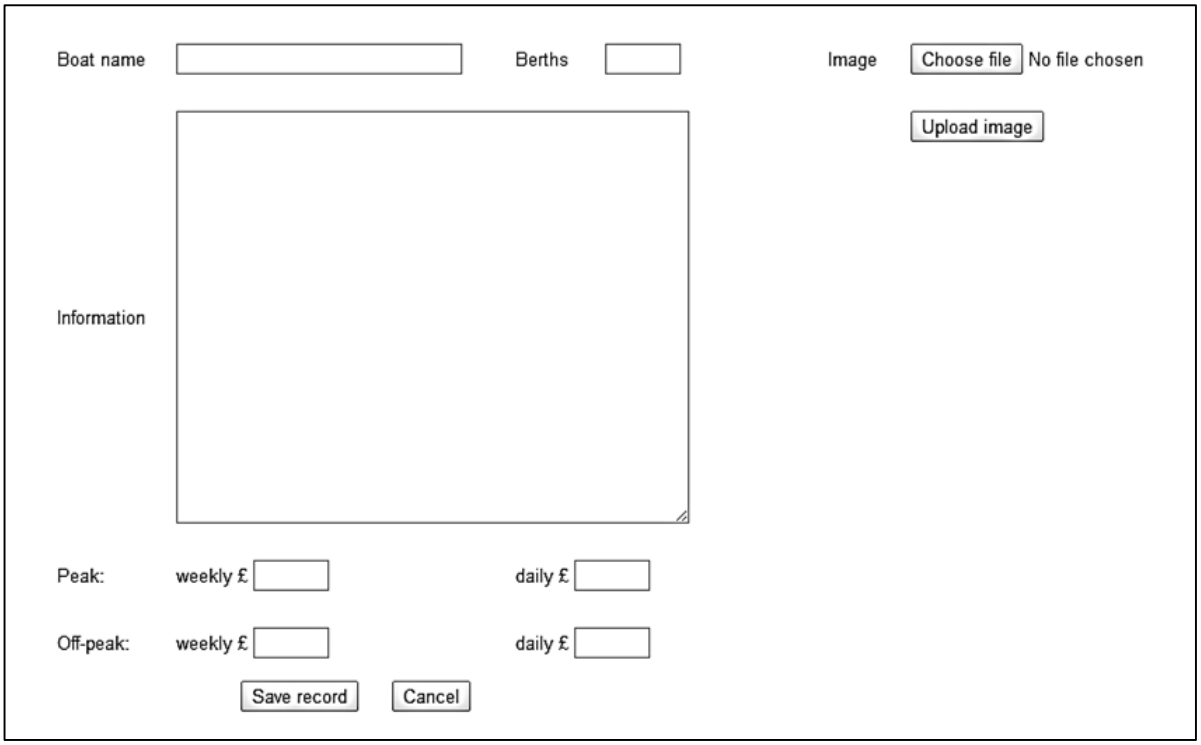
```
      <td>
         <asp:TextBox ID="txtBoatName" runat="server" Width="200px">
         </asp:TextBox>
      </td>
      <td>Berths</td>
      <td>
         <asp:TextBox ID="txtBerths" runat="server" Width="50px">
         </asp:TextBox>
      </td>
  </tr>
  <tr>
      <td>Information</td>
      <td colspan="3">
         <asp:TextBox ID="txtInformation" runat="server" Width="360px"
                              TextMode="MultiLine" Rows="18"></asp:TextBox>
      </td>
  </tr>
  <tr>
     <td >Peak:</td>
     <td>weekly £
         <asp:TextBox ID="txtHigherWeek" runat="server" Width="50px">
         </asp:TextBox>
     </td>
     <td colspan="2">daily £
         <asp:TextBox ID="txtHigherDay" runat="server" Width="50px">
         </asp:TextBox>
     </td>
  </tr>
  <tr>
     <td>Off-peak:</td>
     <td>weekly £
         <asp:TextBox ID="txtLowerWeek" runat="server" Width="50px">
         </asp:TextBox>
     </td>
     <td colspan="2">daily £
         <asp:TextBox ID="txtLowerDay" runat="server" Width="50px">
         </asp:TextBox>
     </td>
   </tr>
  </table>
  <center>
    <asp:Button ID="btnSave" runat="server" Text="Save record" />
       
    <asp:Button ID="btnCancelRecord" runat="server" Text="Cancel" />
  </center>
</div>
<div id="rightColumn" style="float: right">
  <table border="0" cellpadding="10">
   <tr>
     <td>Image</td>
     <td>
        <asp:FileUpload ID="FileUpload1" runat="server" />
     </td>
```

```
      </tr>
      <tr>
         <td></td>
         <td>
            <asp:Button ID="btnUpload" runat="server" Text="Upload image" />
         </td>
      </tr>
    </table>
    <br />
    <asp:Image ID="Image1" runat="server"  width='300px' />
    <br /><br />
  </div>
</div>
</asp:Content>
```

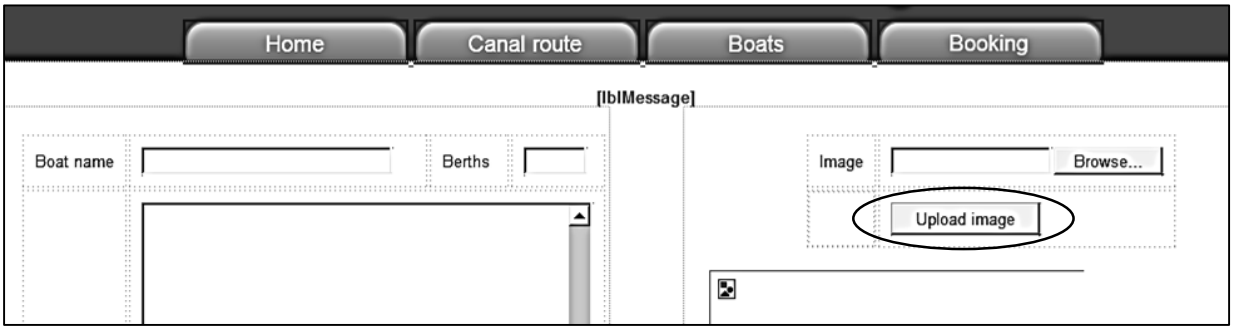
Open the ***Style Sheet*** and add a formatting block for the ***boatEdit*** division.

```
#boatEdit
 {
     width: 1040px;
     height: 600px;
     background-color: #FFFFFF;
     padding:20px;
 }
```

Build and run the ***staffBoats2*** page. Check that data entry boxes and buttons are displayed correctly.



Close the browser and stop debugging. Go to the ***staffBoats2.aspx*** page and select the Design view. Double click the '***Upload image***' button to create a button_click method.



Add lines of code to the ***btnUlpoad_Click( )*** method, and also a ***picbyte*** variable for storing the digitised image when it is uploaded.

```
public partial class staffBoats2 : System.Web.UI.Page
{
    public static byte[] picbyte;

    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void btnUpload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.HasFile)
            {
                FileUpload1.SaveAs(Server.MapPath("Images").ToString() + @"\" +
                    FileUpload1.FileName);
                Image1.ImageUrl = @"Images\" + FileUpload1.FileName;
                picbyte = FileUpload1.FileBytes;
            }
            else
            {
                lblMessage.Text = "Please load an image";
            }

        }
        catch
        {

        }
    }
}
```

Build and run the ***staffBoats2*** web page. Check that a photograph can be uploaded.

Close the browser and stop debugging.

Open the ***canalBoat.cs*** file. Add '***using System.Data***' and '***using System.Data.SqlClient***' directives.

```
using System.Linq;
using System.Web;

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

Add a method to save boat records into the database. Note that the lines beginning:

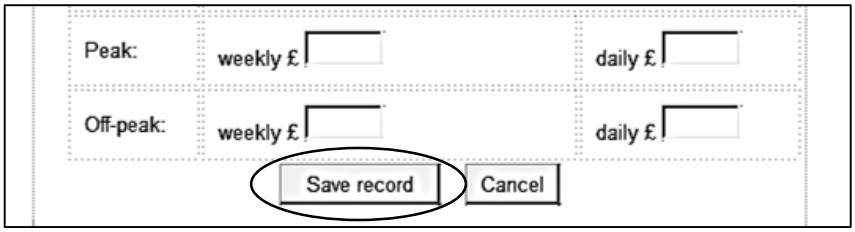
public static string addBoat(...
SqlConnection cnTB = new SqlConnection(...
query = "INSERT INTO boats(...

are single commands without line breaks.

```
public double lowerDay { get; set; }
public bool pictureLoaded { get; set; }

public static string addBoat(byte[] picbyte, int berths, string boatName,
     string boatDescription, double higherWeek, double higherDay,
     double lowerWeek, double lowerDay)
{
    string message = "";
    string query = "";

    SqlParameter picparameter = new SqlParameter();
    picparameter.SqlDbType = SqlDbType.Image;
    picparameter.ParameterName = "pic";
    picparameter.Value = picbyte;
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation +"Integrated Security=True;
        Connect Timeout=30; User Instance=True");
```

```
        try
        {
            cnTB.Open();
            SqlCommand cmBoat = new SqlCommand();
            cmBoat.Connection = cnTB;
            cmBoat.CommandType = CommandType.Text;
            query = "INSERT INTO boats(berths,boatName,boatDescription,higherWeek,
                  higherDay, lowerWeek, lowerDay, boatImage) VALUES ('" + berths +
                  "','" + boatName + "','" + boatDescription + "','" + higherWeek +
                  "','" + higherDay + "','" + lowerWeek + "','" + lowerDay +
                  "', @pic )";
            SqlCommand cmd = new SqlCommand(query, cnTB);
            cmd.Parameters.Add(picparameter);
            cmd.ExecuteNonQuery();
            cnTB.Close();
            message = "Record saved";
        }
        catch
        {
            message = "File error";
        }
        return message;
    }
}
```

Return to the ***staffBoats2.aspx*** page and change to the Design view. Double click the '***Save record***' button to create a button_click( ) method.



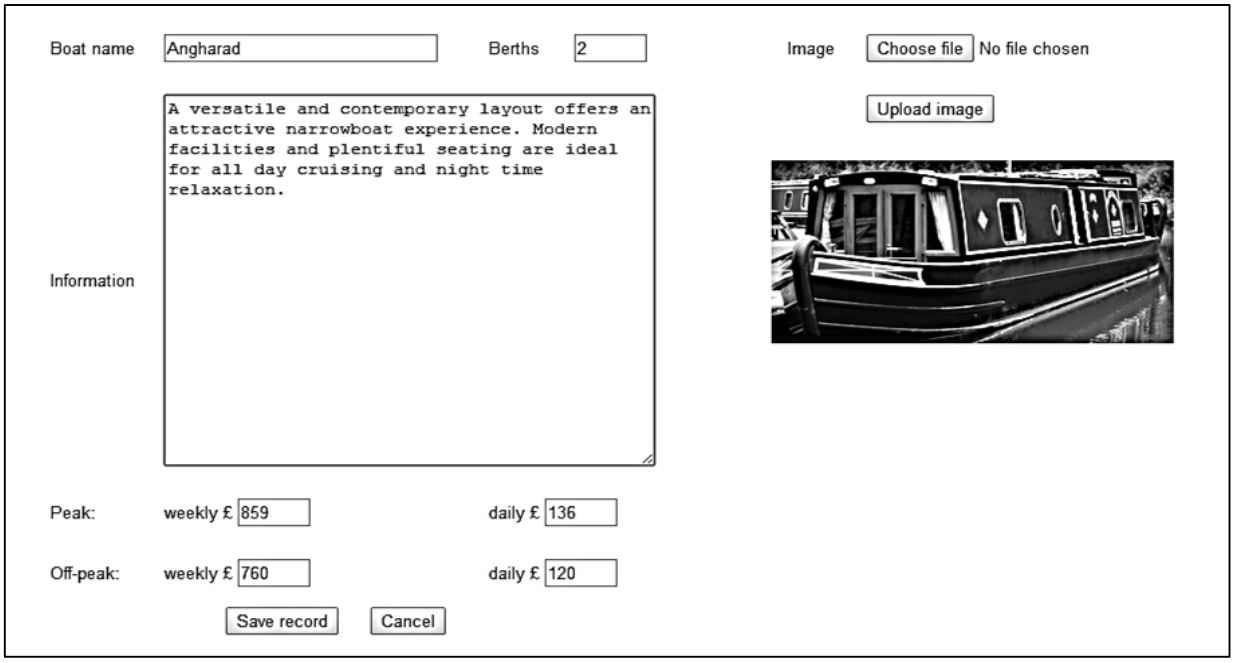
Add lines of code to replace any apostrophies which might cause a file error, then call the method to save the boat record into the database. Note that the line beginning:

lblMessage.Text = canalBoat.addBoat(...

is a single command and should be entered without line breaks.

```
protected void btnSave_Click(object sender, EventArgs e)
{
    string information = txtInformation.Text;
    information = information.Replace("'", "`");
    lblMessage.Text = canalBoat.addBoat(picbyte,
        Convert.ToInt16(txtBerths.Text), txtBoatName.Text, information,
        Convert.ToDouble(txtHigherWeek.Text), Convert.ToDouble(txtHigherDay.Text),
        Convert.ToDouble(txtLowerWeek.Text), Convert.ToDouble(txtLowerDay.Text));
}
```

Build and run the ***staffBoats2*** web page. Enter details and a photograph for a canal boat, then click the '***Save record***' button. The message '***Record saved***' should appear.



Close the browser and stop debugging. Go to the Server explorer window and refresh the database. Open the ***boats*** table and check that the new record is present.

boatID	berths	boatName	boatDescription	boatImage	higherWeek	higherDay	lowerWeek	lowerDay
1	2	Angharad	A versatile and contemporary layout...	<Binary data>	859	136	760	120

Re-run the ***staffBoats2*** web page and enter more boats, choosing accommodation sizes of 2, 4, 6 or 8 berths. You will need to clear the previous text box entries before adding each new boat.

Close the browser and stop debugging. Return to the Server explorer window, refresh the database, and check that the records have been inserted correctly.

boatID	berths	boatName	boatDescription	boatImage	higherWeek	higherDay	lowerWeek	lowerDay
1	2	Angharad	A versatile and contemporary layout...	<Binary data>	859	136	760	120
2	2	Sioned	A great choice for just the two of yo...	<Binary data>	920	140	810	128
3	2	Gwen	Single beds 6'3" x2', interior sprung ...	<Binary data>	960	143	810	136
4	4	Bethan	Fitted out to a very high standard. T...	<Binary data>	1280	196	1120	184
5	6	Catrin	Superbly crafted light airy interior in...	<Binary data>	1456	228	1325	216

Our next step is to display the details of the boats on the main ***staffBoats*** page. Go to the C# code for ***staffBoats.aspx*** and add lines of code to the ***Page_Load*** method.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (canalBoat.boatCount == 0)
    {
        canalBoat.loadBoats();
    }
}
```

Move now to the ***canalBoat.cs*** class file and add a ***loadBoats( )*** method.

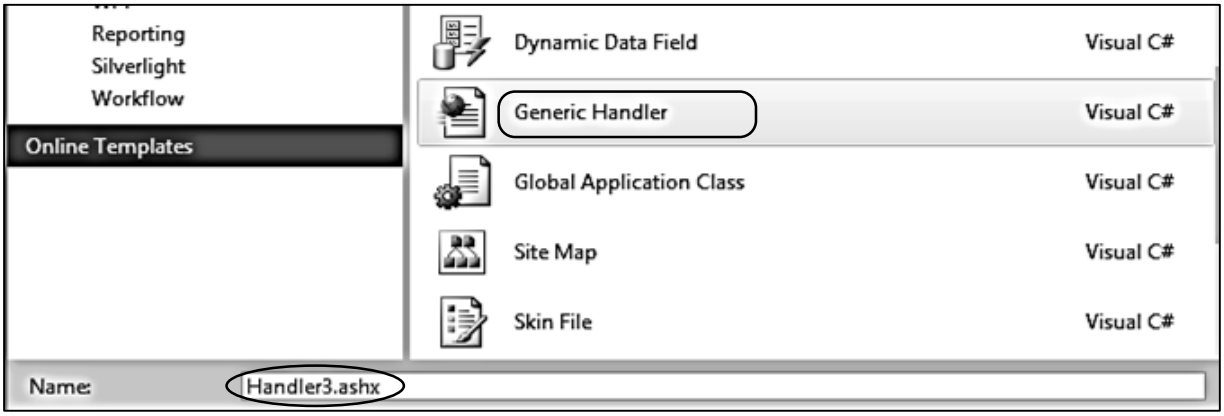
```
public static void loadBoats()
{
    DataSet dsBoats = new DataSet();
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
       AttachDbFilename=" + databaseLocation + "Integrated Security=True;
       Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmBoat = new SqlCommand();
        cmBoat.Connection = cnTB;
        cmBoat.CommandType = CommandType.Text;
        cmBoat.CommandText = "SELECT * FROM boats";
        SqlDataAdapter daBoat = new SqlDataAdapter(cmBoat);
        daBoat.Fill(dsBoats);
        cnTB.Close();
        int countRecords = dsBoats.Tables[0].Rows.Count;
        boatCount = 0;
        for (int i = 0; i < countRecords; i++)
        {
            DataRow drBoat = dsBoats.Tables[0].Rows[i];
            int boatID = (int)drBoat[0];
            int berths = (int)drBoat[1];
            string boatName = Convert.ToString(drBoat[2]);
            string boatDescription = Convert.ToString(drBoat[3]);
            double higherWeek = Convert.ToDouble(drBoat[5]);
            double higherDay = Convert.ToDouble(drBoat[6]);
            double lowerWeek = Convert.ToDouble(drBoat[7]);
            double lowerDay = Convert.ToDouble(drBoat[8]);
            bool pictureLoaded = true;
            if (drBoat.IsNull("boatImage"))
            {
                pictureLoaded = false;
            }
            boatObject[boatCount] = new canalBoat();
            boatObject[boatCount].boatID = boatID;
            boatObject[boatCount].berths = berths;
            boatObject[boatCount].boatName = boatName;
            boatObject[boatCount].boatDescription = boatDescription;
            boatObject[boatCount].higherWeek = higherWeek;
            boatObject[boatCount].higherDay = higherDay;
            boatObject[boatCount].lowerWeek = lowerWeek;
            boatObject[boatCount].lowerDay = lowerDay;
            boatObject[boatCount].pictureLoaded = pictureLoaded;
            boatCount++;
        }
    }
    catch
    {
    }
}
```

Return to the ***staffBoats.aspx*** C# code page and add further lines to the ***Page_Load( )*** method.

```
protected void Page_Load(object sender, EventArgs e)
{
   if (canalBoat.boatCount == 0)
   {
       canalBoat.loadBoats();
   }
   int berthsWanted = Convert.ToInt16(Request.QueryString["berthsWanted"]);
   string s = "";
   s += "<table cellpadding=8 cellspacing =8 bgcolor=white>";
   int photoID;
   for (int i = 0; i < canalBoat.boatCount; i++)
   {
      int berthsProvided = canalBoat.boatObject[i].berths;
      if (berthsProvided > 6)
      {
          berthsProvided = 6;
      }
      if (berthsWanted == berthsProvided)
      {
          s += "<tr>";
          s += "<td >";
          s += "<h3>" + canalBoat.boatObject[i].boatName + "</h3>";
          s += canalBoat.boatObject[i].berths + " berths <br><br>";
          s += canalBoat.boatObject[i].boatDescription + "<br>";
          double priceWeek = canalBoat.boatObject[i].higherWeek;
          string priceStringWeek = String.Format("{0:.##}", priceWeek);
          double priceDay = canalBoat.boatObject[i].higherDay;
          string priceStringDay = String.Format("{0:.##}", priceDay);
          s += "<h3>Peak:     weekly £" + priceStringWeek +
                 "     daily £" + priceStringDay + "</h3>";
          priceWeek = canalBoat.boatObject[i].lowerWeek;
          priceStringWeek = String.Format("{0:.##}", priceWeek);
          priceDay = canalBoat.boatObject[i].lowerDay;
          priceStringDay = String.Format("{0:.##}", priceDay);
          s += "<h3>Off-peak:     weekly £" + priceStringWeek +
                "     daily £" + priceStringDay + "</h3>";
          s += "<td >";
          photoID = canalBoat.boatObject[i].boatID;
          if (canalBoat.boatObject[i].pictureLoaded == true)
          {
              s += "<img src='Handler3.ashx?imgid=" + photoID +
                                                  "' width='200' border='0' >";
          }
          s += "</td>";
          s += "<td>";
          s += "<a href='staffBoats2.aspx?action=edit&boatID=";
          s += canalBoat.boatObject[i].boatID;
          s += "'> edit </a><br><br>";
          s += "<a href='staffBoats2.aspx?action=delete&boatID=";
```

```
            s += canalBoat.boatObject[i].boatID;
            s += "'> delete </a>";
            s += "</td>";
            s += "</tr>";
            s += "<tr>";
            s += "<td colspan=3><hr></td>";
            s += "</tr>";
        }
    }
    s += "</table>";
    Label1.Text = s;
}
```

A handler will be needed to load and display the photograhs. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Generic Handler***. Accept the name '**Handler3**'.



Add lines of code to the handler file as shown below. Note that the lines beginning:

conn = new System.Data.SqlClient.SqlConnection(...

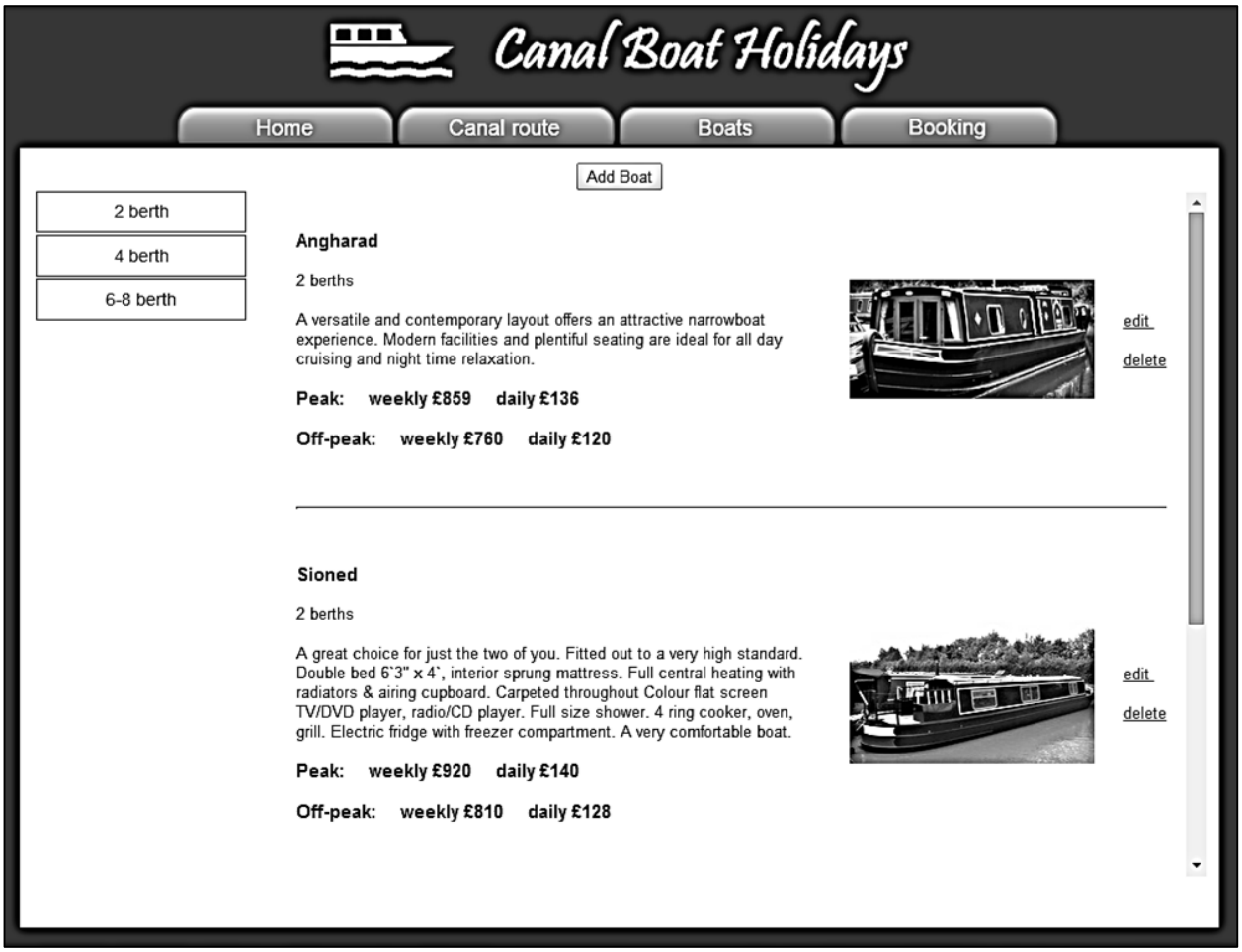
sqlcmd = new System.Data.SqlClient.SqlCommand(...

are single commands and should be entered without line breaks.

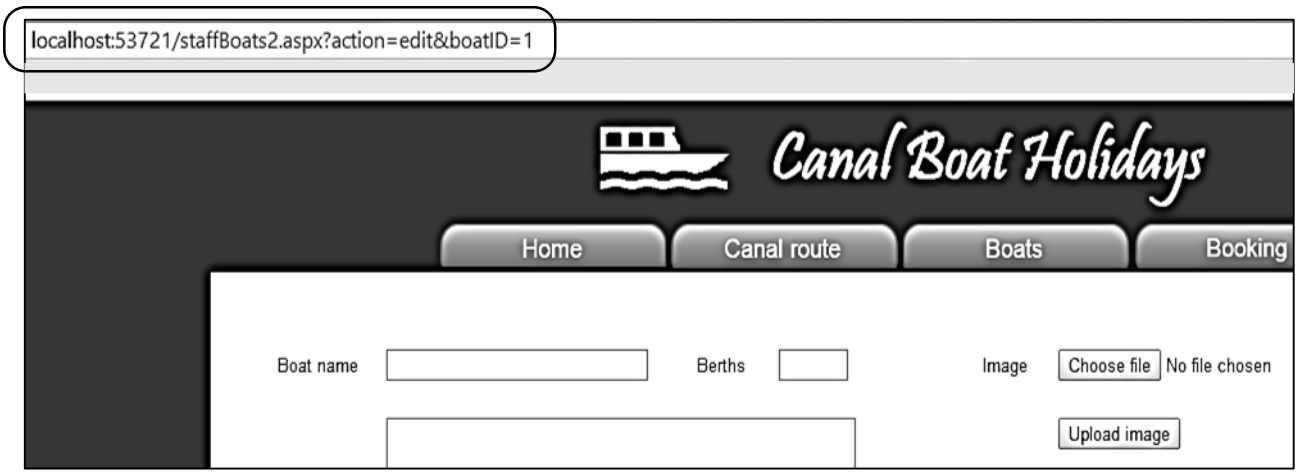
```
public class Handler3 : IHttpHandler
{
    string databaseLocation = "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";

    public void ProcessRequest(HttpContext context)
    {
        System.Data.SqlClient.SqlDataReader rdr = null;
        System.Data.SqlClient.SqlConnection conn = null;
        System.Data.SqlClient.SqlCommand sqlcmd = null;
        try
        {
            conn = new System.Data.SqlClient.SqlConnection(@"Data Source
                =.\SQLEXPRESS; AttachDbFilename=" + databaseLocation +
                "Integrated Security=True; Connect Timeout=30; User Instance=True");
```

```
        sqlcmd = new System.Data.SqlClient.SqlCommand("SELECT boatImage FROM boats
            WHERE boatID=" + context.Request.QueryString["imgid"], conn);
        conn.Open();
         rdr = sqlcmd.ExecuteReader();
         while (rdr.Read())
         {
            context.Response.ContentType = "image/jpg";
            context.Response.BinaryWrite((byte[])rdr["boatImage"]);
         }
         if (rdr != null)
            rdr.Close();
    }
    finally
    {
       if (conn != null)
          conn.Close();
    }
}
```

Build and run the staffBoats web page. Check that boats with different numbers of berths can be displayed.



With the web page still running, click on the **edit** option alongside one of the canal boats. The program will load the staffBoats2 page and takes with it the ***boatID*** number as part of the URL. The variable '***action***' is also set to '***edit***'.



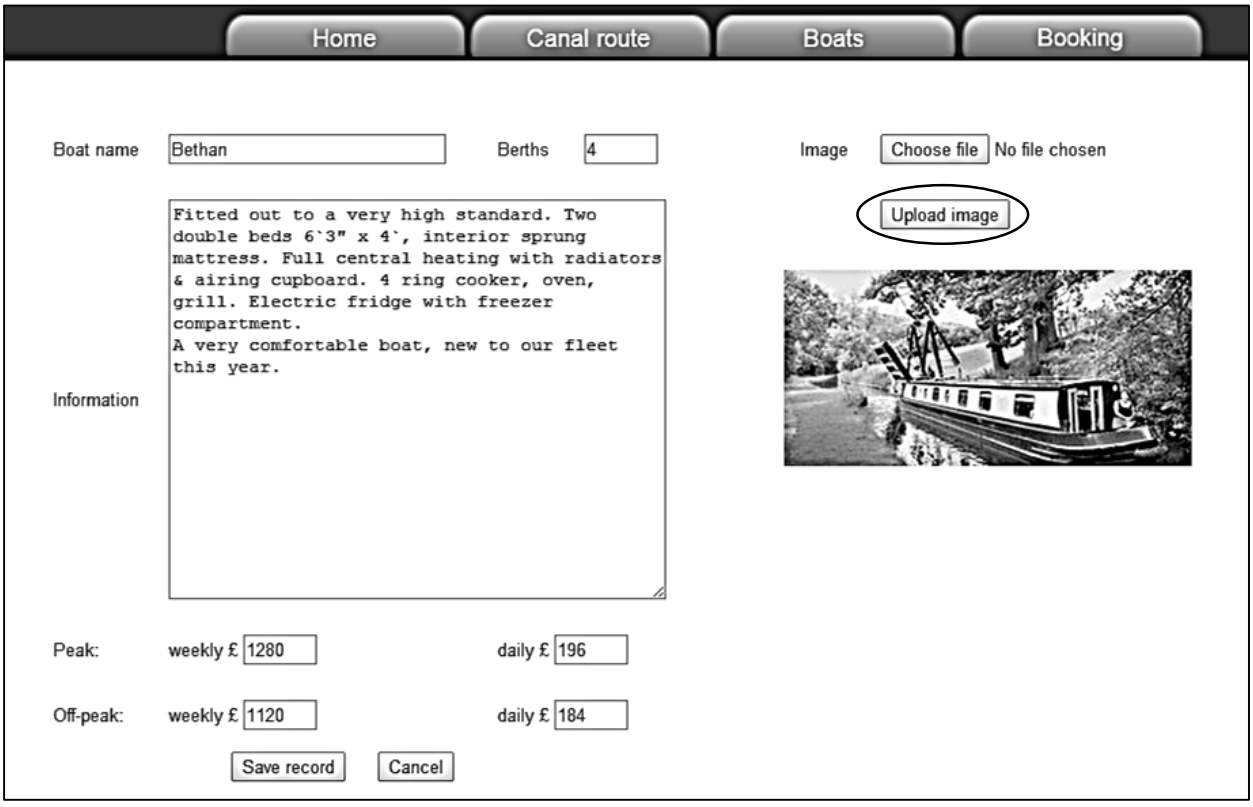
Close the browser and stop debugging. We will now work on the edit option.

Open the ***staffBoats2.aspx*** page. Add the variables '**action**' and '**boatID**' at the start of the class, and insert lines of code into the ***Page_Load( )*** method. This code will identify that a boat record is to be edited, and will select and display the current boat record and photograph. Note that all lines containing '***Convert.ToString...***' are single commands which should be entered without line breaks.

```
public partial class staffBoats2 : System.Web.UI.Page
{
   public static byte[] picbyte;
   string action;
   int boatIDwanted;

   protected void Page_Load(object sender, EventArgs e)
   {
      action = Request.QueryString["action"];
      boatIDwanted = Convert.ToInt16(Request.QueryString["boatID"]);
      if (txtBoatName.Text.Length == 0)
      {
         if (action == "edit")
         {
            for (int i = 0; i < canalBoat.boatCount; i++)
            {
               if (canalBoat.boatObject[i].boatID == boatIDwanted)
               {
                  txtBoatName.Text = canalBoat.boatObject[i].boatName;
                  txtBerths.Text = Convert.ToString(canalBoat.boatObject[i].berths);
                  txtInformation.Text = canalBoat.boatObject[i].boatDescription;
                  txtHigherWeek.Text =
                              Convert.ToString(canalBoat.boatObject[i].higherWeek);
                  txtHigherDay.Text =
                              Convert.ToString(canalBoat.boatObject[i].higherDay);
                  txtLowerWeek.Text =
                              Convert.ToString(canalBoat.boatObject[i].lowerWeek);
                  txtLowerDay.Text =
                              Convert.ToString(canalBoat.boatObject[i].lowerDay);
```

```
                if (canalBoat.boatObject[i].pictureLoaded == true)
                {
                Image1.ImageUrl = "Handler3.ashx?imgid=" +
                                        Convert.ToString(boatIDwanted);
            }
            else
            {
                Image1.ImageUrl = "";
            }
            picbyte = null;
        }
      }
    }
  }
}
```

Build and run the ***staffBoats*** web page. Select a canal boat, then click the **edit** option. Check that the ***staffBoats2*** page opens and displays the correct boat details and photograph. Test that an alternative photograph can be loaded.



Close the web browser and stop debugging.

Return to the C# page for ***staffBoats2.aspx*** and modify the ***btnSave_Click( )*** method to allow for both editing an existing boat record and adding a new boat.

```
protected void btnSave_Click(object sender, EventArgs e)
{
   string information = txtInformation.Text;
   information = information.Replace("'", "`");

   if (action == "edit")
   {
     lblMessage.Text = canalBoat.updateBoat(picbyte, Convert.ToInt16(txtBerths.Text),
        txtBoatName.Text, information, Convert.ToDouble(txtHigherWeek.Text),
        Convert.ToDouble(txtHigherDay.Text), Convert.ToDouble(txtLowerWeek.Text),
        Convert.ToDouble(txtLowerDay.Text), boatIDwanted);
   }
   if (action == "new")
   {
     lblMessage.Text = canalBoat.addBoat(picbyte, Convert.ToInt16(txtBerths.Text),
        txtBoatName.Text, information, Convert.ToDouble(txtHigherWeek.Text),
        Convert.ToDouble(txtHigherDay.Text), Convert.ToDouble(txtLowerWeek.Text),
        Convert.ToDouble(txtLowerDay.Text));
   }
   canalBoat.loadBoats();
}
```

Go to the ***canalBoat.cs*** class file and add an ***updateBoat( )*** method. This has two alternative SQL queries, depending on whether the photograph is being changed. Note that the lines beginning:

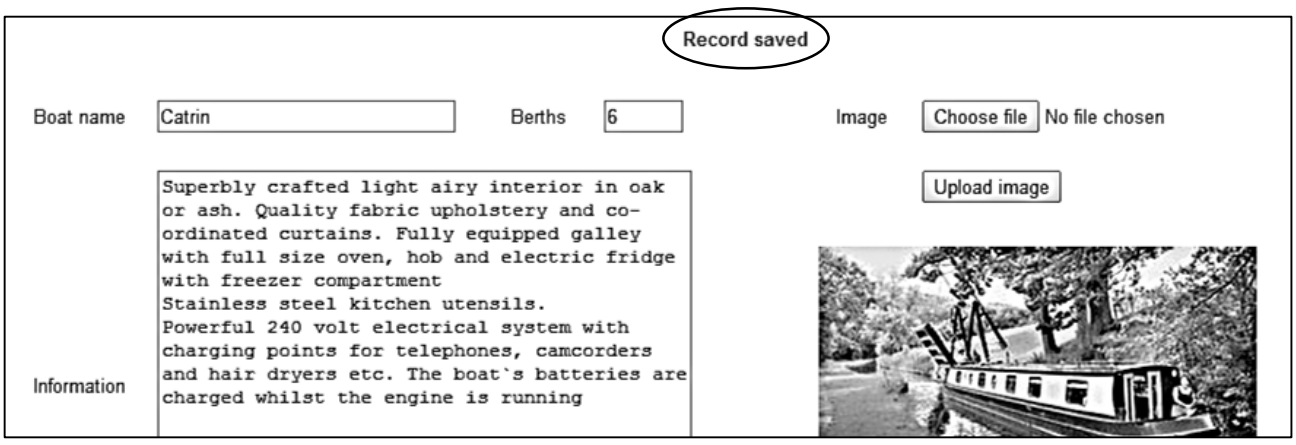
public static string updateBoat(...
SqlConnection cnTB = new SqlConnection(...
query = "UPDATE ...

are single commands and should be entered without line breaks.

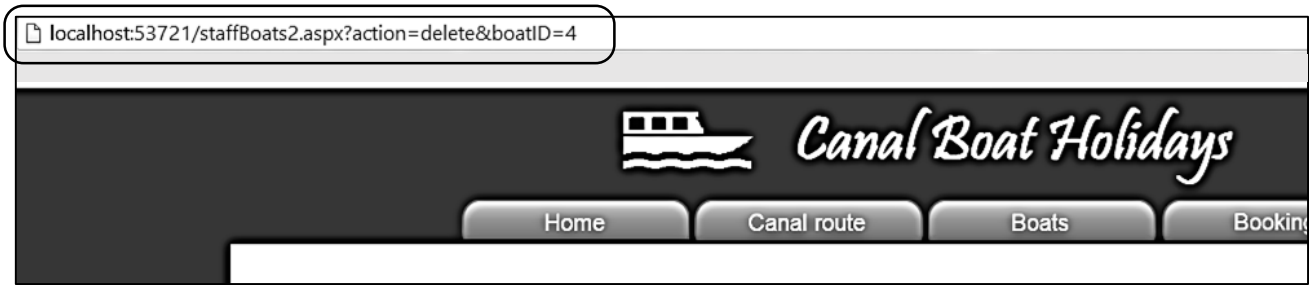
```
 public double lowerDay { get; set; }
 public bool pictureLoaded { get; set; }

 public static string updateBoat(byte[] picbyte, int berths, string boatName,
        string boatDescription, double higherWeek, double higherDay,
        double lowerWeek, double lowerDay, int boatSelected)
{
    string message = "";
    string query = "";
    SqlParameter picparameter = new SqlParameter();
    picparameter.SqlDbType = SqlDbType.Image;
    picparameter.ParameterName = "pic";
    picparameter.Value = picbyte;
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
         AttachDbFilename=" + databaseLocation + "Integrated Security=True;
         Connect Timeout=30; User Instance=True");
    try
    {
       cnTB.Open();
```

```
        SqlCommand cmBoat = new SqlCommand();
        cmBoat.Connection = cnTB;
        cmBoat.CommandType = CommandType.Text;

        if (picbyte == null)
        {
           query = "UPDATE boats SET berths='"+berths+"', boatName='" + boatName +
                "', boatDescription='" + boatDescription + "', higherWeek='" +
                higherWeek + "', higherDay='" + higherDay + "', lowerWeek='" +
                lowerWeek + "', lowerDay='" + lowerDay + "' WHERE boatID = '" +
                boatSelected + "'";
        }
        else
        {
          query = "UPDATE boats SET berths='" + berths + "', boatName='" + boatName +
                "', boatDescription='" + boatDescription + "', higherWeek='" +
                higherWeek + "', higherDay='" + higherDay + "', lowerWeek='" +
                lowerWeek + "', lowerDay='" + lowerDay + "', boatImage= @pic WHERE
                boatID = '" + boatSelected + "'";
        }
        SqlCommand cmd = new SqlCommand(query, cnTB);
        if (!(picbyte == null))
        {
            cmd.Parameters.Add(picparameter);
        }
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record saved";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

Build and run the staffBoats web page. Select a boat and click the **edit** option. Check that changes can be made to any of the data, including replacement of the photograph, and the updated record is saved correctly.



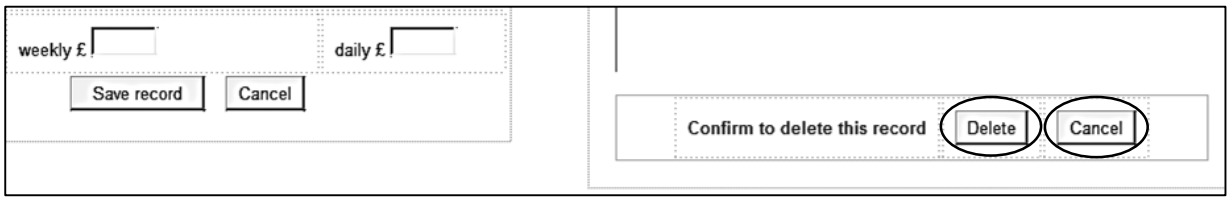
Select a canal boat and click the **delete** option. You will again be directed to the ***staffBoats2*** page, but notice that the variable '***action***' in the URL is set to '***delete***'. We now need to complete the program code to allow the deletion of records.



Close the browser and stop debugging. Open staffBoats2.aspx and add lines of code near the end of the file.

```
        <asp:Image ID="Image1" runat="server"  width='300px' />
        <br /><br />

    <asp:Panel ID="pnlConfirm" runat="server">
      <table cellpadding=10>
        <tr>
          <td><asp:Label ID="lblConfirm" runat="server"
                  Text="Confirm to delete this record" Font-Bold="True"
                  ForeColor="#FF3300">
          </asp:Label></td>
          <td>
             <asp:Button ID="btnConfirmDelete" runat="server" Text="Delete"/>
          </td>
          <td>
             <asp:Button ID="btnCancel" runat="server" Text="Cancel" />
          </td>
        </tr>
      </table>
    </asp:Panel>

    </div>
   </div>
 </asp:Content>
```

Change to the Design view. A panel will appear when a record is to be deleted, requesting that the user confirms their decision.



Double click the '***Delete***' and '***Cancel***' buttons to create ***button_click( )*** methods.

Select the C# code page for ***staffBoats2.aspx***. Go to the start of the ***Page_Load( )*** method and add lines of code to set the visibility of the screen components. Also modify the ***IF*** condition.

```
protected void Page_Load(object sender, EventArgs e)
{
    btnUpload.Visible = true;
    btnSave.Visible = true;
    FileUpload1.Visible = true;
    pnlConfirm.Visible = false;
    btnCancelRecord.Visible = true;
    action = Request.QueryString["action"];
    boatIDwanted = Convert.ToInt16(Request.QueryString["boatID"]);
    if (txtBoatName.Text.Length == 0)
    {
        if (action == "edit" || action == "delete")
        {
            for (int i = 0; i < canalBoat.boatCount; i++)
            {
                if (canalBoat.boatObject[i].boatID == boatIDwanted)
                {
                    txtBoatName.Text = canalBoat.boatObject[i].boatName;
```

The correct buttons would be visible in the case of a record being edited, but we need to display different options if the record is being deleted. Add a block of lines at the end of the ***Page_Load( )*** method to do this.

```
                else
                {
                    Image1.ImageUrl = "";
                }
                picbyte = null;
            }
        }
    }
    if (action == "delete")
    {
        btnUpload.Visible = false;
        btnSave.Visible = false;
        FileUpload1.Visible = false;
        pnlConfirm.Visible = true;
        btnCancelRecord.Visible = false;
    }
}
```

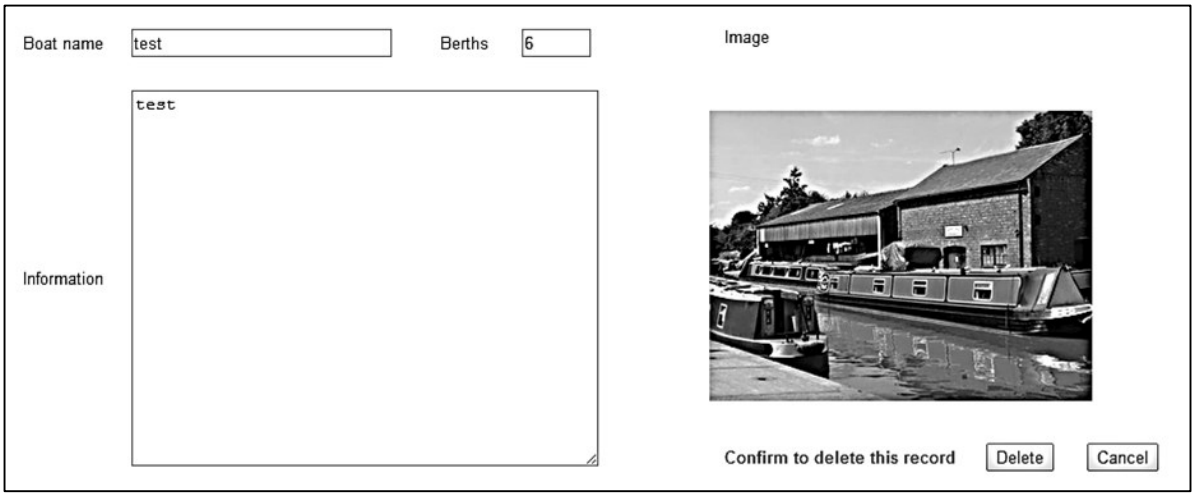
Move to the bottom of the C# page and add lines of code to the ***button_click( )*** methods.

```
protected void btnConfirmDelete_Click(object sender, EventArgs e)
{
    lblMessage.Text = canalBoat.deleteBoat(boatIDwanted);
    canalBoat.loadBoats();
}

protected void btnCancel_Click(object sender, EventArgs e)
{
    Response.Redirect("staffBoats.aspx?berthsWanted=2");
}
```

Go now to the ***canalBoat.cs*** class file and add a deleteBoat( ) method.

```
public double lowerDay { get; set; }
public bool pictureLoaded { get; set; }

public static string deleteBoat(int boatSelected)
{
    string message = "";
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmStock = new SqlCommand();
        cmStock.Connection = cnTB;
        cmStock.CommandType = CommandType.Text;
        string query;
        query = "DELETE FROM boats WHERE boatID='" + boatSelected + "'";
        message = query;
        SqlCommand cmd = new SqlCommand(query, cnTB);
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record deleted";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

Build and run the ***staffBoats*** web page. Use the '***Add boat***' option to create a test record. Select this record through the menu system, then click the **delete** option. The boat details should now be displayed, along with the '***confirm***' message and buttons.



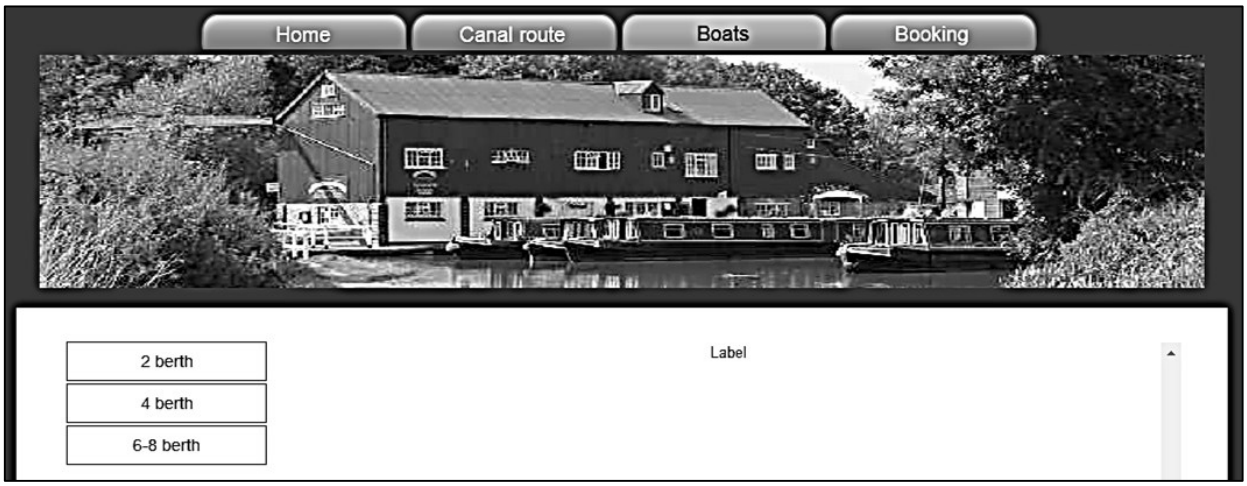
Click the '***Cancel***' button. The program returns to the boat menu. Check that the test record is still present.
Repeat the delete process, but this time click the '***Delete***' button. Check that the record has now been removed.

This completes that staff boat editing section. We will now work on the public web page display.

Close the browser and stop debugging. Select the ***boats.aspx*** page and add lines of code to the '***content2***' section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="boatContent" >
   <div id="boats">
      <ul id="cat">
        <li><a href="boats.aspx?berthsWanted=2">2 berth</a></li>
        <li><a href="boats.aspx?berthsWanted=4">4 berth</a></li>
        <li><a href="boats.aspx?berthsWanted=6">6-8 berth</a></li>
      </ul>
   </div>
   <div id="boatDisplay">
     <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
   </div>
  </div>
</asp:Content>
```

Build and run the ***boats.aspx*** web page. Check that a menu is displayed, in a similar way to the staff page.



Close the browser and stop debugging. Go to the C# page for ***boats.aspx*** and add lines of code to the ***Page_Load( )*** method.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (canalBoat.boatCount == 0)
    {
       canalBoat.loadBoats();
    }
    int berthsWanted = Convert.ToInt16(Request.QueryString["berthsWanted"]);
    string s = "";
    s += "<table cellpadding=8 cellspacing =8 bgcolor=white>";
    int photoID;
    for (int i = 0; i < canalBoat.boatCount; i++)
    {
       int berthsProvided = canalBoat.boatObject[i].berths;
       if (berthsProvided > 6)
       {
          berthsProvided = 6;
       }
       if (berthsWanted == berthsProvided)
       {
          s += "<tr>";
          s += "<td >";
          s += "<h3>" + canalBoat.boatObject[i].boatName + "</h3>";
          s += canalBoat.boatObject[i].berths + " berths <br><br>";
          s += canalBoat.boatObject[i].boatDescription + "<br>";
          double priceWeek = canalBoat.boatObject[i].higherWeek;
          string priceStringWeek = String.Format("{0:.##}", priceWeek);
          double priceDay = canalBoat.boatObject[i].higherDay;
          string priceStringDay = String.Format("{0:.##}", priceDay);
          s += "<h3>Peak:     weekly £" + priceStringWeek +
               "    daily £" + priceStringDay + "</h3>";
```

```
            priceWeek = canalBoat.boatObject[i].lowerWeek;
            priceStringWeek = String.Format("{0:.##}", priceWeek);
            priceDay = canalBoat.boatObject[i].lowerDay;
            priceStringDay = String.Format("{0:.##}", priceDay);
            s += "<h3>Off-peak:     weekly £" + priceStringWeek +
                 "    daily £" + priceStringDay + "</h3>";
            s += "<td >";
            photoID = canalBoat.boatObject[i].boatID;
            if (canalBoat.boatObject[i].pictureLoaded == true)
            {
                s += "<img src='Handler3.ashx?imgid=" + photoID +
                                                "' width='200' border='0' >";
            }
            s += "</td>";
            s += "</tr>";
            s += "<tr>";
            s += "<td><h3>";
            s += "<a href='booking.aspx?boatID=";
            s += canalBoat.boatObject[i].boatID;
            s += "'> Check available dates </a></h3></td>";
            s += "</tr>";
            s += "<tr>";
            s += "<td colspan=3><hr></td>";
            s += "</tr>";
          }
        }
        s += "</table>";
        Label1.Text = s;
}
```

Build and run the ***boats.aspx*** web page. Check that boats in each size category are displayed correctly. Notice that a link is provided for each boat, to allow the user to check availability. We will work on the booking procedure in the next section.