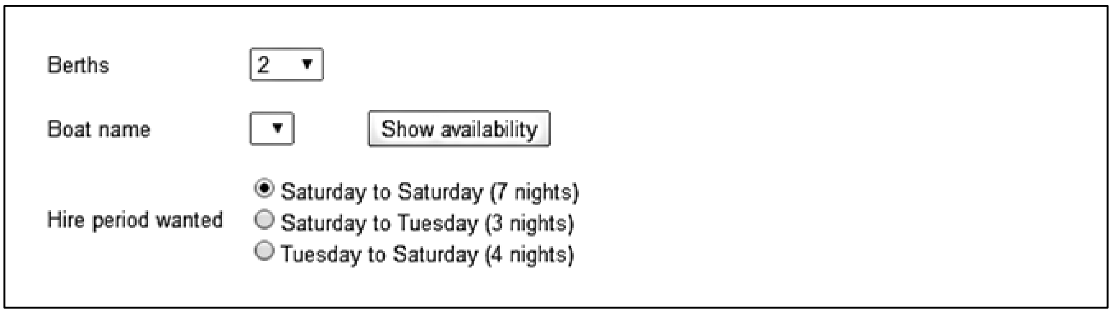
Close the browser and stop debugging. Go to the ***booking.aspx*** page and add code to the '***content2***' section. This will provide drop down lists for selecting the number of berths and boat name, and choosing a week, weekend or mid-week hire period.

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

  <div id="leftColumn">
    <table cellpadding="6">
      <tr>
         <td>Berths</td>
         <td>
            <asp:DropDownList ID="ddlBerths" runat="server" AutoPostBack="True" >
               <asp:ListItem>2</asp:ListItem>
               <asp:ListItem>4</asp:ListItem>
               <asp:ListItem>6-8</asp:ListItem>
            </asp:DropDownList>
        </td>
     </tr>
     <tr>
        <td>Boat name</td>
        <td>
           <asp:DropDownList ID="ddlBoatNames" runat="server">
           </asp:DropDownList>
        </td>
        <td>
          <asp:Button ID="btnAvailability" runat="server"
                                                    Text="Show availability" />
        </td>
     </tr>
     <tr>
        <td>Hire period wanted</td>
        <td colspan="2">
          <asp:RadioButton ID="rbWeek" runat="server" GroupName="nights"
                      Checked="true" Text="Saturday to Saturday (7 nights)" />
          <br />
          <asp:RadioButton ID="rbWeekend" runat="server" GroupName="nights"
                                    Text="Saturday to Tuesday (3 nights)" />
          <br />
          <asp:RadioButton ID="rbMidweek" runat="server" GroupName="nights"
                                    Text="Tuesday to Saturday (4 nights)" />
        </td>
      </tr>
    </table>
  </div>

</asp:Content>
```

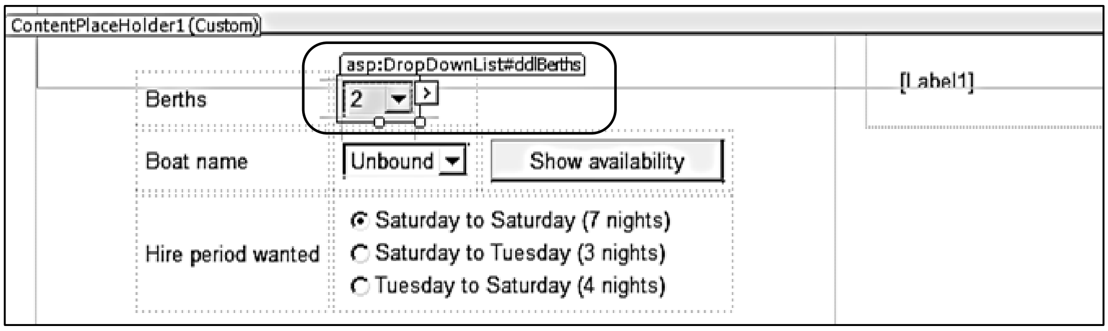
Build and run the ***booking.aspx*** web page. Check that the components are displayed correctly. There are currently no boat names shown in the second ***drop down list***. We will arrange for the correct names to be added when the number of berths is selected. We have combined the three ***radio buttons*** into a group, called '***nights***', so that only one of the buttons can be selected at a time.


Berths
2
Boat name
Show availability
Hire period wanted
Saturday to Saturday (7 nights)
Saturday to Tuesday (3 nights)
Tuesday to Saturday (4 nights)

Close the browser and stop debugging. Go to the end of 'content2' section of the ***booking.aspx*** page and add a 'rightColumn' division containing a label.

```
          <asp:RadioButton ID="rbMidweek" runat="server" GroupName="nights"
                                        Text="Tuesday to Saturday (4 nights)" />
        </td>
      </tr>
    </table>
  </div>
  <div id="rightColumn">
       <asp:Label ID="Label1" runat="server"></asp:Label>
  </div>
</asp:Content>
```

Change to the Design view. Select the ***ddlBerths*** drop down list component and double click to create a ***SelectedIndexChanged( )*** method.


ContentPlaceHolder1 (Custom)
asp:DropDownList#ddlBerths
Berths
2
[Label1]
Boat name
Unbound
Show availability
Hire period wanted
Saturday to Saturday (7 nights)
Saturday to Tuesday (3 nights)
Tuesday to Saturday (4 nights)

Add code to the method.

```
    protected void ddlBerths_SelectedIndexChanged(object sender, EventArgs e)
    {
       setBoatNames(ddlBerths.Text);
       Label1.Text =
             "Please select a boat and click button to show availability...";
    }
```

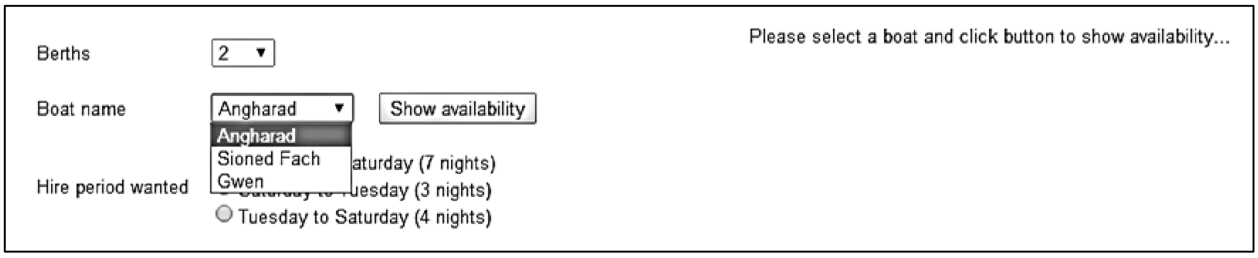
Immediately after this method, insert the code for a ***setBoatNames( )*** method. This will check through the ***canalBoat objects*** and pick out those which have the requested number of berths. The boat names will be added to the second drop down list.

```
protected void setBoatNames(string berths)
{
   ddlBoatNames.Items.Clear();
   if (berths == "6-8")
   {
      berths = "6";
   }
   int berthsWanted = Convert.ToInt16(berths);
   for (int i = 0; i < canalBoat.boatCount; i++)
   {
      int berthsProvided = canalBoat.boatObject[i].berths;
      if (berthsProvided > 6)
      {
         berthsProvided = 6;
      }
      if (berthsWanted == berthsProvided)
      {
         ddlBoatNames.Items.Add(canalBoat.boatObject[i].boatName);
      }
   }
}
```

Before running the page, we must load the ***canalBoat objects***. We can also set the web page to initially list the 2 berth boats, and the user may then re-select a different size category if they wish. Go to the ***Page_Load( )*** method near the start of the ***booking.aspx*** C# code page and add code to carry out these tasks.

```
protected void Page_Load(object sender, EventArgs e)
{
   if (canalBoat.boatCount == 0)
   {
       canalBoat.loadBoats();
   }
   if (ddlBoatNames.Text == "")
   {
      setBoatNames("2");
      Label1.Text =
           "Please select a boat and click button to show availability...";
   }
}
```

Build and run the booking.aspx web page. Check that the correct boat names are displayed in the drop down list for each size category.



Close the browser and stop debugging. Go to the ***booking.aspx*** page Design view and double click the 'Show availability' button to create a ***button_click( )*** method.

```
protected void btnAvailability_Click(object sender, EventArgs e)
{
    boatNameWanted = Convert.ToString(ddlBoatNames.SelectedItem);
    displayBoat(boatNameWanted);
}
```

Add variables at the start of the code page.

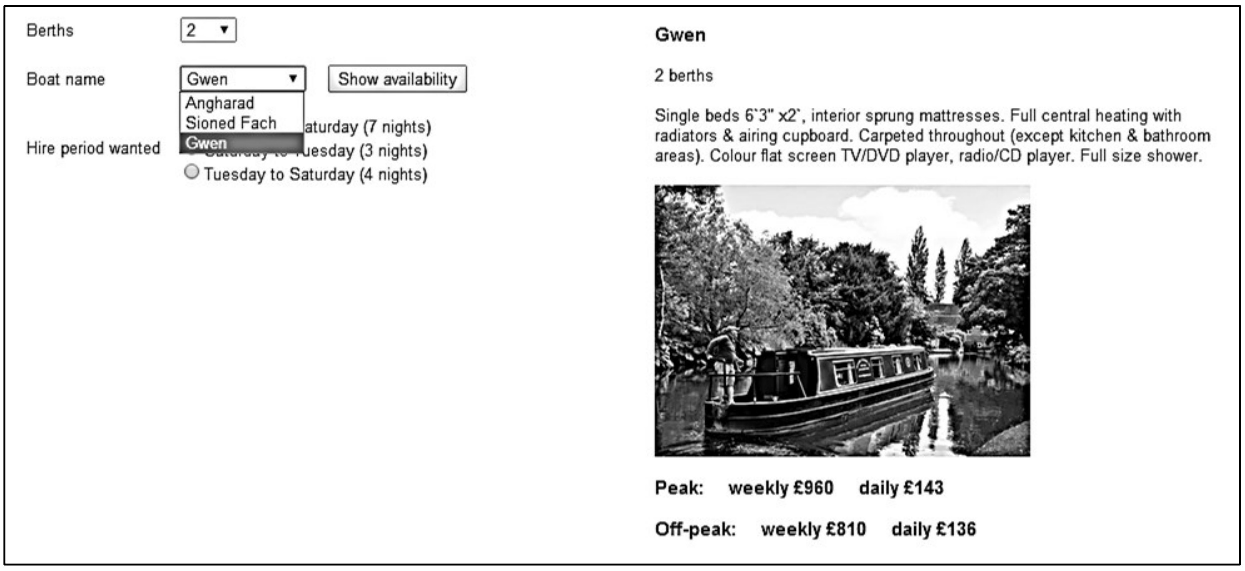
```
public partial class booking : System.Web.UI.Page
{
   public static string boatNameWanted;
   public static int boatIDwanted;
   protected void Page_Load(object sender, EventArgs e)
   {
```

Return to the bottom of the code page and add a ***displayBoat( )*** method after the button_click method. This will display the details of the selected boat by means of the ***label*** component in the right hand column. We can do this easily by copying code which has already been written for the ***boats*** web page.

```
protected void displayBoat(string boatWanted)
{
   string s = "";
   string boatName;
   int photoID;

   for (int i = 0; i < canalBoat.boatCount; i++)
   {
      boatName = canalBoat.boatObject[i].boatName;
      if (boatName == boatWanted)
      {
          boatIDwanted = canalBoat.boatObject[i].boatID;
          s += "<h3>" + canalBoat.boatObject[i].boatName + "</h3>";
          s += canalBoat.boatObject[i].berths + " berths <br><br>";
          s += canalBoat.boatObject[i].boatDescription + "<br><br>";
```

```
                photoID = canalBoat.boatObject[i].boatID;
                if (canalBoat.boatObject[i].pictureLoaded == true)
                {
                    s += "<img src='Handler3.ashx?imgid=" + photoID + "' width='300'
                                                             border='0' >";
                }

                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: &nbsp;&nbsp;&nbsp; weekly £" + priceStringWeek +
                               " &nbsp;&nbsp;&nbsp; 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: &nbsp;&nbsp;&nbsp; weekly £" + priceStringWeek +
                               " &nbsp;&nbsp;&nbsp; daily £" + priceStringDay + "</h3>";
                }
            }
            Label1.Text = s;
        }
```

Build and run the web page. Check that boat details are displayed correctly when a boat is selected from the drop down list.



Gwen

2 berths

Single beds 6'3" x2', interior sprung mattresses. Full central heating with radiators & airing cupboard. Carpeted throughout (except kitchen & bathroom areas). Colour flat screen TV/DVD player, radio/CD player. Full size shower.

Peak: weekly £960 daily £143

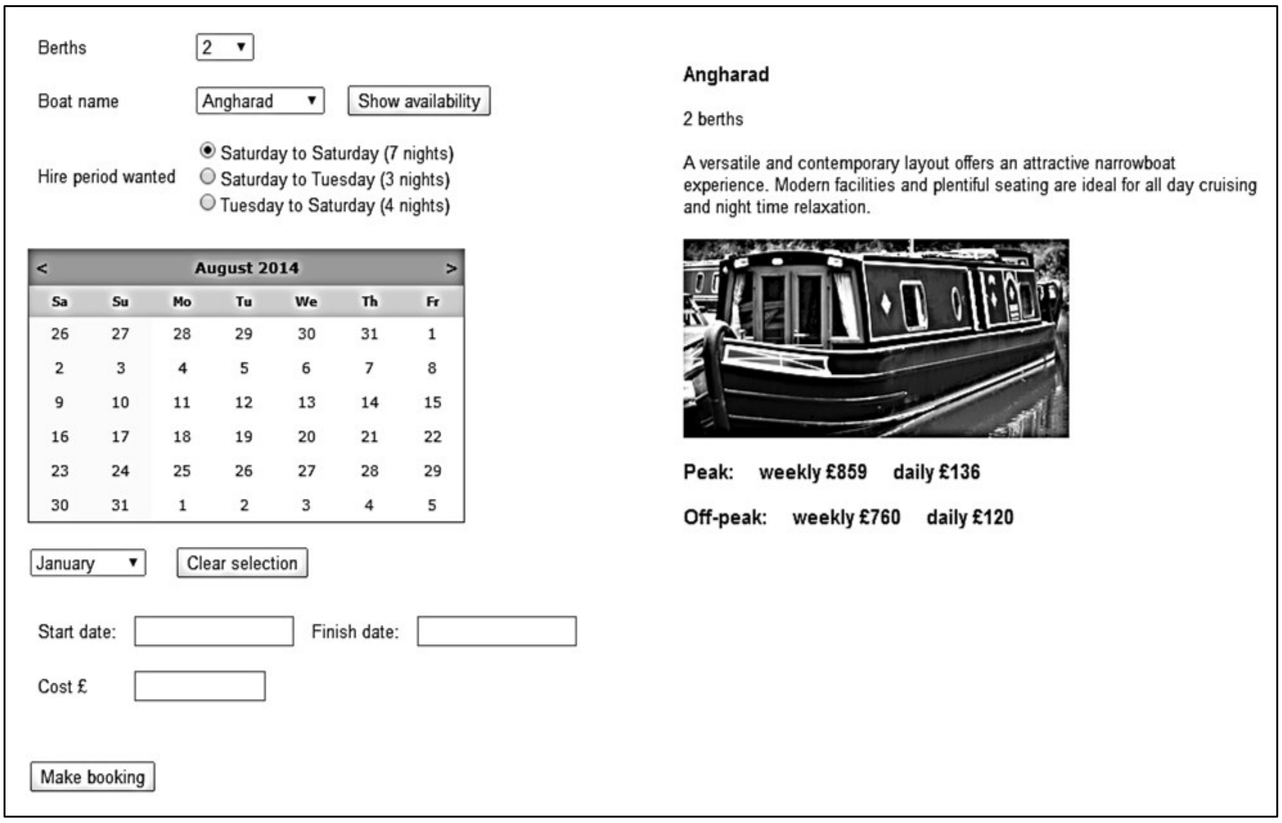
Off-peak: weekly £810 daily £136

Close the browser and stop debugging. The next step is to provide an interactive ***calendar.*** This will display the available hire dates for any selected boat, and allow the user to input their choice of hire period. We will use similar techniques to the ***Holiday cottage bookings*** project in Chapter 7.

Go to the booking.aspx page and add code at the end of the 'leftColumn' division, following the set of radio group components.

```
        <asp:RadioButton ID="rbMidweek" runat="server" GroupName="nights"
                                    Text="Tuesday to Saturday (4 nights)" />
      </td>
    </tr>
  </table>
  <br />
  <asp:Calendar ID="Calendar1" runat="server" BackColor="White"
     BorderColor="#999999" CellPadding="4" DayNameFormat="Shortest"
     Font-Names="Verdana" Font-Size="8pt" ForeColor="Black"
     Height="200px" Width="340px" FirstDayOfWeek="Saturday" >
     <DayHeaderStyle BackColor="#CCCCCC" Font-Bold="True" Font-Size="7pt" />
     <NextPrevStyle VerticalAlign="Bottom" />
     <OtherMonthDayStyle ForeColor="#808080" />
     <SelectedDayStyle BackColor="#666666" Font-Bold="True" ForeColor="White" />
     <SelectorStyle BackColor="#CCCCCC" />
     <TitleStyle BackColor="#999999" BorderColor="Black" Font-Bold="True" />
     <WeekendDayStyle BackColor="#FFFFCC" />
  </asp:Calendar>
  <br />
  <asp:DropDownList ID="ddlMonth" runat="server" AutoPostBack="True" >
      <asp:ListItem Value="1">January</asp:ListItem>
      <asp:ListItem Value="2">February</asp:ListItem>
      <asp:ListItem Value="3">March</asp:ListItem>
      <asp:ListItem Value="4">April</asp:ListItem>
      <asp:ListItem Value="5">May</asp:ListItem>
      <asp:ListItem Value="6">June</asp:ListItem>
      <asp:ListItem Value="7">July</asp:ListItem>
      <asp:ListItem Value="8">August</asp:ListItem>
      <asp:ListItem Value="9">September</asp:ListItem>
      <asp:ListItem Value="10">October</asp:ListItem>
      <asp:ListItem Value="11">November</asp:ListItem>
      <asp:ListItem Value="12">December</asp:ListItem>
  </asp:DropDownList>
  &nbsp;&nbsp;&nbsp;
  <asp:Button ID="btnClear" runat="server" Text="Clear selection" />
  <br /><br />
  <table cellpadding="6">
    <tr>
       <td> Start date: </td>
       <td>
          <asp:TextBox runat="server" ID="txtStartDate" Width="120px"></asp:TextBox>
       </td>
       <td>Finish date: </td>
       <td>
          <asp:TextBox runat="server" ID="txtFinishDate" Width="120px"></asp:TextBox>
       </td>
   </tr>
       <tr>
          <td>Cost £</td>
         <td>
           <asp:TextBox runat="server" ID="txtCost" Width="98px"></asp:TextBox>
         </td>
       </tr>
     </table>
     <br /><br />
     <asp:Button ID="btnBook" runat="server" Text="Make booking" />
     <br /><br />
     <center>
         <asp:Label ID="lblMessage" runat="server"></asp:Label>
     </center>

   </div>
   <div id="rightColumn">
      <br />
      <asp:Label ID="Label1" runat="server"></asp:Label>
  </div>
```

Build and run the ***booking.aspx*** web page. A ***calendar***, ***text boxes*** and ***buttons*** should have been added. Check that the ***drop down list*** below the calendar shows a list of months.



Angharad

2 berths

A versatile and contemporary layout offers an attractive narrowboat experience. Modern facilities and plentiful seating are ideal for all day cruising and night time relaxation.

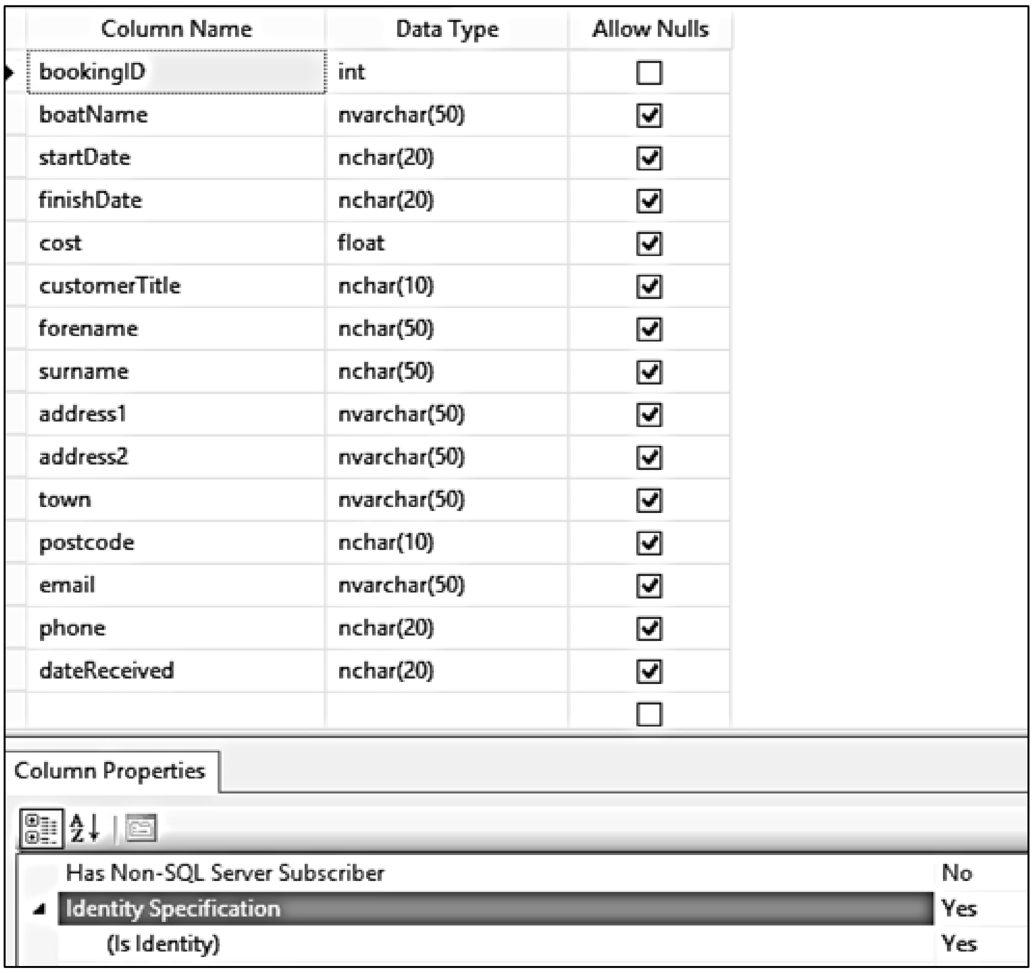
Peak: weekly £859 daily £136

Off-peak: weekly £760 daily £120

Close the browser and stop debugging.

Before going further with programming the on-line booking system, we need to set up a database table to hold the bookings received.

Go to the Server Explorer window and open the ***canalHolidays.mdf*** database. Right click the ***Tables*** icon and select '***Add New Table***'. Insert fields as shown below, making the ***bookingID*** field an auto-number by setting the ***Identity Specification*** / ***(Is Identity)*** property to '***Yes***'



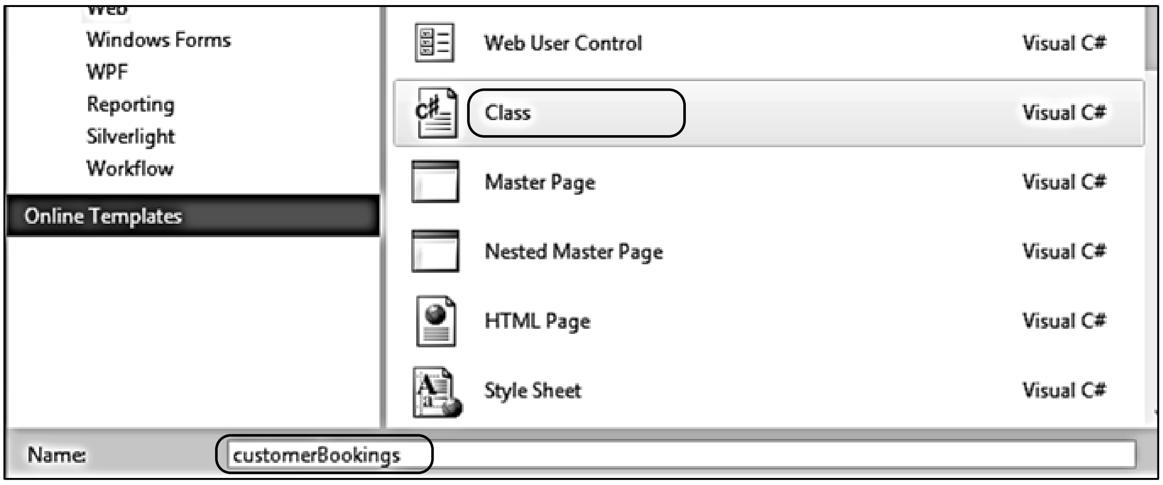
Column Name	Data Type	Allow Nulls
bookingID	int	☐
boatName	nvarchar(50)	✓
startDate	nchar(20)	✓
finishDate	nchar(20)	✓
cost	float	✓
customerTitle	nchar(10)	✓
forename	nchar(50)	✓
surname	nchar(50)	✓
address1	nvarchar(50)	✓
address2	nvarchar(50)	✓
town	nvarchar(50)	✓
postcode	nchar(10)	✓
email	nvarchar(50)	✓
phone	nchar(20)	✓
dateReceived	nchar(20)	✓
		☐

Column Properties

Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes

Close the table and give the name '***bookings***'.

We now require an object class to hold the bookings. Go to the Solution Explorer window and right click the **Canal holidays** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name '**customerBookings**'.



Name: customerBookings

Open the ***customerBookings.cs*** class file and add code as shown below. We begin by including directives to include the ***System.Data.SqlClient*** and ***System.Data*** code modules when the class is compiled. Properties are set up for ***bookingObject*** to hold the data from fields of the ***bookings*** database table. A ***loadBookings( )*** method is then produced to create the set of ***customerBooking*** objects.

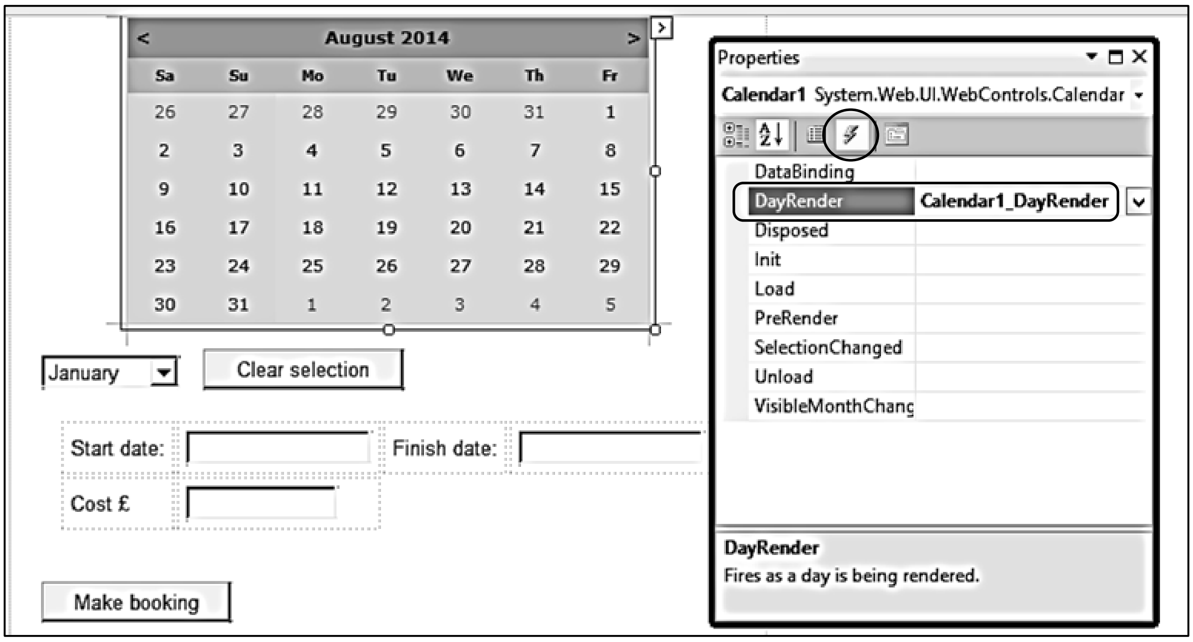
```
using System.Linq;
using System.Web;

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

namespace Canal_holidays
{
   public class customerBookings
   {
      public static int bookingCount = 0;
      public static customerBookings[] bookingObject = new customerBookings[200];
      public static string databaseLocation =
                             "C:\\WEB APPLICATIONS\\canalHolidays.mdf;";
      public int bookingID { get; set; }
      public string boatName { get; set; }
      public string startDate { get; set; }
      public string finishDate { get; set; }
      public double cost { get; set; }
      public string customerTitle { get; set; }
      public string forename { get; set; }
      public string surname { get; set; }
      public string address1 { get; set; }
      public string address2 { get; set; }
      public string town { get; set; }
      public string postcode { get; set; }
      public string email { get; set; }
      public string phone { get; set; }
      public string dateReceived { get; set; }

      public static void loadBookings()
      {
         DataSet dsBookings = new DataSet();
         SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
             AttachDbFilename=" + databaseLocation + "Integrated Security=True;
             Connect Timeout=30; User Instance=True");
         try
         {
             cnTB.Open();
             SqlCommand cmBooking = new SqlCommand();
             cmBooking.Connection = cnTB;
             cmBooking.CommandType = CommandType.Text;
             cmBooking.CommandText = "SELECT * FROM bookings";
             SqlDataAdapter daBooking = new SqlDataAdapter(cmBooking);
             daBooking.Fill(dsBookings);
             cnTB.Close();
```

```
            int countRecords = dsBookings.Tables[0].Rows.Count;
            bookingCount = 0;
            for (int i = 0; i < countRecords; i++)
            {
                DataRow drBooking = dsBookings.Tables[0].Rows[i];
                int bookingID = (int)drBooking[0];
                string boatName = Convert.ToString(drBooking[1]);
                string startDate = Convert.ToString(drBooking[2]);
                string finishDate = Convert.ToString(drBooking[3]);
                double cost = Convert.ToDouble(drBooking[4]);
                string customerTitle = Convert.ToString(drBooking[5]);
                string forename = Convert.ToString(drBooking[6]);
                string surname = Convert.ToString(drBooking[7]);
                string address1 = Convert.ToString(drBooking[8]);
                string address2 = Convert.ToString(drBooking[9]);
                string town = Convert.ToString(drBooking[10]);
                string postcode = Convert.ToString(drBooking[11]);
                string email = Convert.ToString(drBooking[12]);
                string phone = Convert.ToString(drBooking[13]);
                string dateReceived = Convert.ToString(drBooking[14]);
                bookingObject[bookingCount] = new customerBookings();
                bookingObject[bookingCount].bookingID = bookingID;
                bookingObject[bookingCount].boatName = boatName;
                bookingObject[bookingCount].startDate = startDate;
                bookingObject[bookingCount].finishDate = finishDate;
                bookingObject[bookingCount].cost = cost;
                bookingObject[bookingCount].customerTitle = customerTitle;
                bookingObject[bookingCount].forename = forename;
                bookingObject[bookingCount].surname = surname;
                bookingObject[bookingCount].address1 = address1;
                bookingObject[bookingCount].address2 = address2;
                bookingObject[bookingCount].town = town;
                bookingObject[bookingCount].postcode = postcode;
                bookingObject[bookingCount].email = email;
                bookingObject[bookingCount].phone = phone;
                bookingObject[bookingCount].dateReceived = dateReceived;
                bookingCount++;
            }
        }
        catch
        {

        }
    }
```

Return to ***booking.aspx*** and open the C# code page. Add a '***using System.Drawing***' directive at the start of the page.

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

Go now to the ***booking.aspx*** design view. Select the ***calendar*** component. Click the '***methods***' button in the Properties window, indicated by a lightening flash icon. Double click alongside the '***DayRender***' method. This will create a method which operates as each day is added to the calendar, and will allow us to show dates booked or available using different colour formatting.



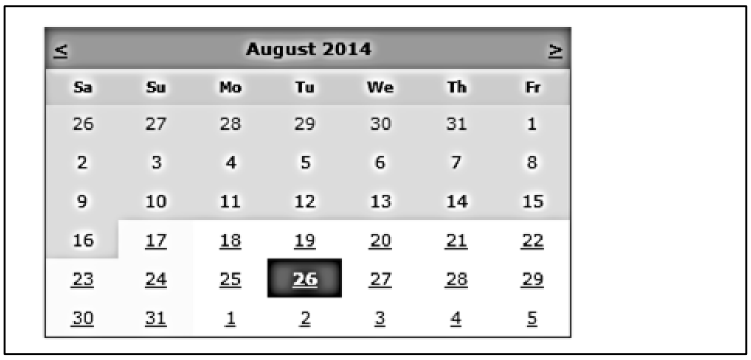
DataBinding	
DayRender	Calendar1_DayRender
Disposed	
Init	
Load	
PreRender	
SelectionChanged	
Unload	
VisibleMonthChang	

DayRender
Fires as a day is being rendered.

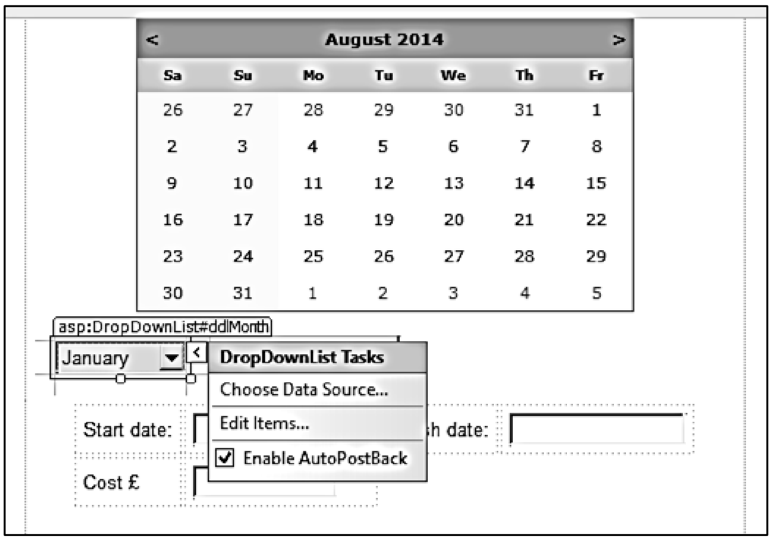
Bookings cannot be made for dates which have passed. Add code to the ***DayRender( )*** method to deactivate these dates.

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    DateTime currentDate = Calendar1.TodaysDate;
    if (e.Day.Date <= currentDate)
    {
       e.Day.IsSelectable = false;
       e.Cell.BackColor = Color.Gainsboro;
    }
}
```

Build and run the ***booking.aspx*** web page. Check that all dates up to and including today's date are shown in grey. A date in the future can be selected by mouse click, but past dates should not be selectable.



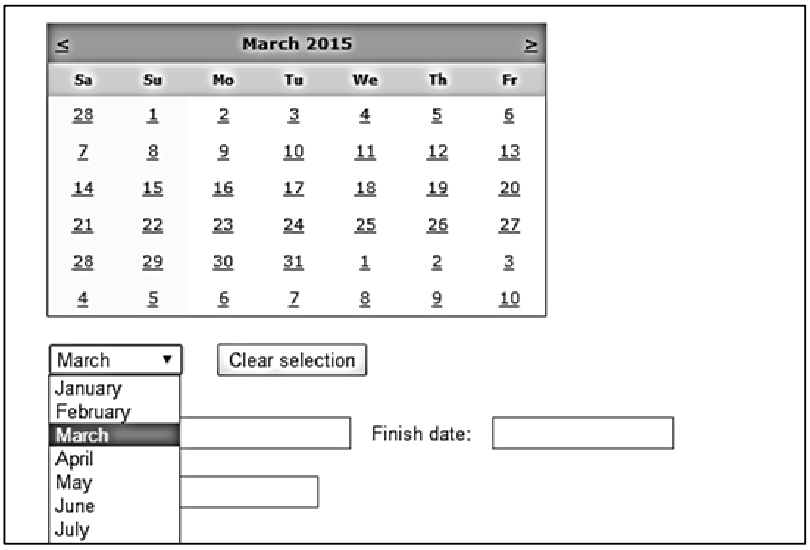
Close the browser and stop debugging. Return to the ***booking.aspx*** design view. Select the ***ddlMonth*** component and click the small arrow icon to open the task list. Make sure that there is a tick alongside '***Enable AutoPostback***' – this is necessary to ensure the web page responds immediately if a different month is selected.



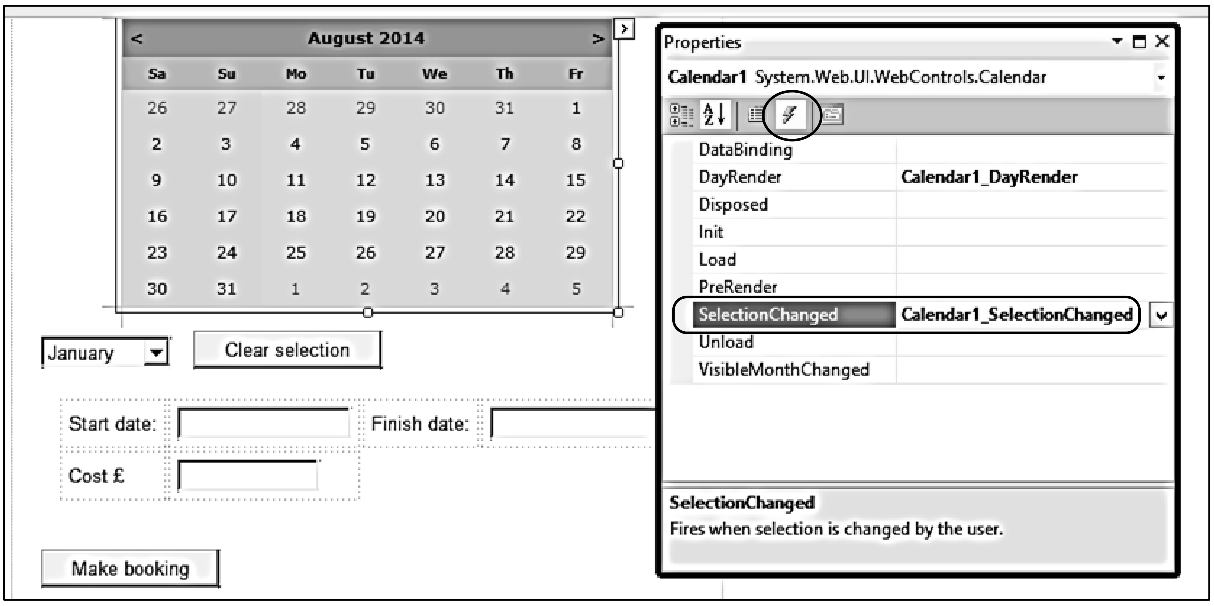
Double click the ***ddlMonth*** component to create a ***SelectedIndexChanged( )*** method and add lines of code. If a month is selected which comes before the current month name, we will assume that it refers to the next calendar year. For example, if in ***August 2014*** the user selects the month of ***March***, we will select ***March 2015*** on the calendar.

```
protected void ddlMonth_SelectedIndexChanged(object sender, EventArgs e)
{
     int month = Convert.ToInt16(ddlMonth.SelectedValue);
     int thisMonth = DateTime.Today.Month;
     int thisYear = DateTime.Today.Year;
     if (month < thisMonth)
         thisYear++;
     string dateString = "01-" + month + "-" + thisYear;
     Calendar1.VisibleDate = Convert.ToDateTime(dateString);
}
```

Build and run the ***booking.aspx*** web page. Check that months can be selected correctly.



Close the web browser and stop debugging. Return to ***booking.aspx*** and change to the design view. Select the ***calendar*** component. Click the '***methods***' button in the Properties window, indicated by the lightening flash icon. Double click alongside the '***SelectionChanged***' method. This will create a method which can carry out actions when the user clicks on a date.



DataBinding	
DayRender	Calendar1_DayRender
Disposed	
Init	
Load	
PreRender	
SelectionChanged	Calendar1_SelectionChanged
Unload	
VisibleMonthChanged	

SelectionChanged
Fires when selection is changed by the user.

We will now add code to the ***SelectionChanged( )*** method. The intention is that the user clicks on any date within the period that they wish to book, and the calendar will indicate the nights selected.

- For a ***week's booking***, Saturday to Friday will be highlighted
- For a ***weekend booking***, Saturday to Monday will be highlighted
- For a ***midweek booking***, Tuesday to Friday will be highlighted.

The program does this by identifying the day selected on the calendar, then filling in the remaining days according to the hire period wanted. We must allow for an incorrect day being selected, for example if the user requires a weekend booking but clicks on a midweek date. In this case, the selection will be removed from the calendar and the user must click on another date.

Go to the ***SelectionChanged( )*** method and add lines of code. Also create the ***clearEntries( )*** method immediately following ***SelectionChanged( )*** .

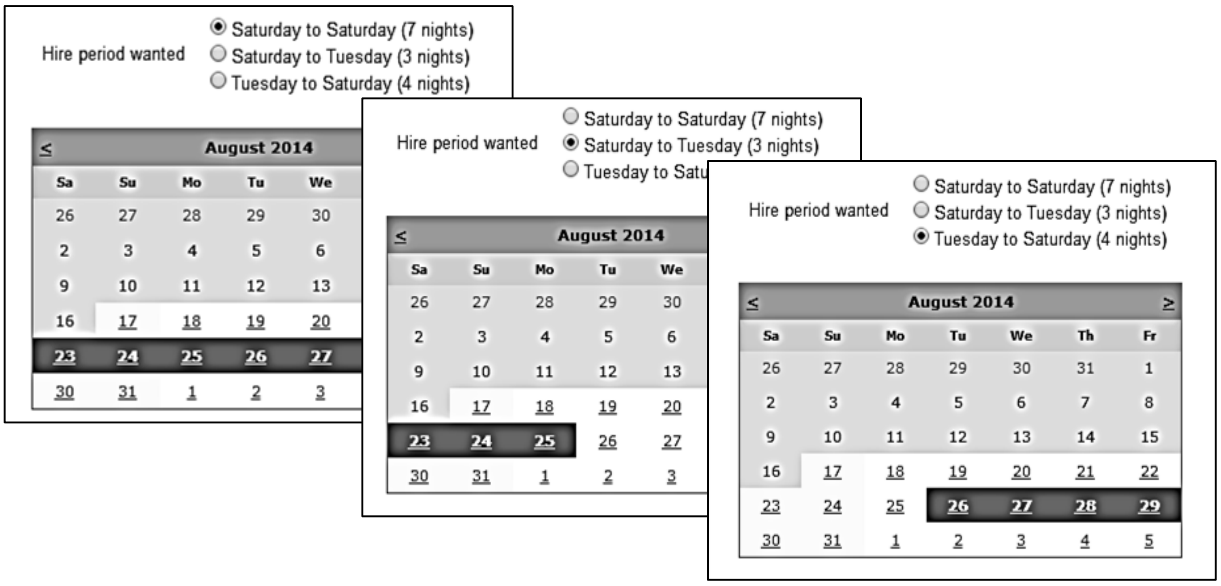
```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    string periodStart = "";
    string periodFinish = "";
    int period = 0;
    DateTime currentDate = Calendar1.SelectedDate;
    int daynumber = Convert.ToInt16(currentDate.DayOfWeek);
    bool dayValid = true;
    DateTime startDate;
    DateTime finishDate;
    string currentDay = Convert.ToString(currentDate.DayOfWeek);
    string startMonth="";
    string finishMonth="";
    int startM=0;
    int finishM=0;

    if (rbWeek.Checked == true)
    {
        periodStart = "Saturday";
        periodFinish = "Friday";
        period=7;
    }
    if (rbWeekend.Checked == true)
    {
        periodStart = "Saturday";
        periodFinish = "Monday";
        if (daynumber > 1 && daynumber < 6)
        {
            dayValid = false;
        }
        period=3;
    }
    if (rbMidweek.Checked == true)
    {
        periodStart = "Tuesday";
        periodFinish = "Friday";
        if (daynumber < 2 || daynumber == 6)
        {
            dayValid = false;
        }
        period=4;
    }
    if (dayValid == false)
    {
        clearEntries();
    }
    else
    {
    }
}
```

```
protected void clearEntries()
{
    Calendar1.SelectedDates.Clear();
}
```

The section of code above checks for the starting and finishing nights for the hire period. We can now highlight the corresponding dates on the calendar. Go to the ***ELSE*** condition at the end of the ***SelectionChanged( )*** method and add lines of code. This is similar to the procedure we used in the ***Holiday cottage bookings*** project for highlighting a complete week's booking.

```
    if (dayValid == false)
     {
         clearEntries();
     }
     else
     {
         while (currentDay != periodStart)
         {
             currentDate = currentDate.AddDays(-1);
             currentDay = Convert.ToString(currentDate.DayOfWeek);
             Calendar1.SelectedDates.Add(currentDate);
         }
         currentDate = Calendar1.SelectedDate;
         currentDay = Convert.ToString(currentDate.DayOfWeek);
         while (currentDay != periodFinish)
         {
             currentDate = currentDate.AddDays(1);
             currentDay = Convert.ToString(currentDate.DayOfWeek);
             Calendar1.SelectedDates.Add(currentDate);
         }
     }
```

Build and run the ***booking.aspx*** web page. Check that the correct nights are highlighted when a booking is selected for a ***week***, ***weekend*** or ***midweek*** period.


Hire period wanted [✓ Saturday to Saturday (7 nights)] Saturday to Tuesday (3 nights) Tuesday to Saturday (4 nights)

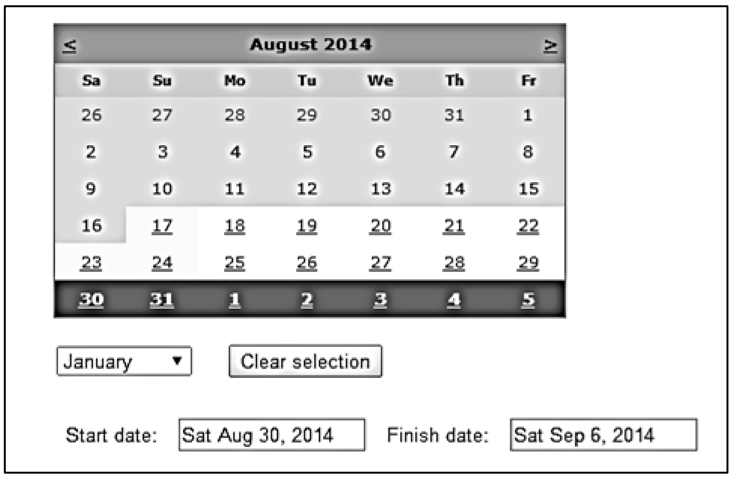
Hire period wanted Saturday to Saturday (7 nights) [✓ Saturday to Tuesday (3 nights)] Tuesday to Saturday (4 nights)

Hire period wanted Saturday to Saturday (7 nights) Saturday to Tuesday (3 nights) [✓ Tuesday to Saturday (4 nights)]

Close the web browser and stop debugging. We can now display the start and finish dates of the holiday in the text boxes below the calendar. Add further lines of code to the ***ELSE*** condition at the end of the ***SelectionChanged( )*** method.

```
else
{
   while (currentDay != periodStart)
   {
       currentDate = currentDate.AddDays(-1);
       currentDay = Convert.ToString(currentDate.DayOfWeek);
       Calendar1.SelectedDates.Add(currentDate);
   }
   startDate = currentDate;
   currentDate = Calendar1.SelectedDate;
   currentDay = Convert.ToString(currentDate.DayOfWeek);
   while (currentDay != periodFinish)
   {
       currentDate = currentDate.AddDays(1);
       currentDay = Convert.ToString(currentDate.DayOfWeek);
       Calendar1.SelectedDates.Add(currentDate);
   }
   currentDate = currentDate.AddDays(1);
   finishDate = currentDate;
   string format = "ddd MMM d, yyyy";
   txtStartDate.Text = startDate.ToString(format);
   txtFinishDate.Text = finishDate.ToString(format);
}
```

Build and run the booking.aspx web page. Check that the start and finish dates of the holiday are shown correctly. Please note that the calendar records the nights when the boat is on hire, whilst the finish date is the last morning of the holiday when the customer returns the boat.



Start date: Sat Aug 30, 2014   Finish date: Sat Sep 6, 2014

In this example, a boat has been hired for seven nights, from 30 August to 5 September, and is then returned on the morning of 6 September. It is necessary to display bookings in this way, as the boat should be shown as available for hire by another customer beginning on 6 September.

Close the browser and stop debugging. The next step is to calculate the price for the holiday. This will depend on:

- The boat chosen,
- The number of nights chosen,
- The time of year, with either ***peak*** or ***off-peak*** rates applying. We will assume that the peak period is from ***April to August***, with off-peak rates applying at other times. We will also assume that if any part of the hire period falls within a peak month, then the peak rate will be applied to the whole of the booking.

Go to the end of the ***SelectionChanged( )*** method and add further lines of code. Create a ***getCost( )*** method immediately below ***SelectionChanged( )***.

```
            finishDate = currentDate;
            string format = "ddd MMM d, yyyy";
            txtStartDate.Text = startDate.ToString(format);
            txtFinishDate.Text = finishDate.ToString(format);
            startMonth = startDate.Month.ToString("00");
            startM = Convert.ToInt16(startMonth);
            finishMonth = finishDate.Month.ToString("00");
            finishM = Convert.ToInt16(finishMonth);
            double cost = getCost(startM, finishM, boatNameWanted, period);
            format = "0.00";
            txtCost.Text = cost.ToString(format);
        }
    }

    protected double getCost(int startM, int finishM, string boatWanted, int period)
    {
        double cost=0;
        string rate="off-peak";
        string boatName;
        double higherWeek=0;
        double higherDay=0;
        double lowerWeek=0;
        double lowerDay=0;

        return cost;
    }
```

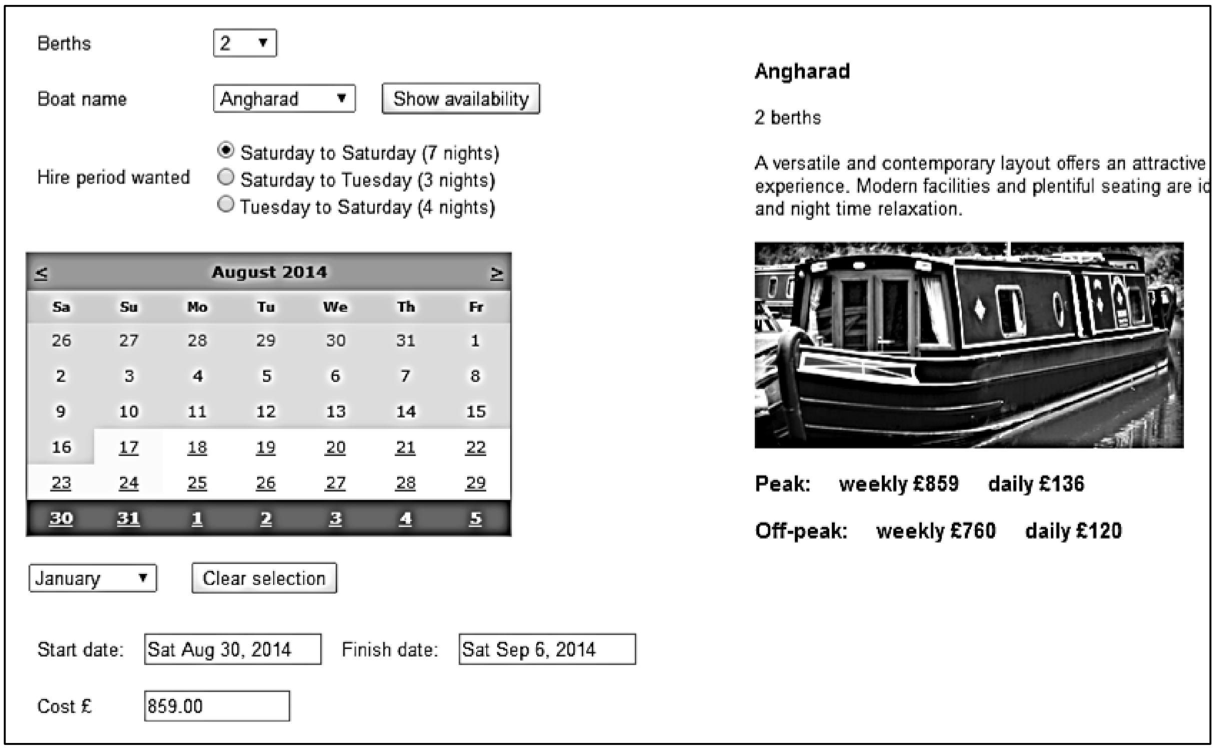
This code obtains the month numbers for the start and finish dates of the holiday. For example, if the holiday begins on 30 August and ends on 6 September, the start month value is 8 and the finish month value is 9. These ***month numbers*** are passed to the ***getCost( )*** method, along with the ***name of the boat*** required and the ***number of nights*** for which it will be hired. We therefore have all the information needed to find the cost of the holiday.

Add code to the ***getCost( )*** method

```
        double lowerWeek=0;
        double lowerDay=0;

        if (finishM >= 4 && startM <= 8)
        {
            rate = "peak";
        }
        for (int i = 0; i < canalBoat.boatCount; i++)
        {
            boatName = canalBoat.boatObject[i].boatName;
            if (boatName == boatWanted)
            {
                higherWeek = canalBoat.boatObject[i].higherWeek;
                higherDay = canalBoat.boatObject[i].higherDay;
                lowerWeek = canalBoat.boatObject[i].lowerWeek;
                lowerDay = canalBoat.boatObject[i].lowerDay;
            }
        }
        switch (period)
        {
            case 3:
                if (rate == "peak")
                {
                    cost = higherDay * 3;
                }
                else
                {
                    cost = lowerDay * 3;
                }; break;
            case 4:
                if (rate == "peak")
                {
                    cost = higherDay * 4;
                }
                else
                {
                    cost = lowerDay * 4;
                }; break;
            case 7:
                if (rate == "peak")
                {
                    cost = higherWeek;
                }
                else
                {
                    cost = lowerWeek;
                }; break;
        }
        return cost;
    }
```

The method begins by determining whether the start or finish month falls within the peak period. It checks for the required boat and obtains the weekly and daily hire costs. A ***CASE*** structure then selects the number of nights booked, and calculates the cost according to whether the booking is charged at peak or off-peak rate.

Build and run the ***booking.aspx*** web page. Check the price calculations by selecting various different boats, length of hire period and month.


Berths: 2
Boat name: Angharad [Show availability]
Hire period wanted: (●) Saturday to Saturday (7 nights) ( ) Saturday to Tuesday (3 nights) ( ) Tuesday to Saturday (4 nights)

August 2014 calendar; January [Clear selection]

Start date: Sat Aug 30, 2014 Finish date: Sat Sep 6, 2014
Cost £ 859.00

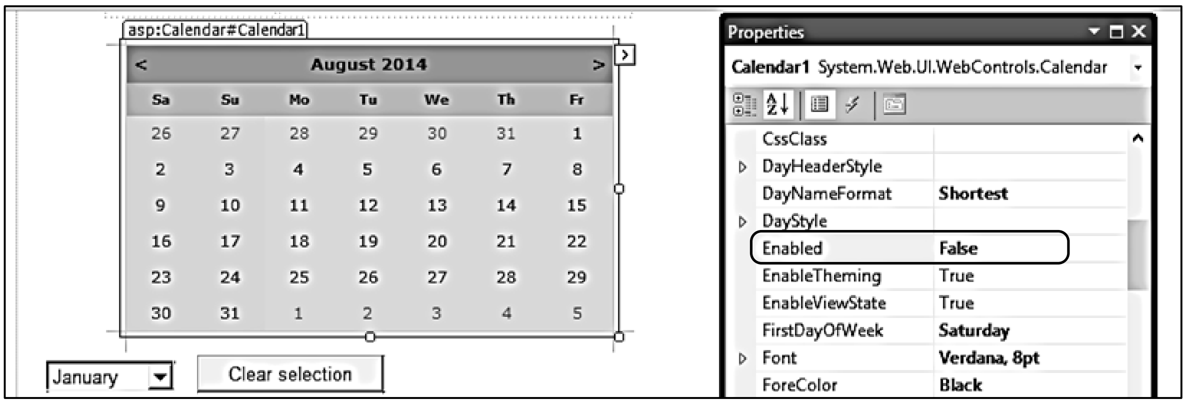
Angharad
2 berths
A versatile and contemporary layout offers an attractive experience. Modern facilities and plentiful seating are id and night time relaxation.

Peak: weekly £859 daily £136
Off-peak: weekly £760 daily £120

Close the browser and stop debugging. During the testing you may have noticed a couple of problems which we will correct now…

If the user selects a holiday date without first selecting a canal boat, an error in the cost occurs. To avoid this, we will disable the calendar until a boat is chosen.

Go to the booking.aspx page and change to Design view. Select the calendar component. Move to the Properties window and set the value of 'Enabled' to false. This means that it will not be possible to select a date on the calendar when the page first opens.


asp:Calendar#Calendar1

Properties
Calendar1 System.Web.UI.WebControls.Calendar

CssClass	
DayHeaderStyle	
DayNameFormat	Shortest
DayStyle	
Enabled	False
EnableTheming	True
EnableViewState	True
FirstDayOfWeek	Saturday
Font	Verdana, 8pt
ForeColor	Black

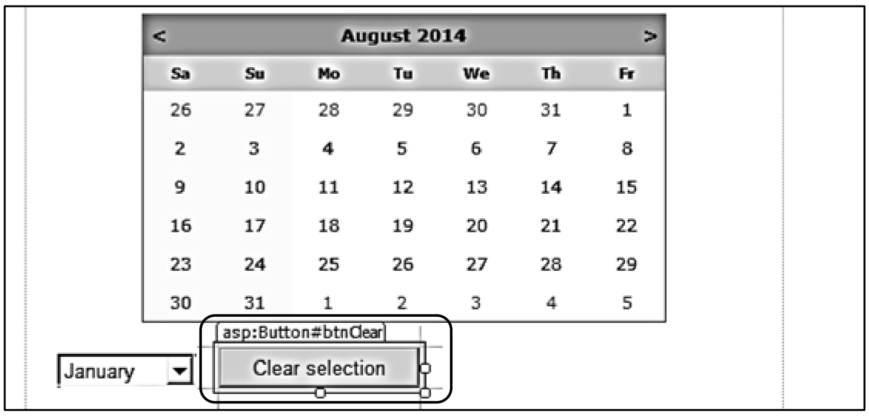
Go now to the C# code page for ***booking.aspx*** and locate the ***displayBoat( )*** method. Add lines of code to enable the calendar so that dates can now be selected.

```
protected void displayBoat(string boatWanted)
{
    string s = "";
    string boatName;
    int photoID;

    Calendar1.Enabled = true;
    clearEntries();

    for (int i = 0; i < canalBoat.boatCount; i++)
    {
```

Another problem found during testing is that the text boxes for start and finish holiday dates and cost are not cleared when the calendar is reset. Locate the ***clearEntries( )*** method and add lines of code to do this.

```
protected void clearEntries()
{
    Calendar1.SelectedDates.Clear();

    txtStartDate.Text = "";
    txtFinishDate.Text = "";
    txtCost.Text = "";
}
```

Return to the ***booking.aspx*** Design view and double click the '***Clear selection***' button to create a button_click( ) method.



Add a line to ***btnClear_Click( )*** method to call the ***clearEntries( )*** method.

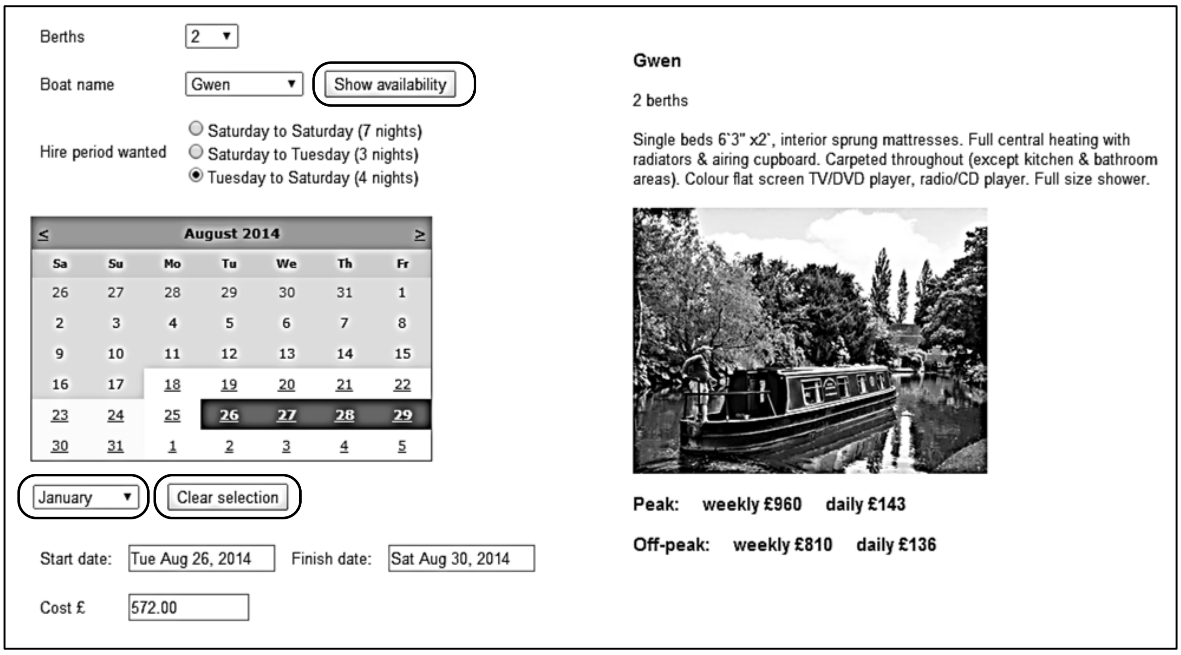
```
protected void btnClear_Click(object sender, EventArgs e)
{
    clearEntries();
}
```

We will also call the ***clearEntries( )*** method if a different month is chosen from the drop down list. Locate the ***ddlMonth_SelectedIndexChanged( )*** method and add the line of code.

```
protected void ddlMonth_SelectedIndexChanged(object sender, EventArgs e)
{
    int month = Convert.ToInt16(ddlMonth.SelectedValue);
    int thisMonth = DateTime.Today.Month;
    int thisYear = DateTime.Today.Year;
    clearEntries();
    if (month < thisMonth)
        thisYear++;
    string dateString = "01-" + month + "-" + thisYear;
    Calendar1.VisibleDate = Convert.ToDateTime(dateString);
}
```

Bulid and run the ***booking.aspx*** web page and check that the changes that you have made are functioning correctly:

- It should not be possible to select a date on the calendar until a boat has been selected.
- The '***clear selection***' button clears both the calendar and the text boxes.
- The text boxes and calendar are cleared if a different month or different boat is selected from the drop down lists.



Berths: 2
Boat name: Gwen
Hire period wanted: Saturday to Saturday (7 nights) / Saturday to Tuesday (3 nights) / [✓ Tuesday to Saturday (4 nights)]

August 2014

Sa	Su	Mo	Tu	We	Th	Fr
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Start date: Tue Aug 26, 2014 Finish date: Sat Aug 30, 2014
Cost £ 572.00

Gwen

2 berths

Single beds 6'3" x2', interior sprung mattresses. Full central heating with radiators & airing cupboard. Carpeted throughout (except kitchen & bathroom areas). Colour flat screen TV/DVD player, radio/CD player. Full size shower.

Peak: weekly £960 daily £143

Off-peak: weekly £810 daily £136

Close the browser and stop debugging. We have now completed the inputs for an independently operating Booking page. You may remember, however, that we also made a link to Bookings from the Boats page. In that case, the user would already have chosen a boat and only needs to select the date of their booking. We will program this option next…

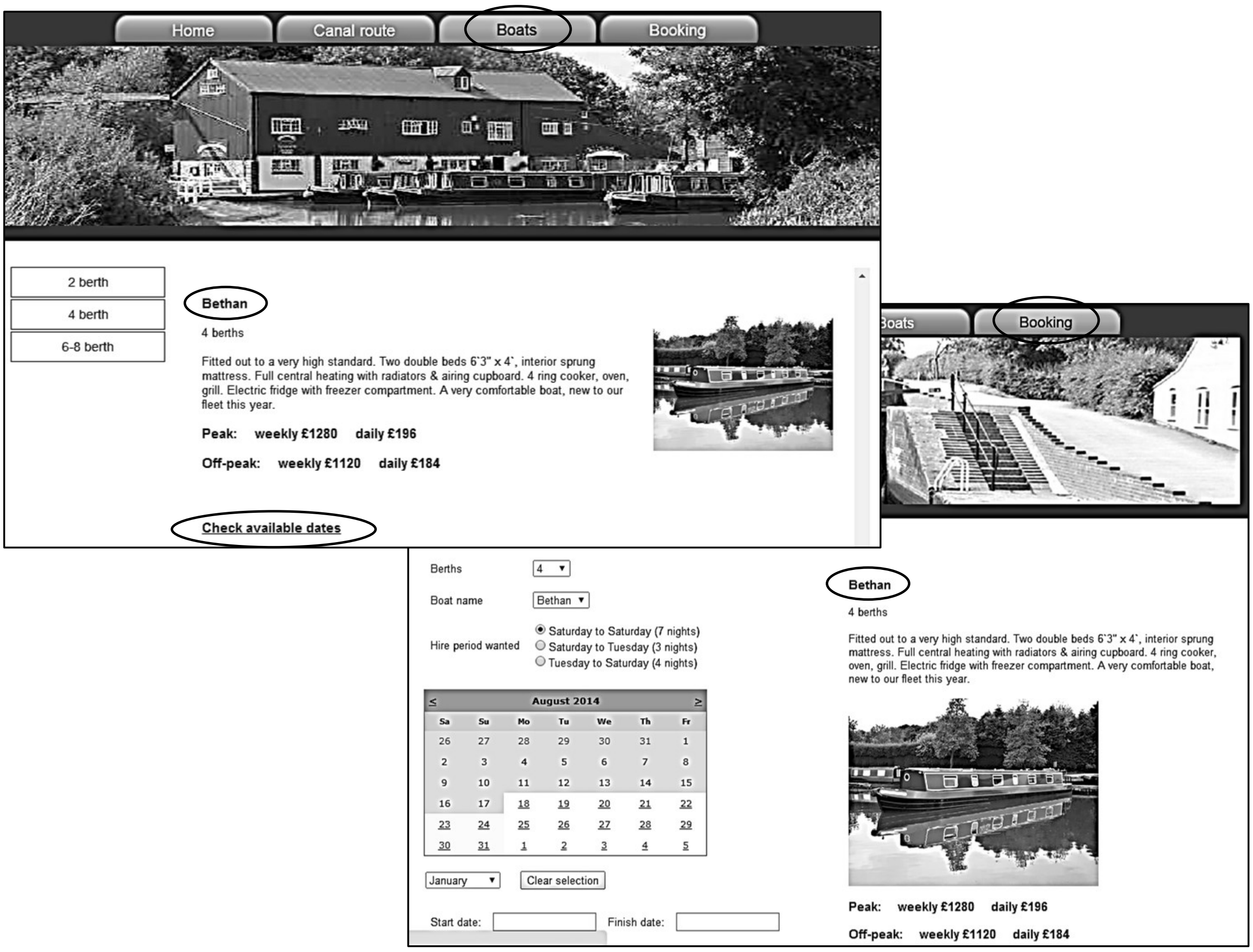
Open the C# code page for ***booking.aspx*** and add lines to the ***Page_Load( )*** method. These enable the ***Berths*** and ***Boat Name*** drop down lists, so that a boat can be selected. This is the situation if the Booking page is being used independantly.

```
protected void Page_Load(object sender, EventArgs e)
{
    btnAvailability.Visible = true;
    ddlBerths.Enabled = true;
    ddlBoatNames.Enabled = true;

    if (canalBoat.boatCount == 0)
    {
        canalBoat.loadBoats();
    }
```

Go to the end of the ***Page_Load( )*** method. Add a section of code which will check the page URL to see if a canal boat had been previously selected on the Boats web page. If so, the Berths and Boat Name drop down lists are disabled, and details of the selected boat are immediately displayed.

```
        if (ddlBoatNames.Text == "")
        {
            setBoatNames("2");
            Label1.Text =
                  "Please select a boat and click button to show availability...";
        }

        boatIDwanted = Convert.ToInt16(Request.QueryString["boatID"]);
        if (boatIDwanted > 0 && txtStartDate.Text=="")
        {
            btnAvailability.Visible = false;
            ddlBerths.Enabled = false;
            ddlBoatNames.Enabled = false;

            for (int i = 0; i < canalBoat.boatCount; i++)
            {
                if (boatIDwanted == canalBoat.boatObject[i].boatID)
                {
                    boatNameWanted = canalBoat.boatObject[i].boatName;
                    int berthsWanted = canalBoat.boatObject[i].berths;
                    string berths = Convert.ToString(berthsWanted);
                    if (berthsWanted >= 6)
                    {
                        berths = "6-8";
                    }
                    displayBoat(boatNameWanted);
                    ddlBerths.Text = berths;
                    setBoatNames(berths);
                    ddlBoatNames.Text = boatNameWanted;
                }
            }
        }
    }
```

Build and run the web site. Choose the ***Boats*** page and select a canal boat. Click the '***Check available dates***' option. Check that you are taken to the ***Booking*** page, with the correct boat displayed. You should then be able to select a hire period and the cost will be calculated, as before.



Bethan
4 berths

Fitted out to a very high standard. Two double beds 6'3" x 4', interior sprung mattress. Full central heating with radiators & airing cupboard. 4 ring cooker, oven, grill. Electric fridge with freezer compartment. A very comfortable boat, new to our fleet this year.

Peak: weekly £1280 daily £196

Off-peak: weekly £1120 daily £184

Check available dates

Berths 4

Boat name Bethan

Hire period wanted [✓ Saturday to Saturday (7 nights)] Saturday to Tuesday (3 nights) Tuesday to Saturday (4 nights)

August 2014

Sa	Su	Mo	Tu	We	Th	Fr
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

January Clear selection

Start date: Finish date:

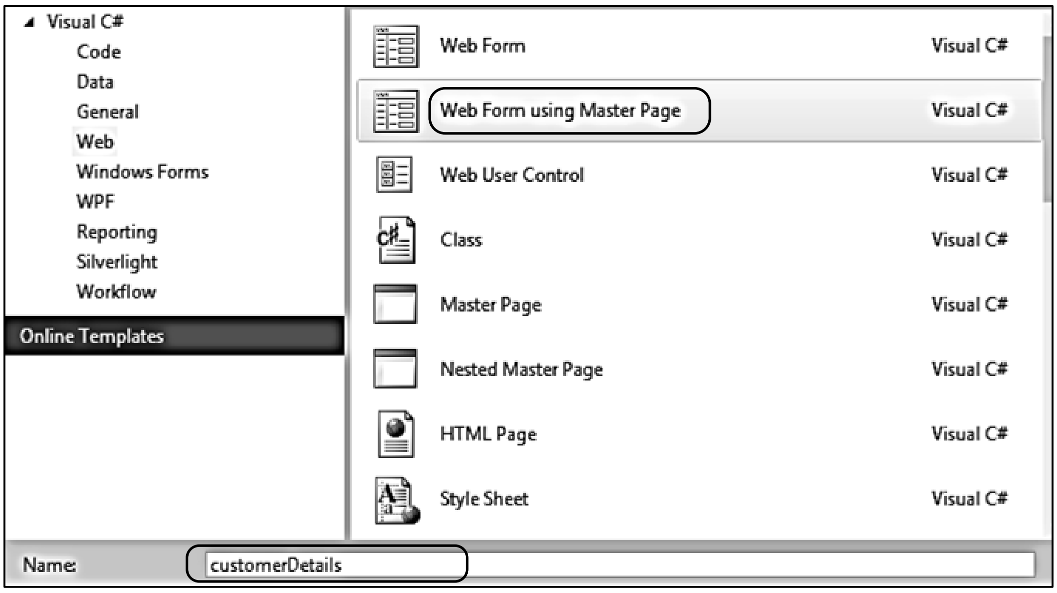
Bethan
4 berths

Fitted out to a very high standard. Two double beds 6'3" x 4', interior sprung mattress. Full central heating with radiators & airing cupboard. 4 ring cooker, oven, grill. Electric fridge with freezer compartment. A very comfortable boat, new to our fleet this year.

Peak: weekly £1280 daily £196

Off-peak: weekly £1120 daily £184

We can now enter contact details for the customer. This will be done on another page. 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 '**customerDetails**'. Select ***Site.Master*** as the master page.



Web Form	Visual C#
Web Form using Master Page	Visual C#
Web User Control	Visual C#
Class	Visual C#
Master Page	Visual C#
Nested Master Page	Visual C#
HTML Page	Visual C#
Style Sheet	Visual C#

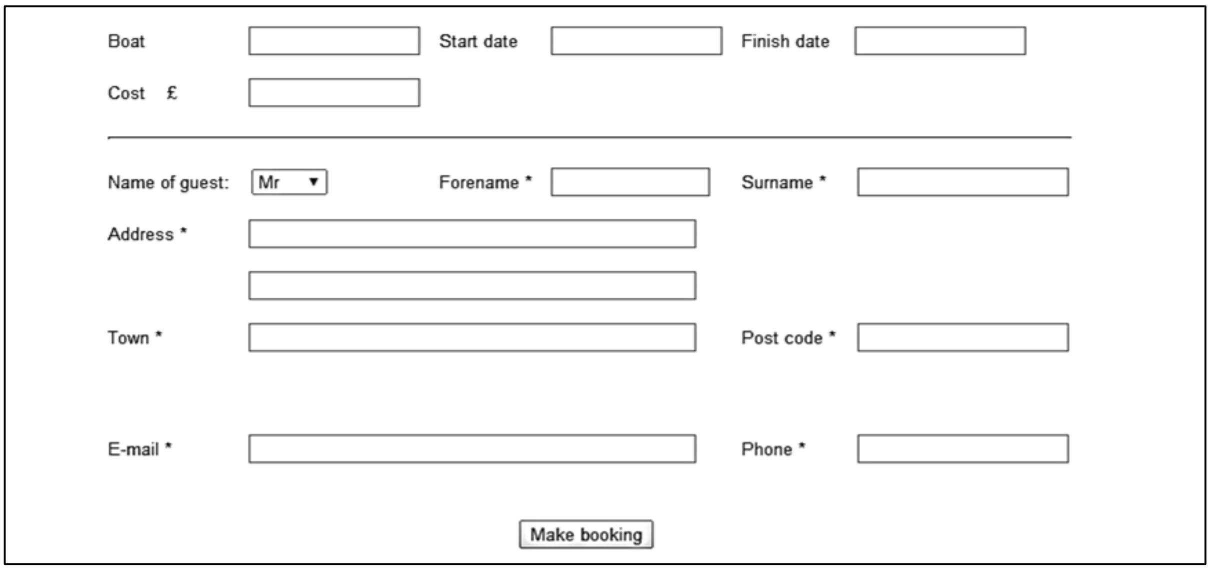
Name: customerDetails

Open the customerDetails.aspx page and add lines of code to set up a data entry form.

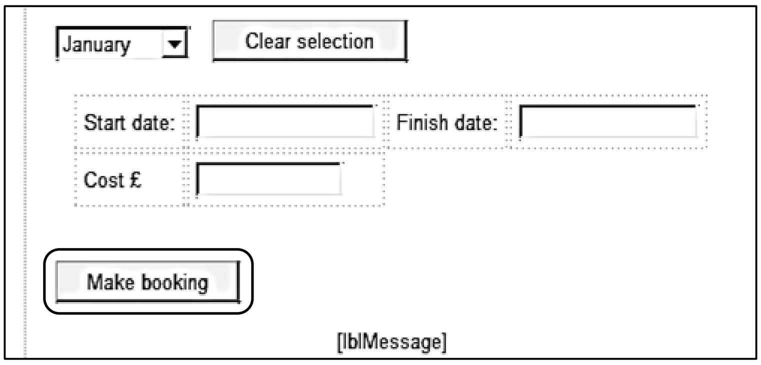
```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <table cellpadding="6">
   <tr>
     <td>Boat</td>
     <td>
       <asp:TextBox ID="txtBoatName" runat="server" Width="120px"></asp:TextBox>
     </td>
     <td>Start date</td>
     <td>
       <asp:TextBox ID="txtStartDate" runat="server" Width="120px"></asp:TextBox>
     </td>
     <td>Finish date</td>
     <td>
       <asp:TextBox ID="txtFinishDate" runat="server" Width="120px"></asp:TextBox>
     </td>
  </tr>
  <tr>
    <td>Cost &nbsp;&nbsp;&nbsp; £</td>
    <td>
      <asp:TextBox ID="txtCost" runat="server" Width="120px"></asp:TextBox>
    </td>
  </tr>
  <tr>
     <td colspan="6"><hr /></td>
  </tr>
  <tr>
     <td>Name of guest:</td>
     <td>
        <asp:DropDownList ID="ddlTitle" runat="server">
           <asp:ListItem Value="Mr"></asp:ListItem>
           <asp:ListItem Value="Mrs"></asp:ListItem>
           <asp:ListItem Value="Miss"></asp:ListItem>
           <asp:ListItem Value="Ms"></asp:ListItem>
        </asp:DropDownList>
     </td>
     <td>Forename *</td>
     <td>
         <asp:TextBox ID="txtForename" runat="server" Width="110px"></asp:TextBox>
     </td>
     <td>Surname *</td>
     <td>
         <asp:TextBox ID="txtSurname" runat="server"></asp:TextBox>
     </td>
   </tr>
   <tr>
     <td>Address *</td>
     <td colspan="3">
        <asp:TextBox ID="txtAddress1" runat="server" Width="320px"></asp:TextBox>
     </td>
   </tr>
```

```
      <tr>
        <td></td>
        <td colspan="3">
         <asp:TextBox ID="txtAddress2" runat="server" Width="320px"></asp:TextBox>
        </td>
      </tr>
      <tr>
        <td>Town *</td>
        <td colspan="3">
           <asp:TextBox ID="txtTown" runat="server" Width="320px"></asp:TextBox>
        </td>
        <td>Post code *</td>
        <td>
           <asp:TextBox ID="txtPostcode" runat="server"></asp:TextBox>
        </td>
      </tr>
      <tr><td><br /><br /></td></tr>
      <tr>
        <td>E-mail *</td>
        <td colspan="3">
           <asp:TextBox ID="txtEmail" runat="server" Width="320px"></asp:TextBox>
        </td>
        <td>Phone *</td>
        <td>
           <asp:TextBox ID="txtPhone" runat="server"></asp:TextBox>
        </td>
     </tr>
   </table>
   <br /><br />
   <asp:Button ID="btnBook" runat="server" Text="Make booking" />
   <br /><br /><br /><br />
   <h3>
     <asp:Label ID="Label1" runat="server"></asp:Label>
   </h3>
   <br />
 </asp:Content>
```

Build and run the ***customerDetails*** web page. Check that the data entry form is displayed correctly.



Close the browser and stop debugging. We will first import the boat name, holiday dates and cost from the ***Booking*** page. Go to the ***booking.aspx*** page and change to the Design view. Double click the '***Make booking***' button to create a ***button_click( )*** method.



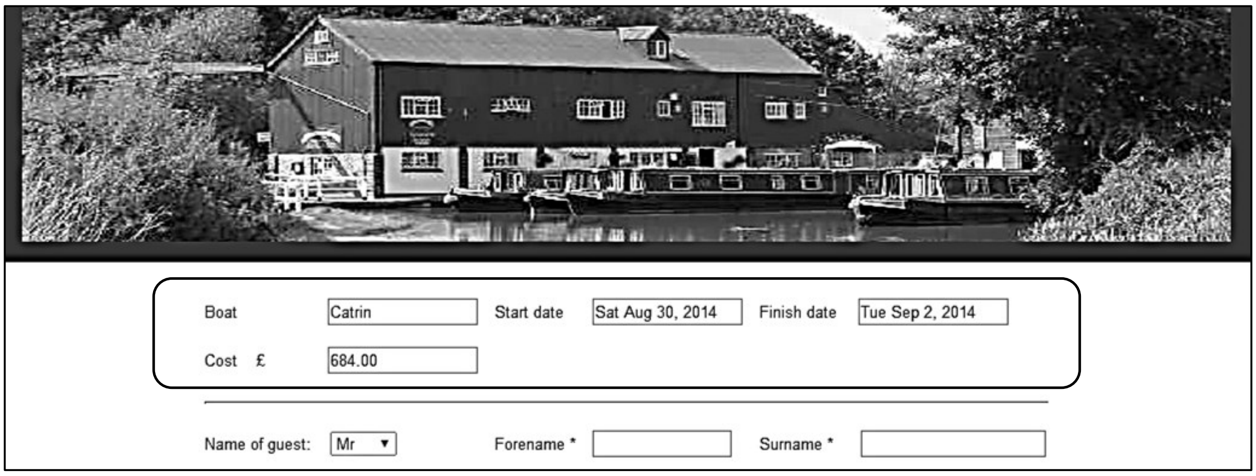
Add a line to the ***btnBook_Click( )*** method which will load the ***customer details*** page, taking with it the boat name, hire dates and cost.

```
  protected void btnBook_Click(object sender, EventArgs e)
  {
     Response.Redirect("customerDetails.aspx?boatName=" + boatNameWanted +
            "&startDate=" + txtStartDate.Text + "&finishDate=" +
            txtFinishDate.Text + "&cost=" + txtCost.Text);
  }
```

Go now to the C# page for customerDetails.aspx. Add variables at the start of the program, and insert lines of code into the ***Page_Load( )*** method.

```
public partial class customerDetails : System.Web.UI.Page
{
   string boatName;
   string startDate;
   string finishDate;
   string cost;

   protected void Page_Load(object sender, EventArgs e)
   {
       boatName = Convert.ToString(Request.QueryString["boatName"]);
       startDate = Convert.ToString(Request.QueryString["startDate"]);
       finishDate = Convert.ToString(Request.QueryString["finishDate"]);
       cost = Convert.ToString(Request.QueryString["cost"]);
       txtBoatName.Text = boatName;
       txtStartDate.Text = startDate;
       txtFinishDate.Text = finishDate;
       txtCost.Text = cost;
   }
```

Build and run the ***booking.aspx*** web page. Select a canal boat and hire dates, then click the '***Make booking***' button. The ***customer details*** page should load. Check that the booking which you entered is displayed correctly.

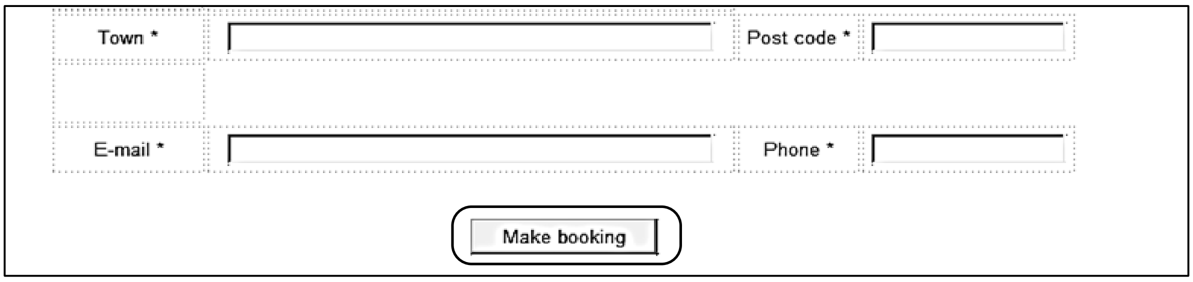


Boat: Catrin
Start date: Sat Aug 30, 2014
Finish date: Tue Sep 2, 2014
Cost £: 684.00
Name of guest: Mr
Forename *
Surname *

Close the browser and stop debugging. We are now ready to save customer details into the database. Open the ***customerBookings.cs*** class file and add a ***makebooking( )*** method

```
    public string phone { get; set; }
    public string dateReceived { get; set; }

    public static void makebooking(string boatName, string startDate,
           string finishDate, string cost, string customerTitle, string forename,
           string surname, string address1, string address2, string town,
           string postcode, string email, string phone)
    {
       string dateReceived = DateTime.Now.ToString("dd/MM/yyyy");
       SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
             AttachDbFilename=" + databaseLocation + "Integrated Security=True;
             Connect Timeout=30; User Instance=True");
       try
       {
          cnTB.Open();
          SqlCommand cmBooking = new SqlCommand();
          cmBooking.Connection = cnTB;
          cmBooking.CommandType = CommandType.Text;
          cmBooking.CommandText = "INSERT INTO bookings(boatName, startDate,
               finishDate, cost,customerTitle, forename, surname, address1,
               address2,town,postcode,email,phone,dateReceived) VALUES ('" +
               boatName + "','" + startDate + "','" + finishDate + "','" + cost +
               "','" + customerTitle + "','" + forename + "','" + surname + "','" +
               address1 + "','" + address2 + "','" + town + "','" + postcode +
               "','" + email + "','" + phone + "','" + dateReceived + "')";
           cmBooking.ExecuteNonQuery();
           cnTB.Close();
       }
       catch
       {
       }
   }
```

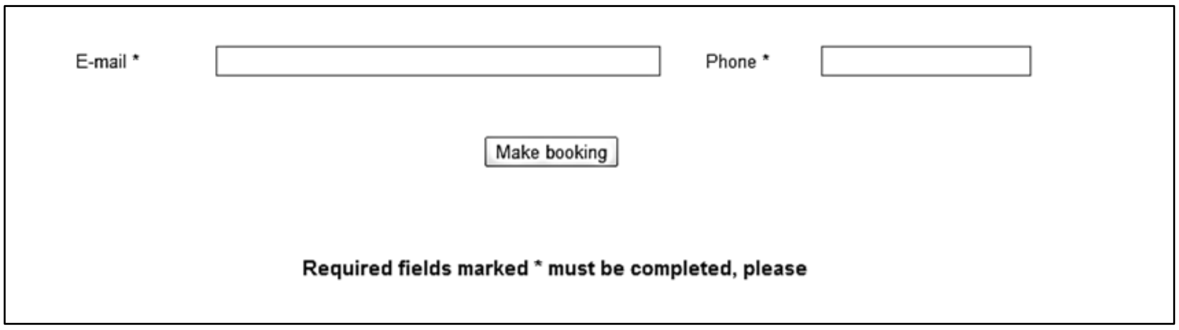
Return to the ***customerDetails.aspx*** page and change to Design view. Double click the '***Make booking***' button to create a button_click( ) method.



A number of the data fields, maked with a * symbol on the page, require essential information. We will give an error message to the user if any of these fields are left blank. Insert lines of code into the ***btnBook_Click( )*** method to do this. Also add a ***checkEntries( )*** method. This works by checking the lengths of the text strings entered, and identifies a length of zero as a blank entry.

```
protected void btnBook_Click(object sender, EventArgs e)
{
   bool completed = checkEntries();
   if (completed == true)
   {
     Label1.Text = "Thank you for your booking. Staff of Canal Boat Holidays will
         contact you shortly to make further arrangements for your holiday.";
     customerBookings.makebooking(boatName, startDate, finishDate, cost,
         ddlTitle.Text, txtForename.Text, txtSurname.Text, txtAddress1.Text,
         txtAddress2.Text, txtTown.Text, txtPostcode.Text, txtEmail.Text,
         txtPhone.Text);
     customerBookings.loadBookings();
   }
   else
   {
     Label1.Text = "Required fields marked * must be completed, please";
   }
}

protected bool checkEntries()
{
   bool completed = true;
   if (txtForename.Text.Length == 0) completed = false;
   if (txtSurname.Text.Length == 0) completed = false;
   if (txtAddress1.Text.Length == 0) completed = false;
   if (txtTown.Text.Length == 0) completed = false;
   if (txtPostcode.Text.Length == 0) completed = false;
   if (txtEmail.Text.Length == 0) completed = false;
   if (txtPhone.Text.Length == 0) completed = false;
   return completed;
}
```

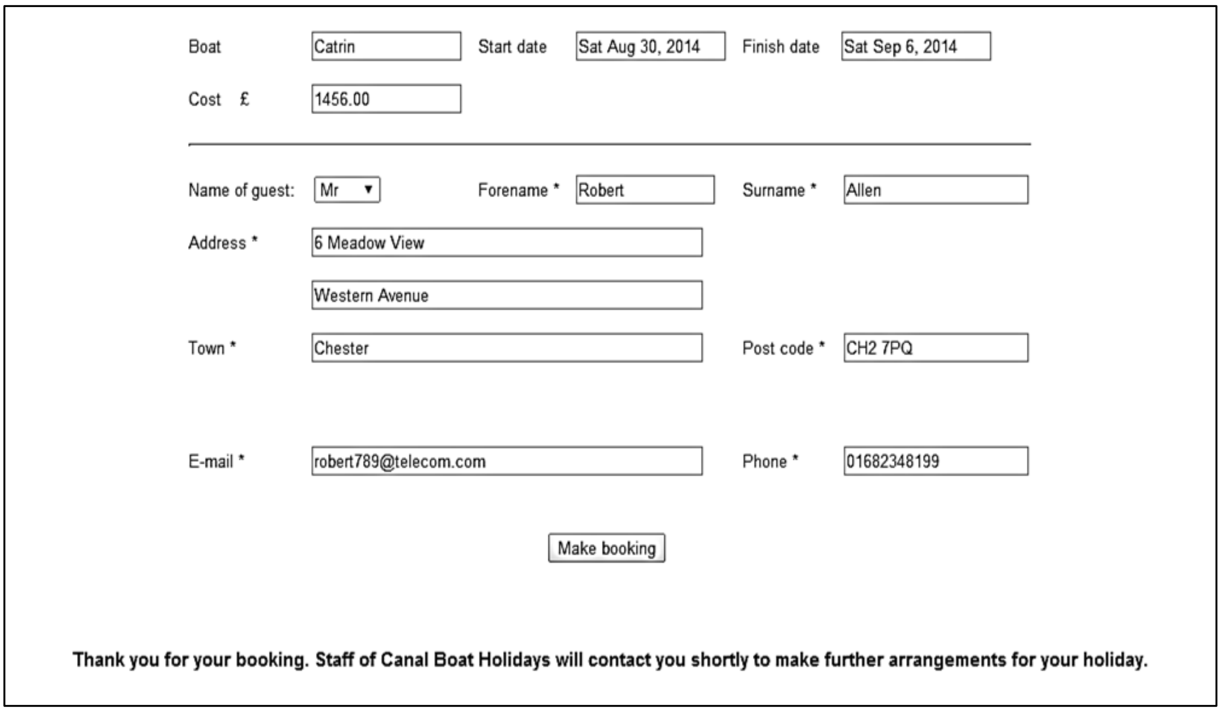
Build and run the ***customerDetails*** web page. Leave some of the required fields blank, then click the '***Make booking***' button. Check that the error message appears.



E-mail *		Phone *	

Required fields marked * must be completed, please

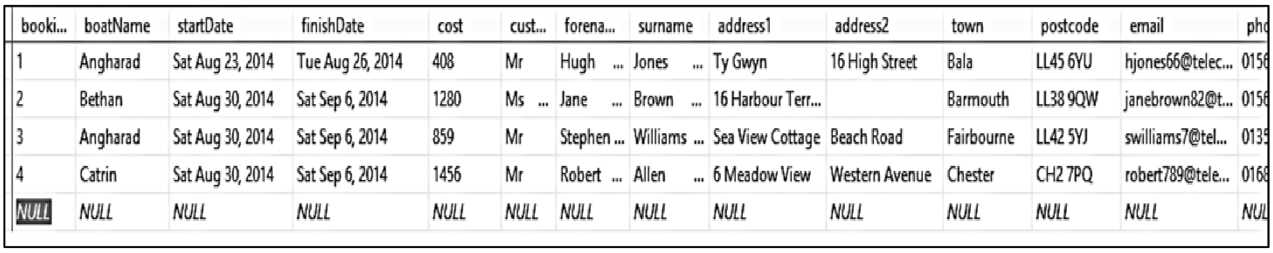
Enter a full set of data for a booking, then click the '***Make booking***' button again. A booking acknowledgement message should be displayed. We will assume that ***Canal Boat Holidays*** will contact the customer to arrange payment, though you could of course add a further web page to handle payment by credit or debit card.



Boat	Catrin	Start date	Sat Aug 30, 2014	Finish date	Sat Sep 6, 2014
Cost £	1456.00				
Name of guest:	Mr	Forename *	Robert	Surname *	Allen
Address *	6 Meadow View				
	Western Avenue				
Town *	Chester			Post code *	CH2 7PQ
E-mail *	robert789@telecom.com			Phone *	01682348199

Thank you for your booking. Staff of Canal Boat Holidays will contact you shortly to make further arrangements for your holiday.

Close the browser and stop debugging. Go to the Server Explorer window and click the ***Refresh*** button. Click right on the ***bookings*** table icon and select '***Show Table Data***'. Check that the booking appears correctly in the table. If all is well, add further bookings for different boats and holiday dates.



booki...	boatName	startDate	finishDate	cost	cust...	forena...	surname	address1	address2	town	postcode	email	ph
1	Angharad	Sat Aug 23, 2014	Tue Aug 26, 2014	408	Mr	Hugh ...	Jones ...	Ty Gwyn	16 High Street	Bala	LL45 6YU	hjones66@telec...	015
2	Bethan	Sat Aug 30, 2014	Sat Sep 6, 2014	1280	Ms ...	Jane ...	Brown ...	16 Harbour Terr...		Barmouth	LL38 9QW	janebrown82@t...	015
3	Angharad	Sat Aug 30, 2014	Sat Sep 6, 2014	859	Mr	Stephen ...	Williams ...	Sea View Cottage	Beach Road	Fairbourne	LL42 5YJ	swilliams7@tel...	013
4	Catrin	Sat Aug 30, 2014	Sat Sep 6, 2014	1456	Mr	Robert ...	Allen ...	6 Meadow View	Western Avenue	Chester	CH2 7PQ	robert789@tele...	016
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU

One final task we must now carry out on the ***Bookings*** page is to show exisiting bookings on the calendar, so that double bookings of boats are avoided. Open the C# page for ***booking.aspx*** and add a ***bookedList*** variable at the start of the class. This list will hold the booked dates for the selected canal boat, so that these days can be marked on the calendar.

```
public partial class booking : System.Web.UI.Page
{
    public static string boatNameWanted;
    public static int boatIDwanted;

    public static List<DateTime> bookedList = new List<DateTime>();
```

Go now to the ***Page_Load( )*** method and add code to load the bookings from the database if these have not already been loaded.

```
protected void Page_Load(object sender, EventArgs e)
{
    btnAvailability.Visible = true;
    ddlBerths.Enabled = true;
    ddlBoatNames.Enabled = true;

    if (customerBookings.bookingCount == 0)
    {
        customerBookings.loadBookings();
    }

    if (canalBoat.boatCount == 0)
    {
        canalBoat.loadBoats();
    }
```

Near the end of the ***Page_Load( )*** method, add a line of code to call the ***displayBookings( )*** method if the user arrives at ***Bookings*** after previously selecting a canal boat from the ***Boats*** page. The method takes the name of the required boat as a parameter so that the correct set of exisiting bookings can be displayed on the claendar.

```
            displayBoat(boatNameWanted);
            ddlBerths.Text = berths;
            setBoatNames(berths);
            ddlBoatNames.Text = boatNameWanted;
            displayBookings(boatNameWanted);
        }
    }
```

Insert the ***displayBookings( )*** method immediately after the ***Page_Load( )*** method.

```
protected void displayBookings(string boatWanted)
{
    string boat;
    string startDate;
    string finishDate;
    bookedList.Clear();
    for (int i = 0; i < customerBookings.bookingCount; i++)
    {
        boat = customerBookings.bookingObject[i].boatName;
        if (boat == boatNameWanted)
        {
            startDate = customerBookings.bookingObject[i].startDate;
            finishDate = customerBookings.bookingObject[i].finishDate;
            DateTime today = Convert.ToDateTime(startDate);
            DateTime lastDay = Convert.ToDateTime(finishDate);
            while (today < lastDay)
            {
                bookedList.Add(today);
                today = today.AddDays(1);
            }
        }
    }
}
```

We also need to link the ***displayBookings( )*** method to the '***Show availability***' button, so the calendar display can be updated if a different boat is selected. Add a line of code to do this.

```
protected void btnAvailability_Click(object sender, EventArgs e)
{
    boatNameWanted = Convert.ToString(ddlBoatNames.SelectedItem);
    displayBoat(boatNameWanted);
    displayBookings(boatNameWanted);
}
```

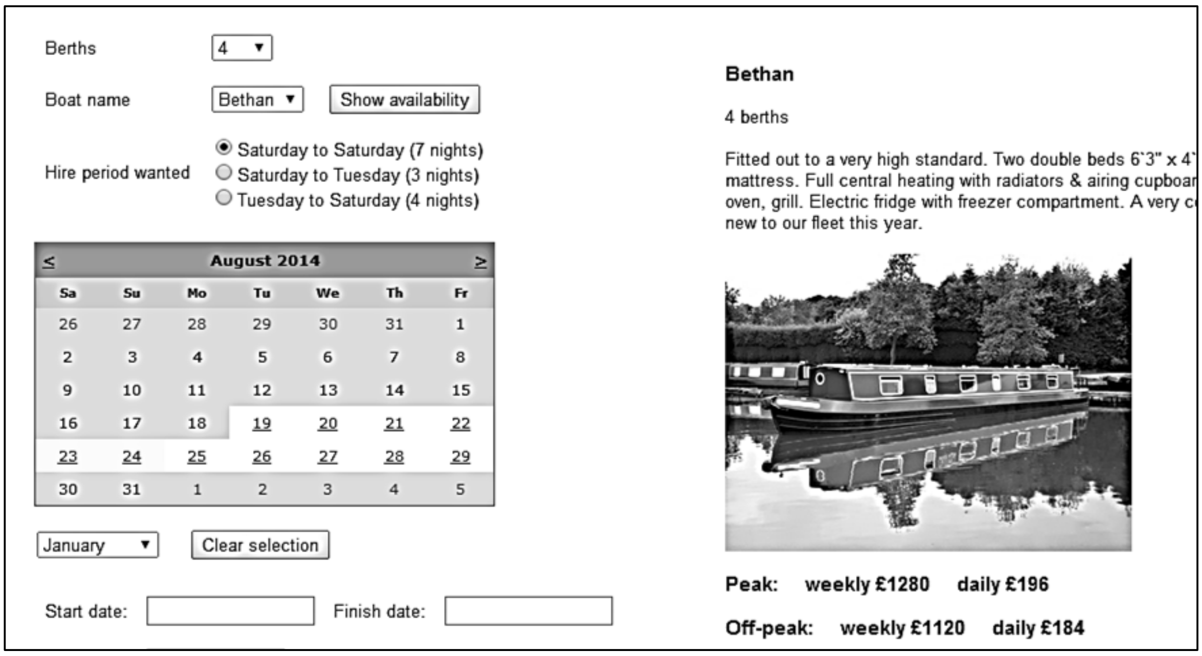
Go now to the ***Calendar_DayRender( )*** method. This currently sets all dates up to and including today's date as unavailable, and shades these in grey. We can use a similar method to obtain the booked dates from bookedList and make these appear as unavailable. Add lines of code to the method as shown below.

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    DateTime currentDate = Calendar1.TodaysDate;

    if (e.Day.Date <= currentDate)
    {
        e.Day.IsSelectable = false;
        e.Cell.BackColor = Color.Gainsboro;
    }

    foreach (DateTime dt in bookedList)
    {
        if (e.Day.Date == dt.Date)
        {
            e.Day.IsSelectable = false;
            e.Cell.BackColor = Color.Gainsboro;
        }
    }
}
```

Finally, we should clear the list of bookings for the current boat if the user selects a different group of boats from the ***Berths*** drop down list. Add a line of code to ***ddlBerths_SelectedIndexChanged( )***.

```
protected void ddlBerths_SelectedIndexChanged(object sender, EventArgs e)
{
    bookedList.Clear();
    setBoatNames(ddlBerths.Text);
    Label1.Text = "Please select a boat and click button to show availability...";
}
```

Build and run the web site. Select boats, either directly using the Bookings page or by first visiting the Boats page. Check that any existing bookings for that boat are indicated on the calendar and these dates are disabled.


Berths: 4
Boat name: Bethan

Hire period wanted
(●) Saturday to Saturday (7 nights)
( ) Saturday to Tuesday (3 nights)
( ) Tuesday to Saturday (4 nights)

< August 2014 >

Sa	Su	Mo	Tu	We	Th	Fr
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

January

Start date: Finish date:

Bethan

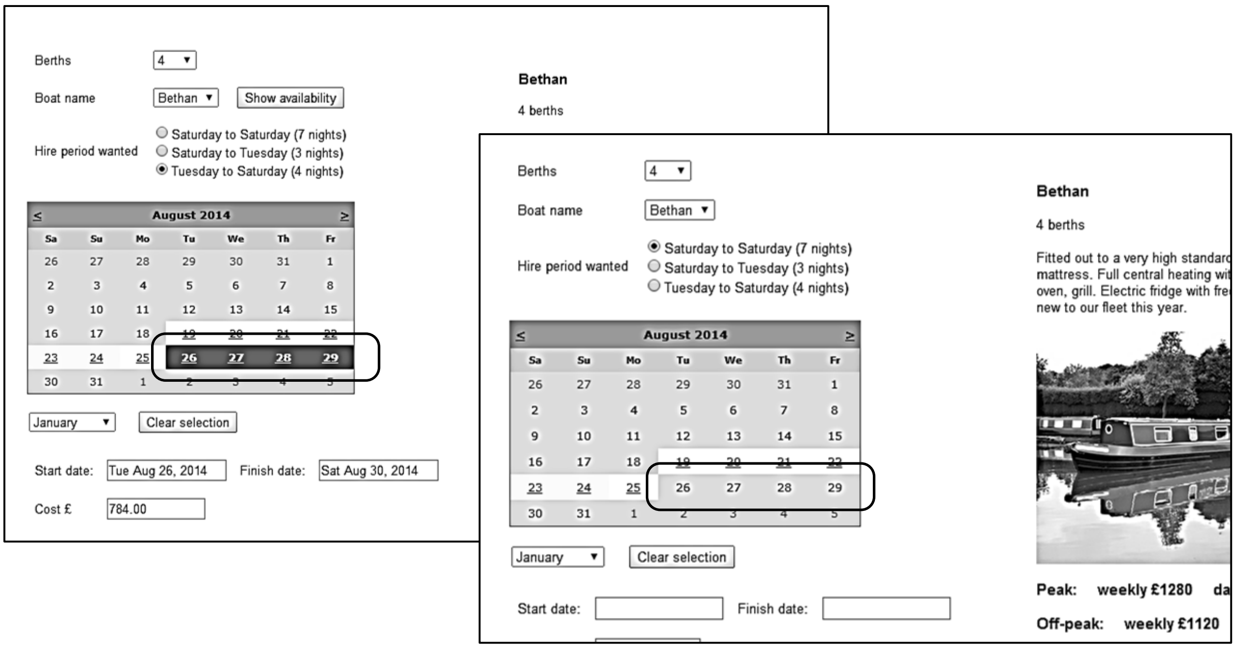
4 berths

Fitted out to a very high standard. Two double beds 6`3" x 4`
mattress. Full central heating with radiators & airing cupboar
oven, grill. Electric fridge with freezer compartment. A very c
new to our fleet this year.

Peak: weekly £1280 daily £196

Off-peak: weekly £1120 daily £184

Add further bookings, and check that these periods appear correctly on the calendar as unavailable for further booking.



Berths 4

Boat name Bethan

Hire period wanted
- ○ Saturday to Saturday (7 nights)
- ○ Saturday to Tuesday (3 nights)
- ● Tuesday to Saturday (4 nights)

≤			August 2014			≥
Sa	Su	Mo	Tu	We	Th	Fr
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	~~19~~	~~20~~	~~21~~	~~22~~
23	24	25	26	27	28	29
30	31	1	2	3	4	5

January

Start date: Tue Aug 26, 2014 Finish date: Sat Aug 30, 2014

Cost £ 784.00

Bethan

4 berths

Berths 4

Boat name Bethan

Hire period wanted
- ● Saturday to Saturday (7 nights)
- ○ Saturday to Tuesday (3 nights)
- ○ Tuesday to Saturday (4 nights)

≤			August 2014			≥
Sa	Su	Mo	Tu	We	Th	Fr
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	~~19~~	~~20~~	~~21~~	~~22~~
23	24	25	26	27	28	29
30	31	1	2	3	4	5

January

Start date: Finish date:

Bethan

4 berths

Fitted out to a very high standar
mattress. Full central heating wi
oven, grill. Electric fridge with fre
new to our fleet this year.

Peak: weekly £1280 da

Off-peak: weekly £1120

Close the web browser and stop debugging. This completes the public web site. We will now add a staff option to display the bookings received. Open the staffBooking.aspx page and add lines of code to the '***Content2***' section.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="bookingsContent">
    <table cellpadding="6">
      <tr>
        <td>Boat name</td>
        <td>
          <asp:DropDownList ID="ddlBoatNames" runat="server"></asp:DropDownList>
        </td>
        <td>
          <asp:Button ID="btnBookings" runat="server" Text="Show bookings" />
        </td>
      </tr>
    </table>
    <br />
    <div id="bookingTable">
       <asp:Label ID="Label1" runat="server" ></asp:Label>
    </div>
  </div>
</asp:Content>
```

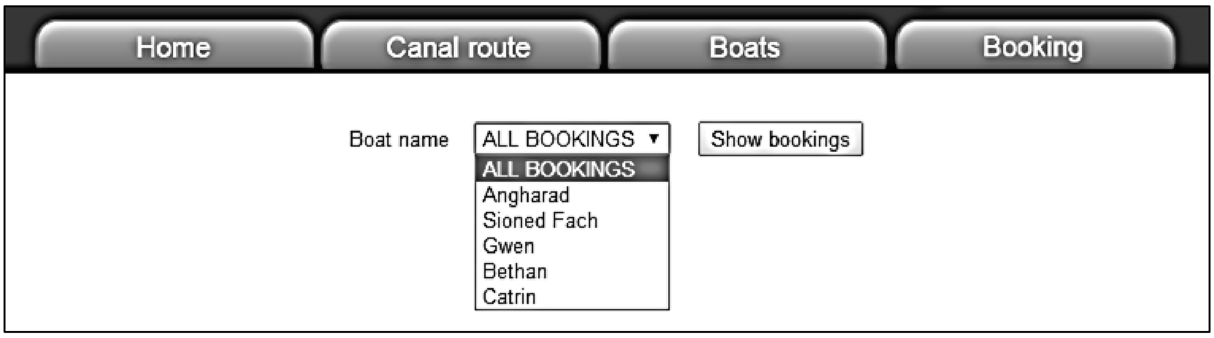
Open the ***Style Sheet*** and add formatting code for the ***bookingsContent*** and ***bookingTable*** divisions.

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

 #bookingTable
{
    height:480px;
    overflow-y: scroll;
    overflow-x: hidden;
}
```

Move now to the C# page for ***staffBooking.aspx*** and add lines of code to the ***Page_Load( )*** method. This program code will load the canal boat and customer bookings records, then add the boat names to the drop down list.

```
protected void Page_Load(object sender, EventArgs e)
{
      if (canalBoat.boatCount == 0)
      {
          canalBoat.loadBoats();
      }
      if (customerBookings.bookingCount == 0)
      {
          customerBookings.loadBookings();
      }
      if (ddlBoatNames.Text == "")
      {
          ddlBoatNames.Items.Clear();
          ddlBoatNames.Items.Add("ALL BOOKINGS");
          for (int i = 0; i < canalBoat.boatCount; i++)
          {
              ddlBoatNames.Items.Add(canalBoat.boatObject[i].boatName);
          }
      }
}
```

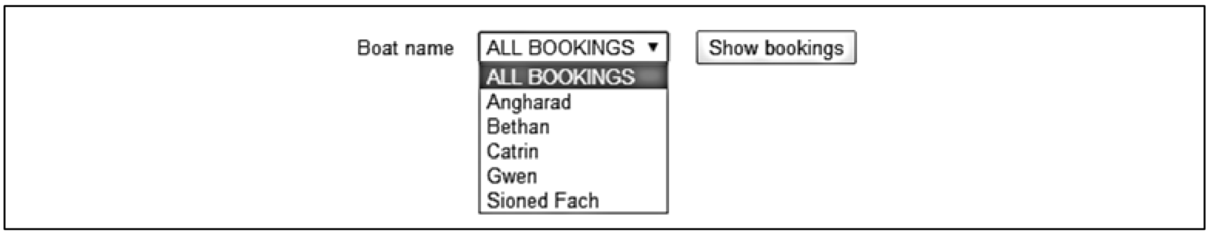
Build and run the staffBooking web page. Check that the boat names are displayed in the drop down list. The boat names are not in sorted order, but it would be more convenient for the user if they were displayed alphabetically. This can easily be arranged…



Close the browser and stop debugging. Go to the ***canalBoat.cs*** class file and find the ***loadBoats( )*** method. Locate the line ***cmBoat.CommandText = "SELECT * FROM boats"*** and modify this by adding a sort command. The boat records will now be loaded in ascending alphabetical order of boat name.

```
try
{
    cnTB.Open();
    SqlCommand cmBoat = new SqlCommand();
    cmBoat.Connection = cnTB;
    cmBoat.CommandType = CommandType.Text;
    cmBoat.CommandText = "SELECT * FROM boats ORDER BY boatName ASC";
    SqlDataAdapter daBoat = new SqlDataAdapter(cmBoat);
```

Build and run the ***staffBooking*** web page again. The boat names should now appear alphabetically in the drop down list.

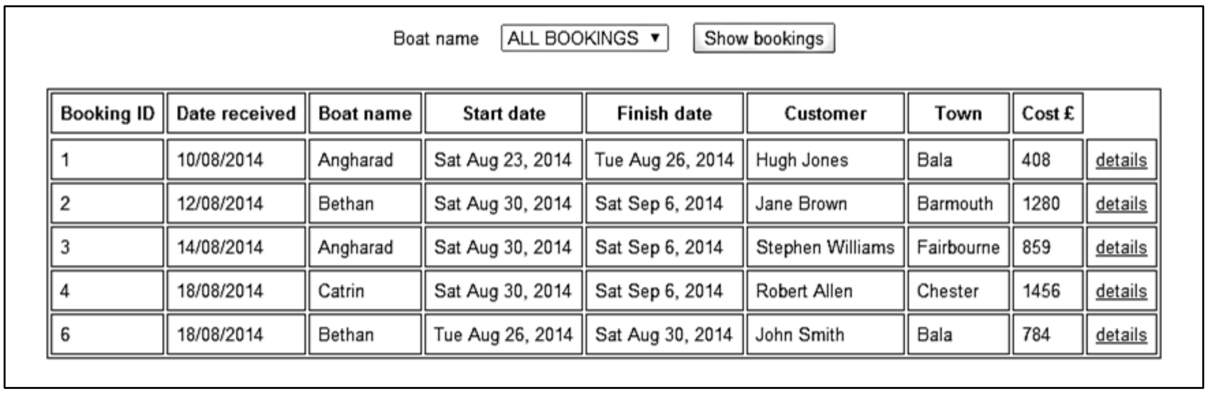


Close the browser and stop debugging. Go to the ***staffBooking.aspx*** page and select Design view. Double click the '***Show bookings***' button to create a button_click( ) method. Add lines of code to produce a table of bookings, as shown below.

```
protected void btnBookings_Click(object sender, EventArgs e)
{
  string s = "";
  string boatname;
  s += "<table border=1 cellpadding=6>";
  s += "<tr>";
  s += "<th>Booking ID</th>";
  s += "<th>Date received</th>";
  s += "<th>Boat name</th>";
  s += "<th>Start date </th>";
  s += "<th>Finish date</th>";
  s += "<th>Customer</th>";
  s += "<th>Town</th>";
  s += "<th>Cost £</th>";
  s += "<tr>";
  string boatSelected = ddlBoatNames.Text;
  bool allBookings = false;
  if (boatSelected == "ALL BOOKINGS")
  {
      allBookings = true;
  }
  for (int i = 0; i < customerBookings.bookingCount; i++)
  {
     boatname = customerBookings.bookingObject[i].boatName;
     if (allBookings == true || boatSelected == boatname)
     {
        s += "<tr>";
        s += "<td>" + customerBookings.bookingObject[i].bookingID + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].dateReceived + "</td>";
        s += "<td>" + boatname + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].startDate + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].finishDate + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].forename + "" +
                          customerBookings.bookingObject[i].surname + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].town + "</td>";
        s += "<td>" + customerBookings.bookingObject[i].cost + "</td>";
        s += "<td><a href=staffBookingDetails.aspx?bookingID=" +
             customerBookings.bookingObject[i].bookingID + ">details</a></td>";
        s += "</tr>";
     }
  }
  s += "</table>";
  Label1.Text = s;
}
```

Notice that we have included an 'ALL BOATS' option, when all booking records are displayed. In other cases, the program first checks whether each record is for the selected boat.

Build and run the staffBooking web page. Check that records are displayed correctly for selected boats and for all boats.



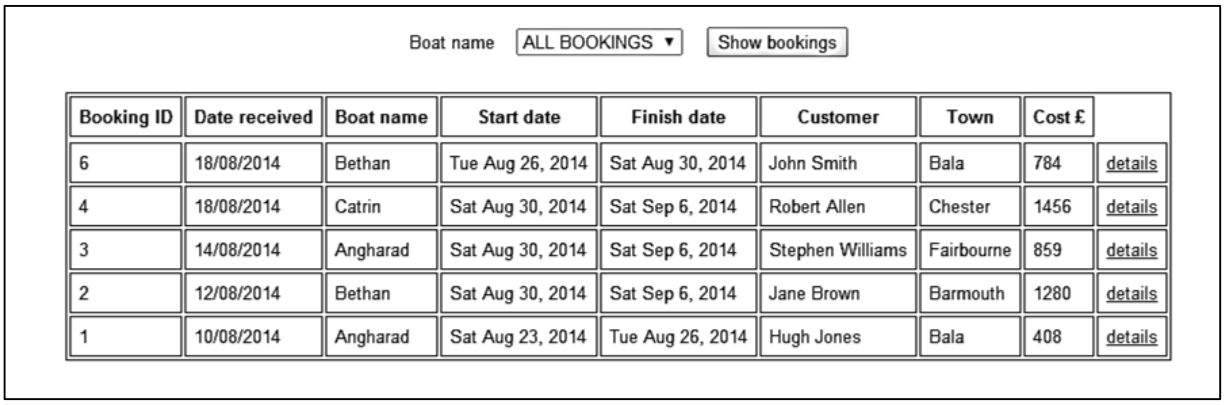
Booking ID	Date received	Boat name	Start date	Finish date	Customer	Town	Cost £	
1	10/08/2014	Angharad	Sat Aug 23, 2014	Tue Aug 26, 2014	Hugh Jones	Bala	408	details
2	12/08/2014	Bethan	Sat Aug 30, 2014	Sat Sep 6, 2014	Jane Brown	Barmouth	1280	details
3	14/08/2014	Angharad	Sat Aug 30, 2014	Sat Sep 6, 2014	Stephen Williams	Fairbourne	859	details
4	18/08/2014	Catrin	Sat Aug 30, 2014	Sat Sep 6, 2014	Robert Allen	Chester	1456	details
6	18/08/2014	Bethan	Tue Aug 26, 2014	Sat Aug 30, 2014	John Smith	Bala	784	details

When operating the business, staff will probably wish to see the newly arrived bookings first, so the bookings should be organised in descending order of date. This can be carried out in a similar way to the sorting of the boat names.

Close the browser and stop debugging. Go to the ***customerBookings.cs*** class file and find the ***loadBookings( )*** method. Locate the line ***cmBooking.CommandText = "SELECT * FROM bookings"*** and modify this by adding a sort command. The booking records will now be loaded in descending order of bookingID, with the most recently received booking first.

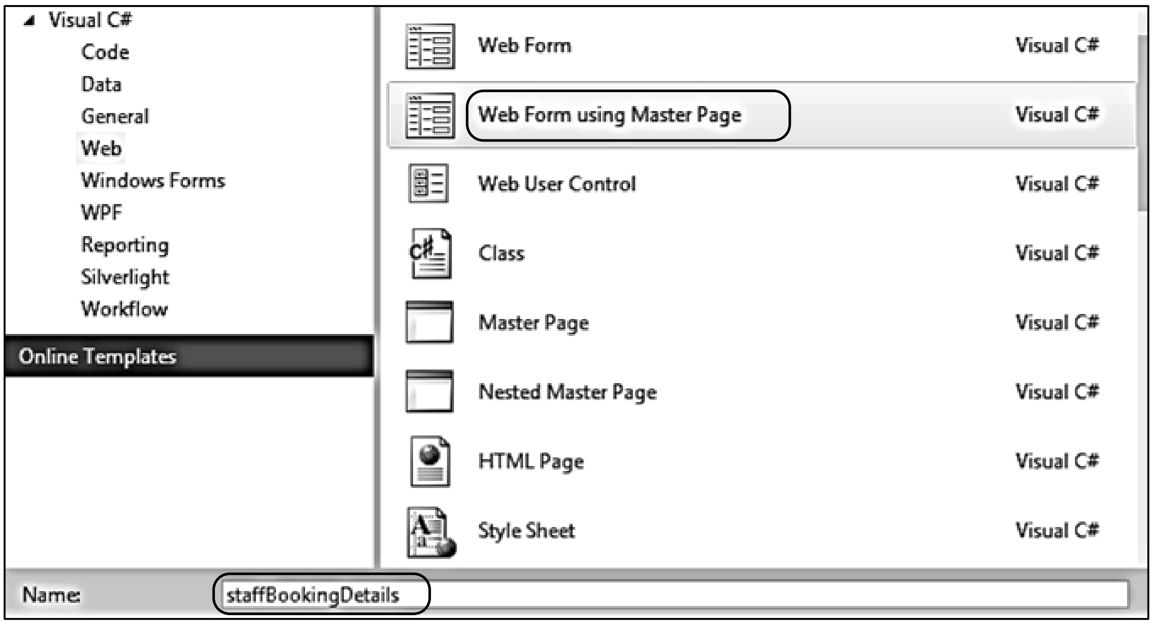
```
try
{
    cnTB.Open();
    SqlCommand cmBooking = new SqlCommand();
    cmBooking.Connection = cnTB;
    cmBooking.CommandType = CommandType.Text;
    cmBooking.CommandText = "SELECT * FROM bookings ORDER BY bookingID DESC";
    SqlDataAdapter daBooking = new SqlDataAdapter(cmBooking);
```

Build and run the ***staffBooking*** web page and check that the bookings are displayed in the required order. Notice that an option has been placed alongside each booking to view details. We will work on this now…



Booking ID	Date received	Boat name	Start date	Finish date	Customer	Town	Cost £	
6	18/08/2014	Bethan	Tue Aug 26, 2014	Sat Aug 30, 2014	John Smith	Bala	784	details
4	18/08/2014	Catrin	Sat Aug 30, 2014	Sat Sep 6, 2014	Robert Allen	Chester	1456	details
3	14/08/2014	Angharad	Sat Aug 30, 2014	Sat Sep 6, 2014	Stephen Williams	Fairbourne	859	details
2	12/08/2014	Bethan	Sat Aug 30, 2014	Sat Sep 6, 2014	Jane Brown	Barmouth	1280	details
1	10/08/2014	Angharad	Sat Aug 23, 2014	Tue Aug 26, 2014	Hugh Jones	Bala	408	details

Close the browser and stop debugging. 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 '**staffBookingDetails**'. Select ***staffSite.Master*** as the master page.



Name: staffBookingDetails

Open the ***staffBookingDetails.aspx*** page and add code to the '***Content2***' section.

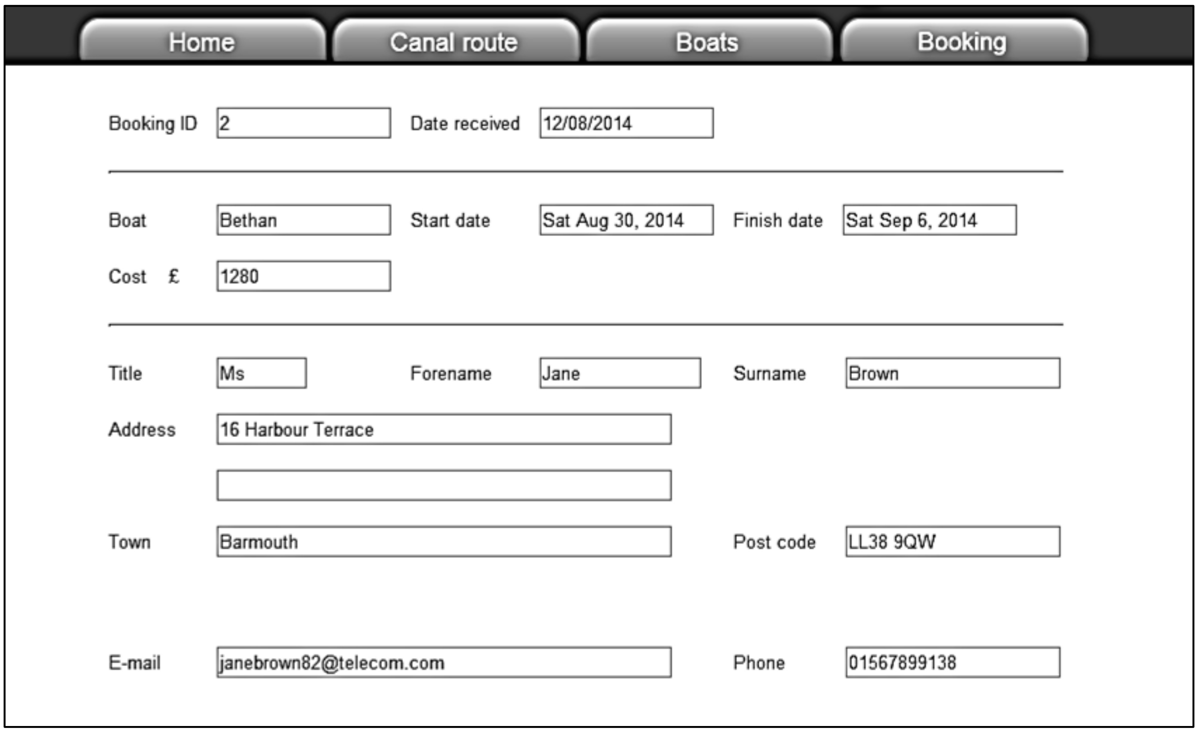
```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

  <div id="bookingsContent">
    <table cellpadding="6">
      <tr>
        <td>Booking ID</td>
        <td>
           <asp:TextBox ID="txtBookingID" runat="server" Width="120px"></asp:TextBox>
        </td>
        <td>Date received</td>
        <td>
          <asp:TextBox ID="txtReceived" runat="server" Width="120px"></asp:TextBox>
        </td>
      </tr>
      <tr>
        <td colspan="6"><hr /></td>
      </tr>
      <tr>
        <td>Boat</td>
        <td>
           <asp:TextBox ID="txtBoatName" runat="server" Width="120px"></asp:TextBox>
        </td>
        <td>Start date</td>
        <td>
           <asp:TextBox ID="txtStartDate" runat="server" Width="120px"></asp:TextBox>
        </td>
        <td>Finish date</td>
         <td>
           <asp:TextBox ID="txtFinishDate" runat="server" Width="120px"></asp:TextBox>
         </td>
       </tr>
       <tr>
         <td>Cost &nbsp;&nbsp;&nbsp; £</td>
         <td>
           <asp:TextBox ID="txtCost" runat="server" Width="120px"></asp:TextBox>
         </td>
       </tr>
       <tr>
         <td colspan="6"><hr /></td>
       </tr>
       <tr>
         <td>Title</td>
         <td>
           <asp:TextBox ID="txtTitle" runat="server" Width="60px"></asp:TextBox>
         </td>
         <td>Forename</td>
         <td>
           <asp:TextBox ID="txtForename" runat="server" Width="111px"></asp:TextBox>
         </td>
         <td>Surname</td>
         <td>
           <asp:TextBox ID="txtSurname" runat="server"></asp:TextBox>
         </td>
       </tr>
       <tr>
         <td>Address</td>
         <td colspan="3">
           <asp:TextBox ID="txtAddress1" runat="server" Width="320px"></asp:TextBox>
         </td>
       </tr>
       <tr>
         <td></td>
         <td colspan="3">
           <asp:TextBox ID="txtAddress2" runat="server" Width="320px"></asp:TextBox>
         </td>
       </tr>
       <tr>
         <td>Town</td>
         <td colspan="3">
            <asp:TextBox ID="txtTown" runat="server" Width="320px"></asp:TextBox>
         </td>
         <td>Post code</td>
         <td>
            <asp:TextBox ID="txtPostcode" runat="server"></asp:TextBox>
         </td>
       </tr>
       <tr>
          <td><br /><br /></td>
       </tr>
       <tr>
```

```
        <td>E-mail</td>
        <td colspan=3>
            <asp:TextBox ID="txtEmail" runat="server" Width="320px"></asp:TextBox>
        </td>
        <td>Phone</td>
        <td><asp:TextBox ID="txtPhone" runat="server"></asp:TextBox></td>
      </tr>
   </table>
 </div>
</asp:Content>
```

Go to the C# code page for ***staffBookingDetails.aspx*** and add a ***bookingIDwanted*** variable. Insert lines of code in the ***Page_Load( )*** method, and add a ***displayBooking( )*** method.

```
public partial class staffBookingDetails : System.Web.UI.Page
 {
   int bookingIDwanted;
    protected void Page_Load(object sender, EventArgs e)
    {
        bookingIDwanted = Convert.ToInt16(Request.QueryString["bookingID"]);
        txtBookingID.Text = Convert.ToString(bookingIDwanted);
        displayBooking(bookingIDwanted);
    }
    protected void displayBooking(int bookingIDwanted)
    {
        int bookingID;
        for (int i = 0; i < customerBookings.bookingCount; i++)
        {
           bookingID = customerBookings.bookingObject[i].bookingID;
           if (bookingID == bookingIDwanted)
           {
            txtReceived.Text = customerBookings.bookingObject[i].dateReceived;
            txtBoatName.Text = customerBookings.bookingObject[i].boatName;
            txtStartDate.Text = customerBookings.bookingObject[i].startDate;
            txtFinishDate.Text = customerBookings.bookingObject[i].finishDate;
            txtCost.Text = Convert.ToString(customerBookings.bookingObject[i].cost);
            txtTitle.Text = customerBookings.bookingObject[i].customerTitle;
            txtForename.Text = customerBookings.bookingObject[i].forename;
            txtSurname.Text = customerBookings.bookingObject[i].surname;
            txtAddress1.Text = customerBookings.bookingObject[i].address1;
            txtAddress2.Text = customerBookings.bookingObject[i].address2;
            txtTown.Text = customerBookings.bookingObject[i].town;
            txtPostcode.Text = customerBookings.bookingObject[i].postcode;
            txtEmail.Text = customerBookings.bookingObject[i].email;
            txtPhone.Text = customerBookings.bookingObject[i].phone;
          }
       }
     }
```

This completes the programming for the booking display. Build and run the ***staffBooking.aspx*** web page, select a canal boat booking then click the **details** option. Check that the full booking information is shown correctly.



Booking ID	2	Date received	12/08/2014		
Boat	Bethan	Start date	Sat Aug 30, 2014	Finish date	Sat Sep 6, 2014
Cost £	1280				
Title	Ms	Forename	Jane	Surname	Brown
Address	16 Harbour Terrace				
Town	Barmouth			Post code	LL38 9QW
E-mail	janebrown82@telecom.com			Phone	01567899138

We will end the example project at this point. For a real booking system, options would also be needed for amending booking details and deleting cancelled bookings. You might like to add these options, using similar techniques to the Hardware Store project in chapter 9.