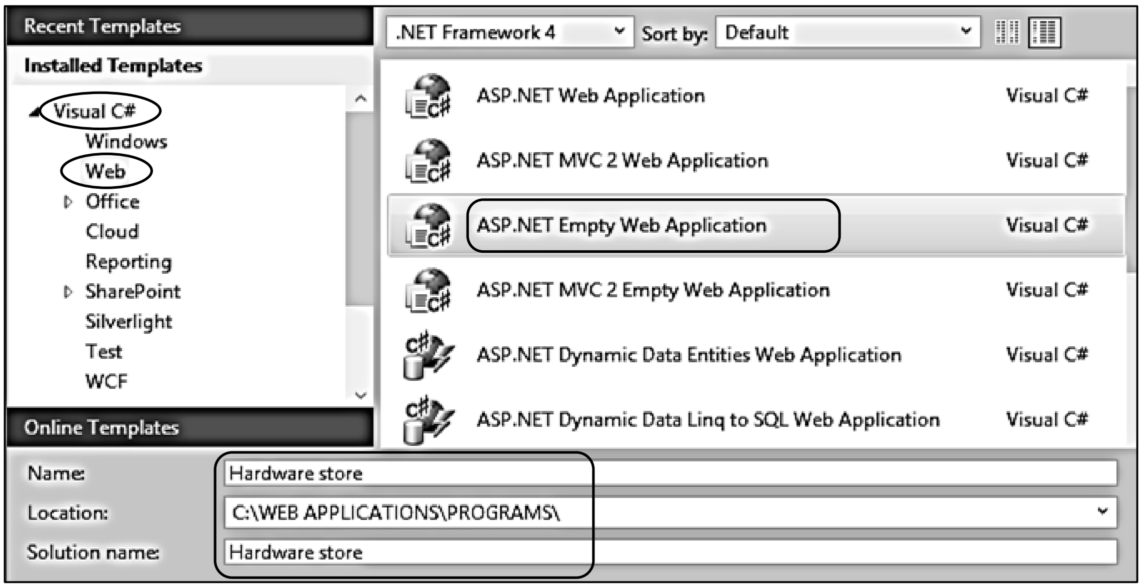
9 Hardware Store

In the final two projects for this book, we will produce larger web applications by making use of the various programming techniques introduced in previous chapters.

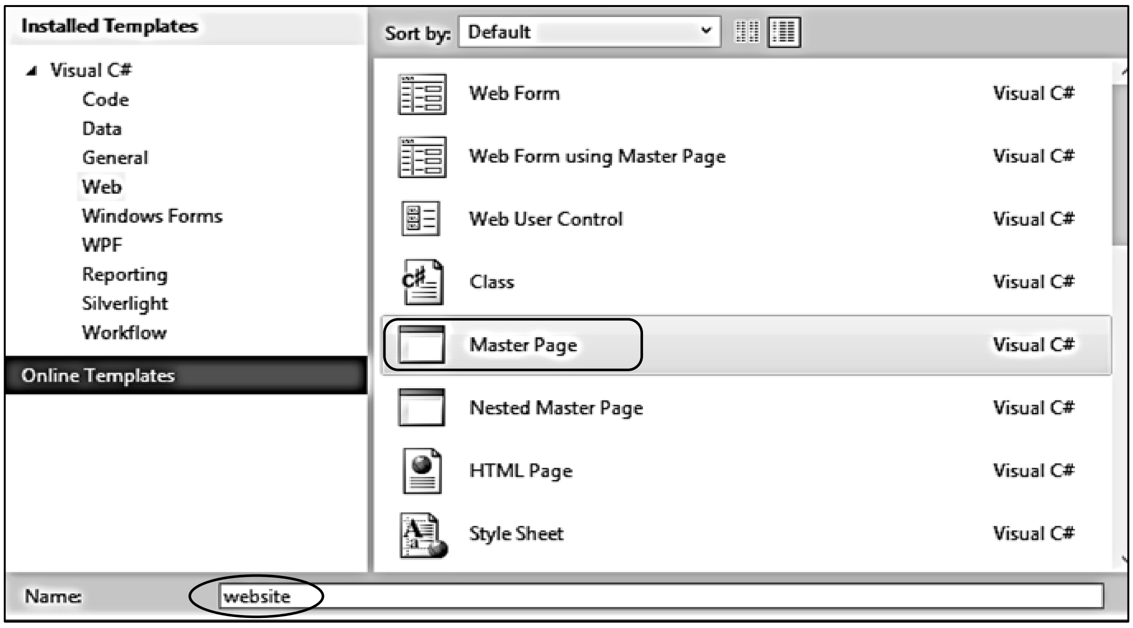
In this section we will set up a web site for a Hardware Store to advertise its products on-line. This application makes use of a menu system similar to the Snowdonia web site in chapter 5, and will display images from a database in a similar way to the Mountin Bike Club project in chapter 6. We will use an object oriented approach for handling product records, similar to the Library Loan system in chapter 8.

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 '***Hardware store***' and select a folder location for the project.


Recent Templates
Installed Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Online Templates
.NET Framework 4
Sort by: Default
ASP.NET Web Application Visual C#
ASP.NET MVC 2 Web Application Visual C#
ASP.NET Empty Web Application Visual C#
ASP.NET MVC 2 Empty Web Application Visual C#
ASP.NET Dynamic Data Entities Web Application Visual C#
ASP.NET Dynamic Data Linq to SQL Web Application Visual C#

Name: Hardware store
Location: C:\WEB APPLICATIONS\PROGRAMS\
Solution name: Hardware store

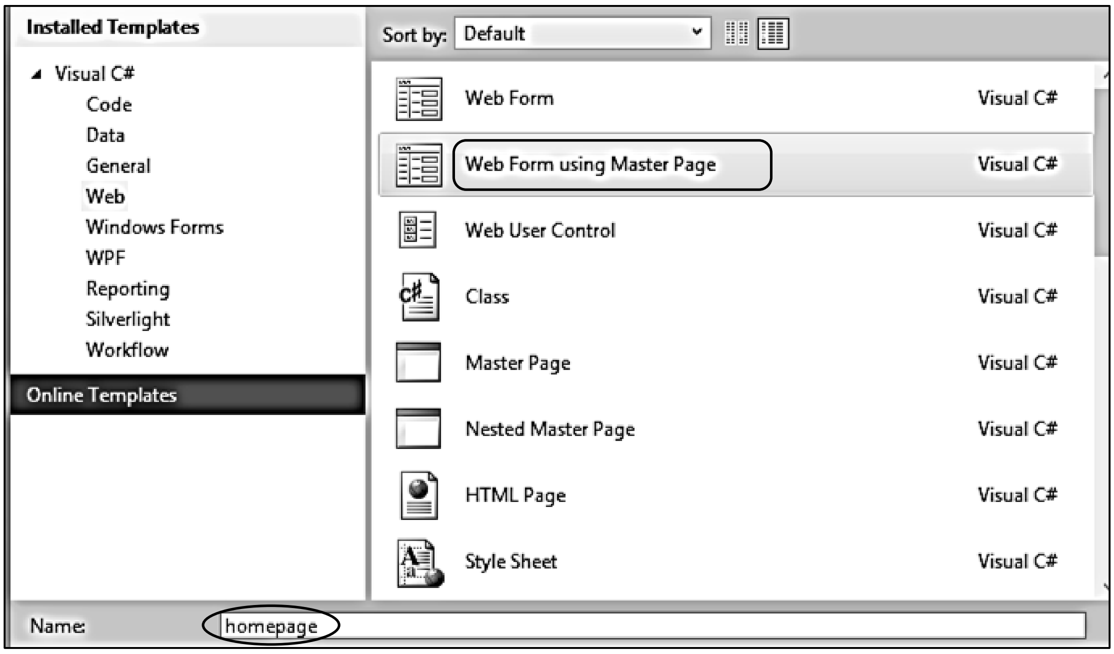
Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Master Page***. Give the name '**website**'.



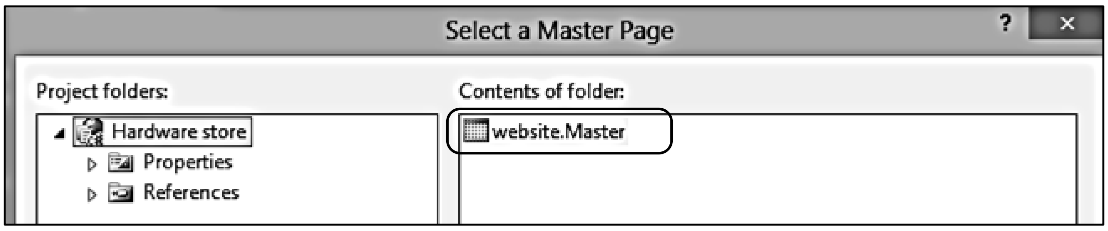
Name: website

The website master will display the name of the hardware store and provide the menu system.

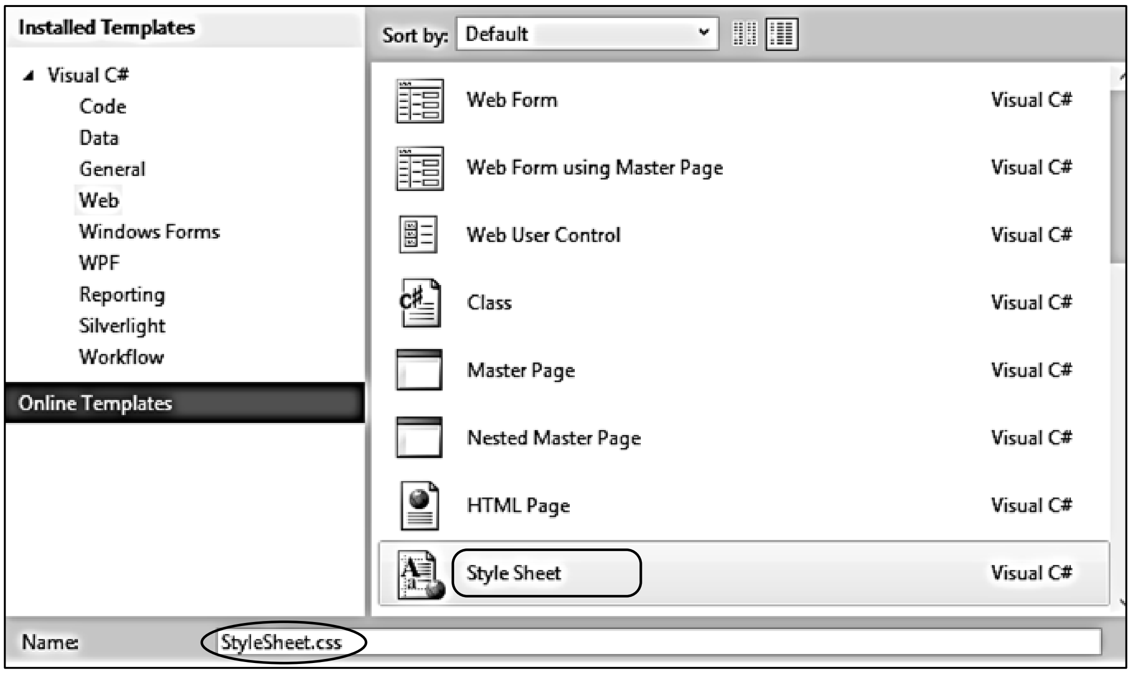
We will create a homepage which uses the website master. Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give the name '**homepage**'.



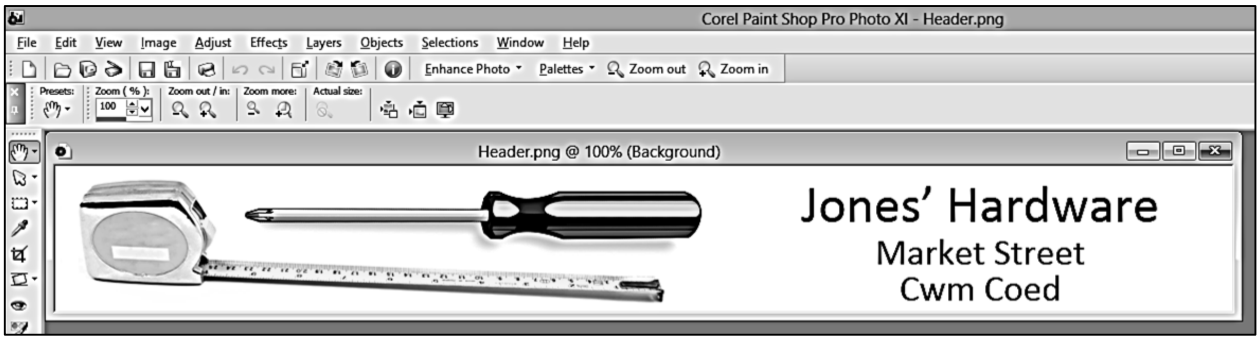
Accept ***website.Master*** as the master page and click the '***OK***' button.



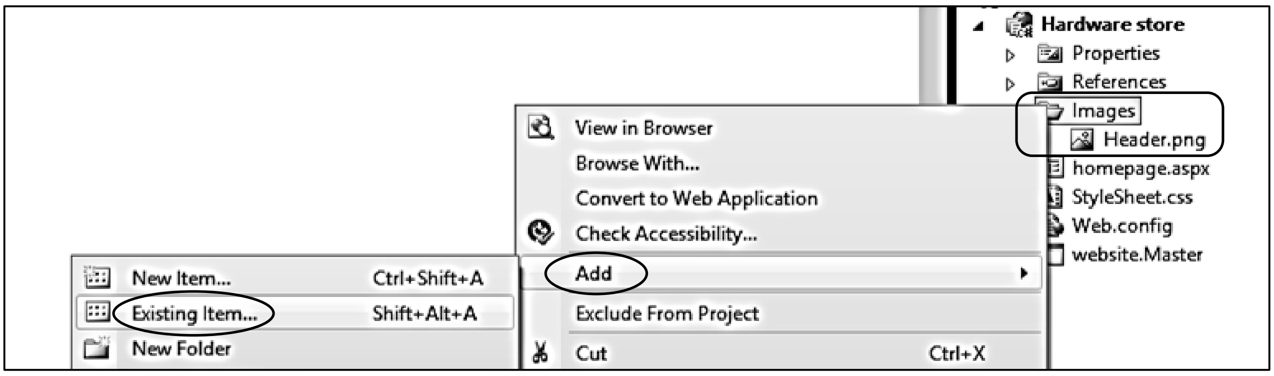
The final step before we begin programming is to create a style sheet. Return to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Style Sheet*** . Give the name '**StyleSheet**'.



Use any suitable graphics editing program such as **Photo Shop** or **Paint Shop** to create a header for the web pages. This should be approximately 1100 pixels wide by 140 pixels high.


Corel Paint Shop Pro Photo XI - Header.png
File Edit View Image Adjust Effects Layers Objects Selections Window Help
Enhance Photo Palettes Zoom out Zoom in
Header.png @ 100% (Background)
Jones' Hardware
Market Street
Cwm Coed

Save the file in .PNG or .JPG format. Return to Visual Studio, go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / Folder*** and give the name '**Images**'. Click right on the ***Images*** folder icon, select ***Add / Existing item***, and upload the ***header*** file to the ***Images*** folder.



Open the StyleSheet and add formatting commands to display the header:

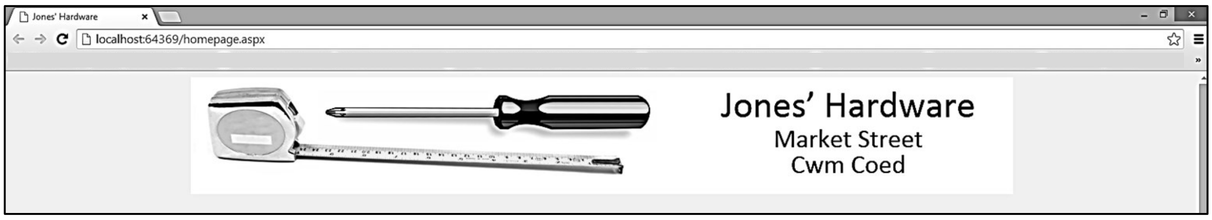
```
body
{
    background-color: #F0F0F0;
    font-family: Arial, Helvetica, sans-serif;
    color: black;
    font-size: small;
}

#wrapper
{
    width: 1092px;
    height: 2000px;
    margin:0 auto;
    background-color: #F0F0F0;
}

#header
{
    height: 155px;
    background-image : url(images/Header.png);
    background-repeat: no-repeat;
    background-color: #FFFFFF;
}
```

Return to the ***website.Master*** code page and add commands to:

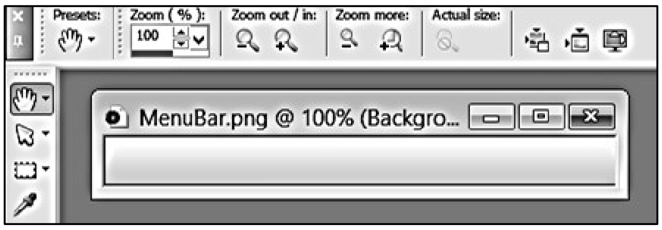
- give the title Jones' Hardware for the browser window
- link to the Style Sheet
- create divisions for the main page and header areas

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Jones' Hardware</title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
    <link rel="Stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
    <form id="form1" runat="server">
        <div id="wrapper">
            <div id="header">
            </div>
            <div>
                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
                </asp:ContentPlaceHolder>
            </div>
        </div>
    </form>
</body>
</html>
```

Select the ***homepage.aspx*** file. Build and run the page, and check that the ***header*** image is displayed correctly in the browser. Close the browser and stop debugging.



We will now create a menu bar, similar to the one used by the Snowdonia website in an earlier chapter. Begin by creating a background image for the menu. A gradient fill in shades of grey would be suitable. This should be 32 pixels in height. Width is not important, as the image will be repeated horizontally to produce the menu bar. Save the file as ***MenuBar.png***

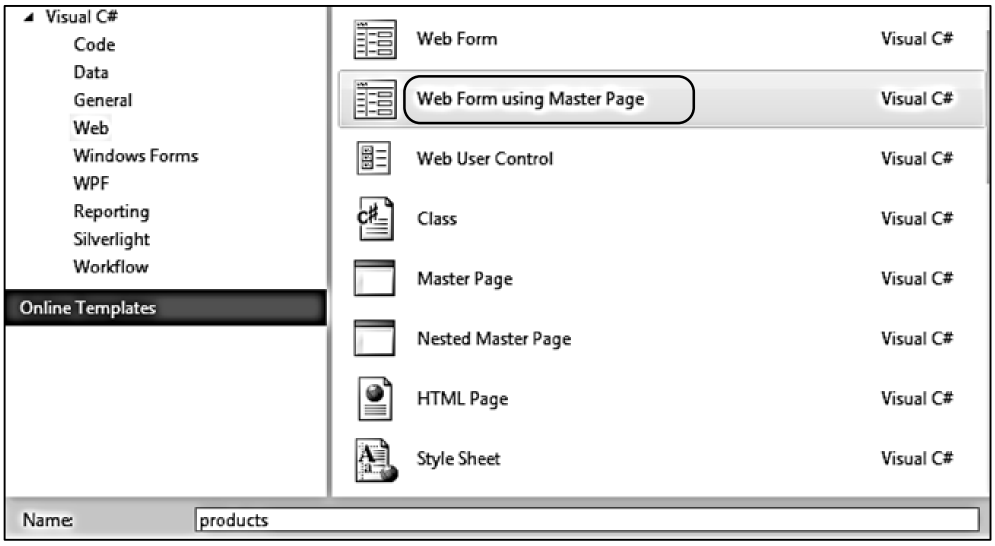


Go to the ***Solution Explorer*** window and upload the ***menuBar*** graphics file to the ***Images*** folder using the ***Add / Exisiting item*** option.

The website will have four public pages:

- **homepage.aspx**, which can give a general description of the business.
- **products.aspx**, which can give general information about products available in the store.
- **services.aspx**, which could give information about services avaialble, such as kitchen or bathroom fitting.
- **contactUs.aspx**, which can display a map of the shop location and other contact details.

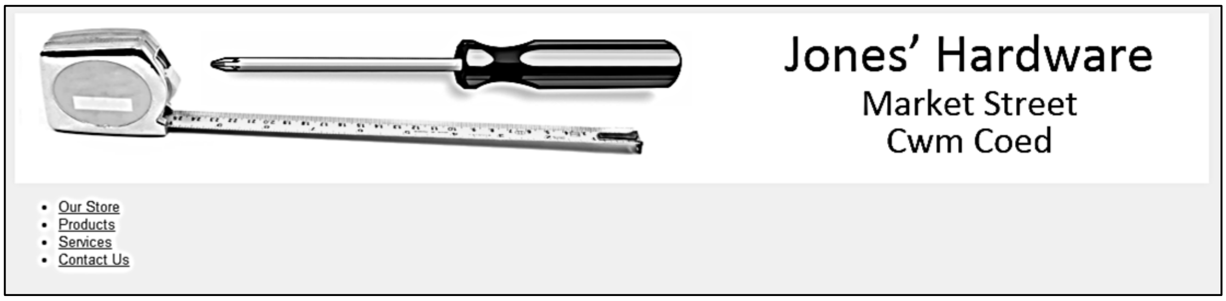
We have already produced the **homepage.aspx** file. In the ***Solution Explorer*** window, right click the **Hardware store** project icon and use the ***Add / New Item*** option to create each of the remaining pages: ***products.aspx, services.aspx*** and ***contactUs.aspx.*** In each case, choose '***Web Form using Master Page'***:



Return to the ***website.Master*** page and add code to create the menu using list items. We will put the menu code into a division called '**navigation**':

```
<form id="form1" runat="server">
   <div id="wrapper">
       <div id="header">
       </div>

       <div id="navigation">
           <ul id="nav">
               <li><a href="homepage.aspx">Our Store</a></li>
               <li><a href="products.aspx">Products</a></li>
               <li><a href="services.aspx">Services</a></li>
               <li><a href="contactUs.aspx">Contact Us</a></li>
           </ul>
       </div>

       <div>
           <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
           </asp:ContentPlaceHolder>
       </div>
   </div>
```

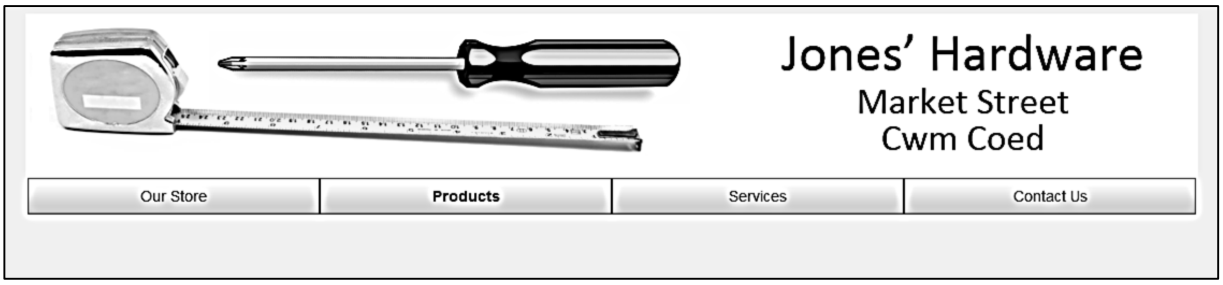
Build and run the homepage. The menu items should appear as a simple bullet point list.



The next step is to create a row of menu buttons from these list items. Close the web browser and stop debugging. Go to the ***StyleSheet*** file and add formatting code to handle the menu system. The new code can be added to the end of the Style Sheet after the ***#header*** block:

```
#navigation
{
   width: 1150px;
   height: 40px;
}
#nav
{
    list-style: none;
}
#nav ul
{
    list-style: none;
    display: none;
}
#nav li
{
    background-image: url(Images/MenuBar.png);
    font-size: 14px;
    float: left;
    position: relative;
    width: 270px;
    height: 32px;
    left: -38px;
    top: -15px;
    border: 1px solid #000000;
}
#nav a:link, #nav a:active, #nav a:visited
{
    display:block;
    color: #000000;
    text-decoration: none;
    padding: 8px;
    text-align: center;
}
#nav a:hover
{
    font-weight:bold;
}
```

Build and run the homepage. The menu should now appear as a set of buttons, with the captions changing to bold font as the mouse moves over them.



Close the browser and stop debugging.

The pages for ***Our Store***, ***Products*** and ***Services*** will display images. Obtain suitable .JPG photographs from the Internet and upload them to the ***Images*** folder using the ***Add / Exisiting Item*** option. Give the images the names ***ourStore.jpg***, ***products.jpg*** and ***services.jpg***.

ourStore.jpg

products.jpg

services.jpg

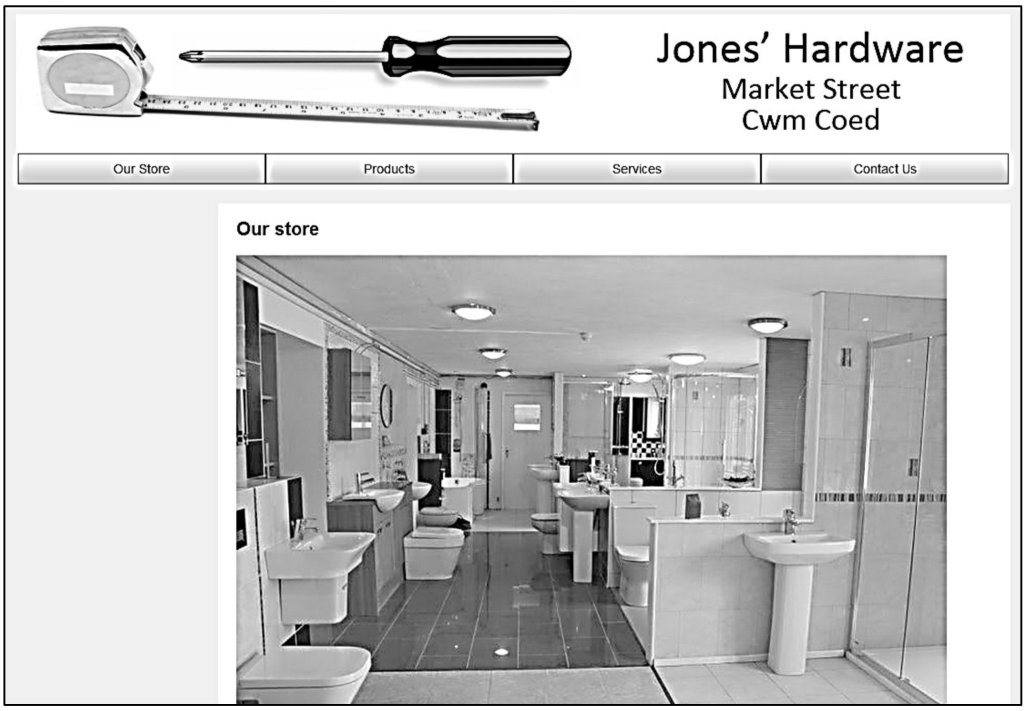
Go to the homepage code and add commands to the '***content2***' section:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/website.Master"
AutoEventWireup="true" CodeBehind="homepage.aspx.cs"
Inherits="Hardware_store.homepage" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="pageContent">
        <h2>Our store</h2>
        <img src="Images/ourStore.jpg" width="780"  />
    </div>
</asp:Content>
```

We have created a division called ***pageContent***. Load the ***StyleSheet*** page and add commands to format this division:

```
    #nav a:hover
    {
        font-weight:bold;
    }

    #pageContent
    {
        background-color: #FFFFFF;
        float: right;
        width: 850px;
        height: 600px;
        padding-left: 20px;
    }
```

Build and run the homepage. The picture and page title should be displayed. Notice that the content area is aligned with the right end of the menu bar. This is deliberate, as we will be using the left section of the web page for additional menu options. Close the browser and stop debugging.



Add similar code to the '***content2***' section of the ***products.aspx*** page:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="pageContent">
        <h2>Products</h2>
        <img src="Images/products.jpg" width=780  />
    </div>
</asp:Content>
```

Add code to the '***content2***' section of the ***services.aspx*** page:

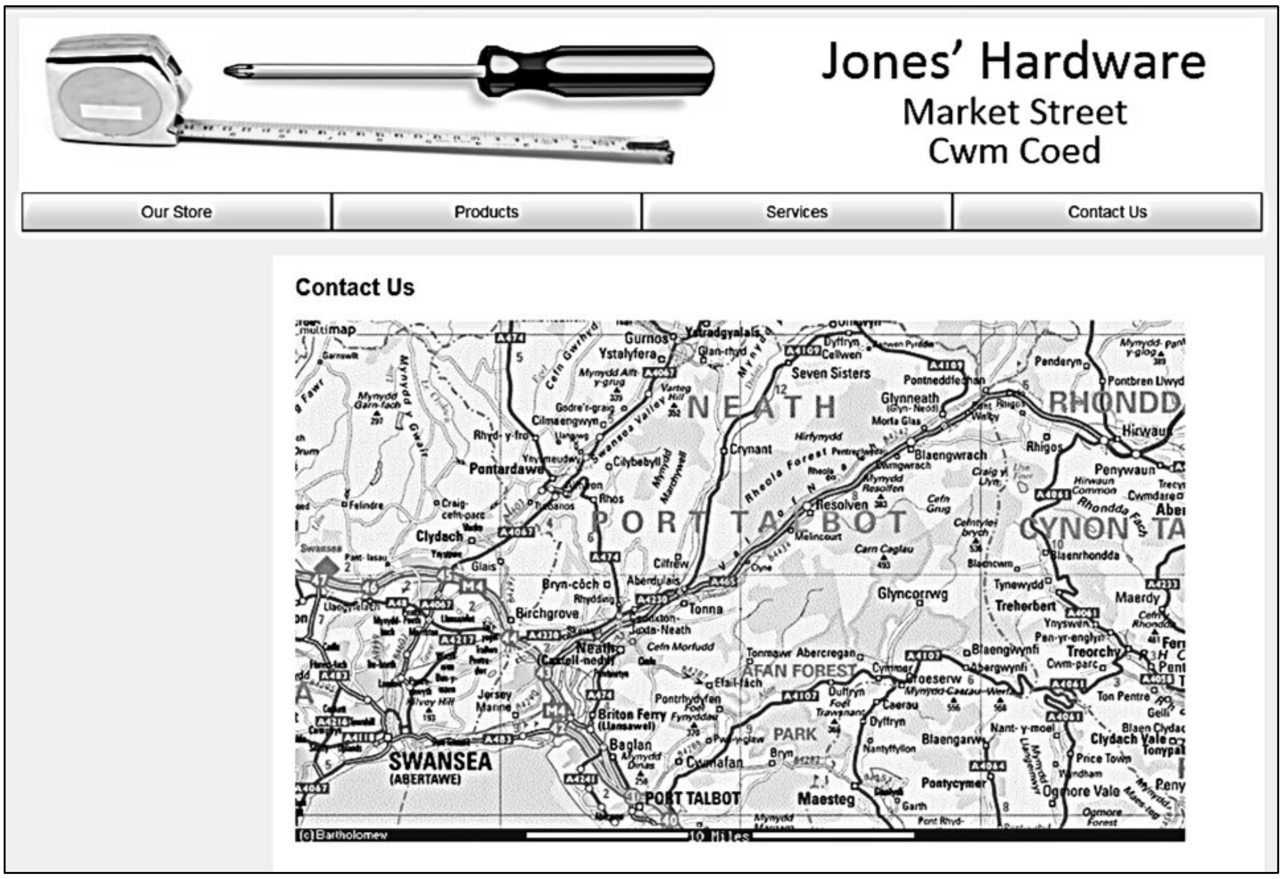
```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="pageContent">
        <h2>Services</h2>
        <img src="Images/services.jpg" width=780  />
    </div>
</asp:Content>
```

Build and run the web site. Check that it is possible to change pages using the menu buttons, and that all images are displayed correctly.

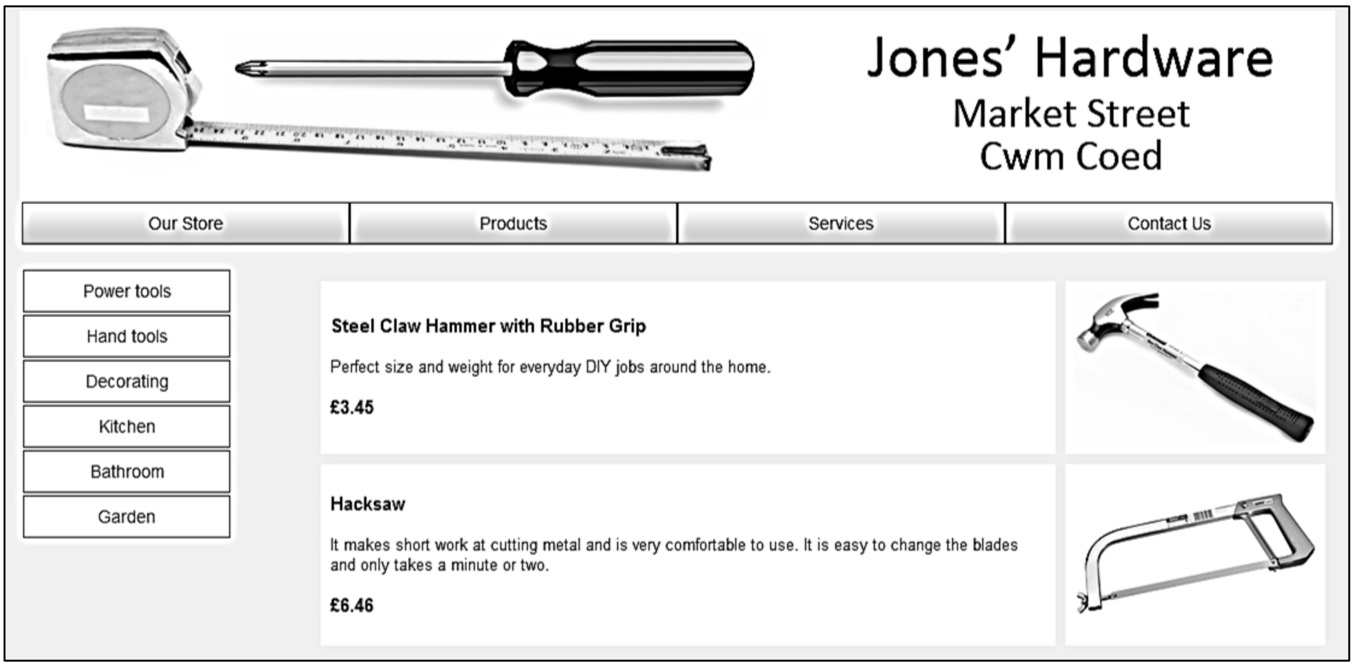
Close the browser and stop debugging. We have not yet added any content to the ***Contact Us*** page. In this case, a map would be approriate. Upload a suitable map file to the ***Images*** folder, and add code to the ***contactUs.aspx*** page using the correct file name for the map:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="pageContent">
        <h2>Contact Us</h2>
        <img src="Images/map.gif" width="780"  />
    </div>
</asp:Content>
```

Build and run the website. Check that the map is displayed correctly on the ***Contact Us*** page:



We can now turn our attention to the main function of the web site, which is to display information about the hardware items for sale. A convenient way to arrange this is to provide a category menu on the left of the page. When the user selects a product category, the available items will be listed, along with picture images:



Steel Claw Hammer with Rubber Grip

Perfect size and weight for everyday DIY jobs around the home.

£3.45

Hacksaw

It makes short work at cutting metal and is very comfortable to use. It is easy to change the blades and only takes a minute or two.

£6.46

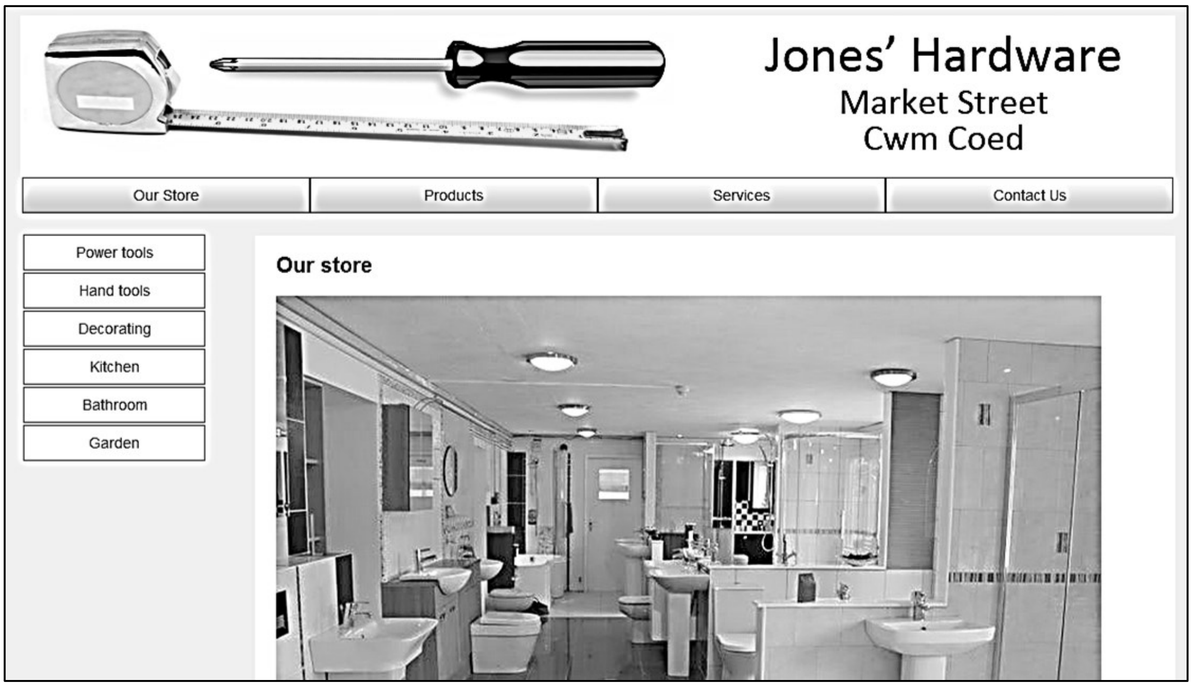
Close the browser, return to the ***website.Master*** code page and stop debugging. Add commands to create the category menu:

```
<div id="navigation">
    <ul id="nav">
        <li><a href="homepage.aspx">Our Store</a></li>
        <li><a href="products.aspx">Products</a></li>
        <li><a href="services.aspx">Services</a></li>
        <li><a href="contactUs.aspx">Contact Us</a></li>
    </ul>
</div>

<div id="products">
    <ul id="cat">
        <li><a href="displayItems.aspx?category=power">Power tools</a></li>
        <li><a href="displayItems.aspx?category=hand">Hand tools</a></li>
        <li><a href="displayItems.aspx?category=decorating">Decorating</a></li>
        <li><a href="displayItems.aspx?category=kitchen">Kitchen</a></li>
        <li><a href="displayItems.aspx?category=bathroom">Bathroom</a></li>
        <li><a href="displayItems.aspx?category=garden">Garden</a></li>
    </ul>
</div>

<div>
    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>
</div>
```

The content menu can be formatted in a similar way to the main menu bar. Open the ***StyleSheet*** file and add code sections to the end of the file:

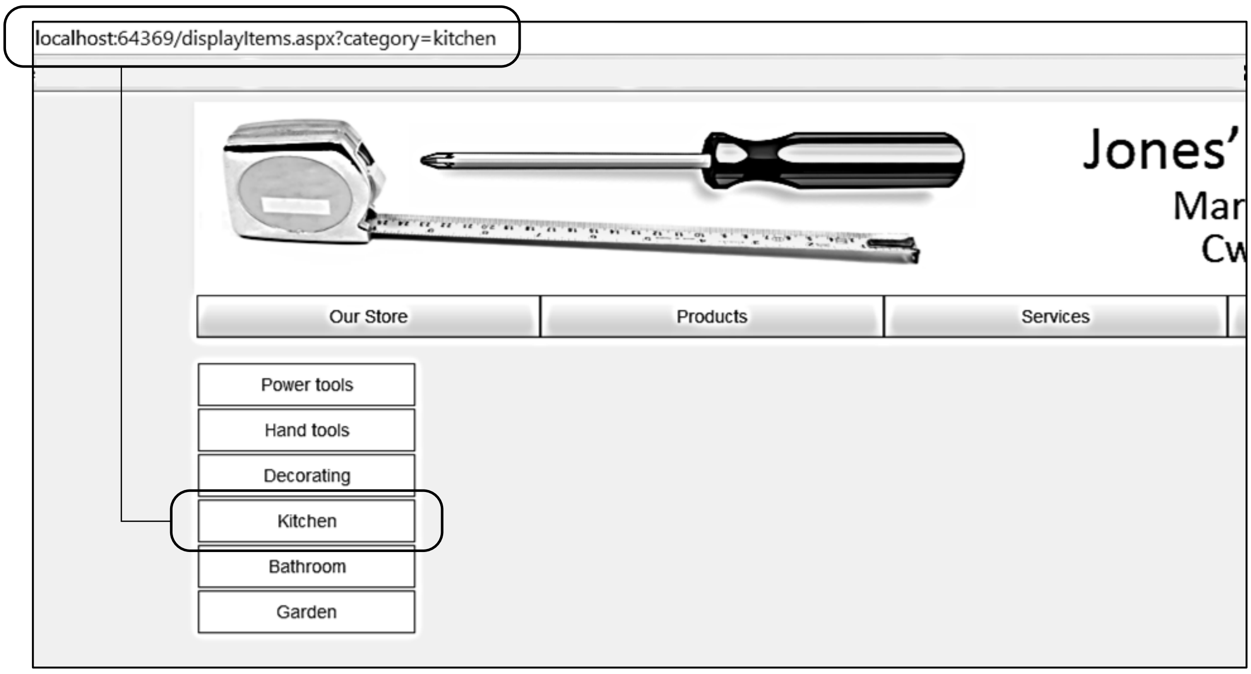
```
#products
{
    background-color: #F0F0F0;
    float: left;
    width: 200px;
    list-style: none;
}

#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 homepage. Check that the category menu is correctly displayed on the left of the page. Close the browser and stop debugging.



When the user selects a category, we will arrange for a page to open to display the available items. Create this by going to the Solution Explorer window and right clicking the **Hardware store** icon. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give this file the name '**displayItems**'.

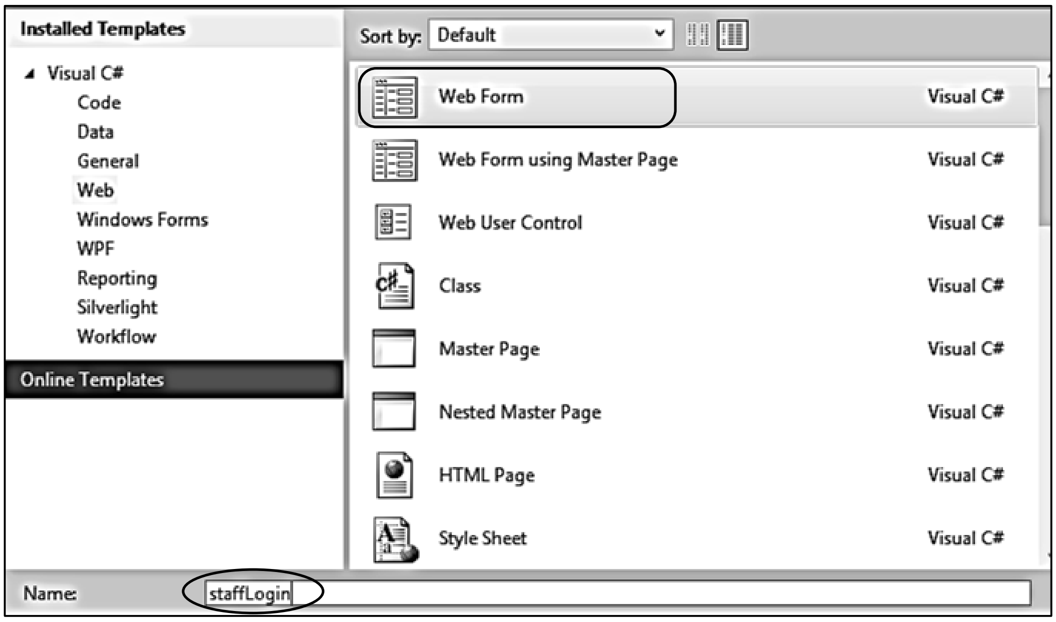
Build and run the website, and click any button on the category menu at the left of the page. The blank ***displayItems.aspx*** page should open. Notice that the category selected is identifed by a variable attached to the page URL:



localhost:64369/displayItems.aspx?category=kitchen

Close the browser and stop debugging. To help the staff of the hardware store who will be operating the web site, we will produce a content management system for easy updating of product information. This will have a web page user interface, so needs to be password protected to prevent unauthorised access.

We begin by creating a staff log-in screen. Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose a simple ***Web Form***. Give the name '**staffLogin**'.



Open the ***staffLogin.aspx*** page and add code to:

- give the title '***Staff***' for the browser window
- link to the Style Sheet
- create divisions for the main page and header areas

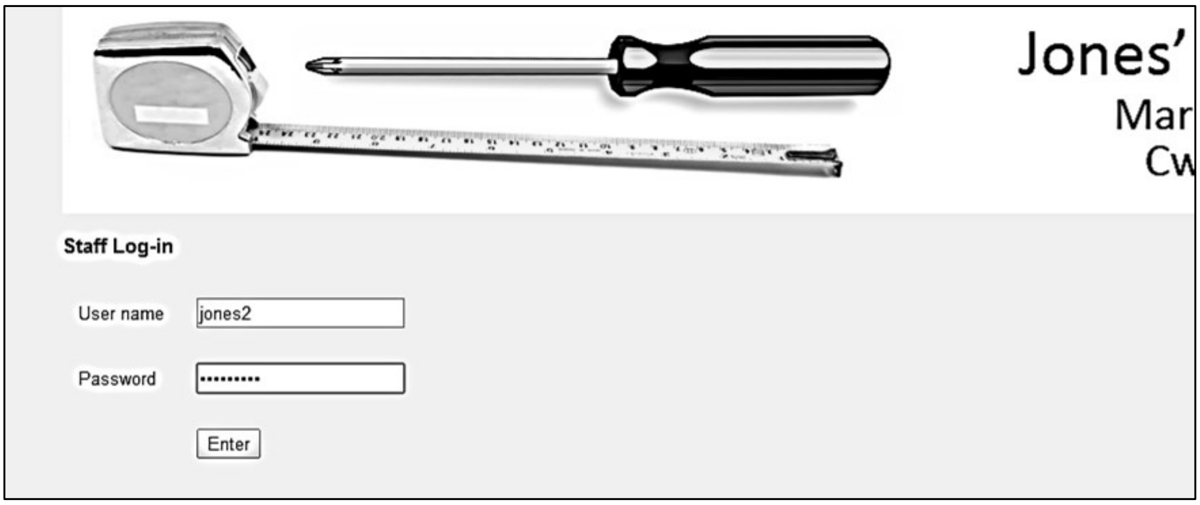
```
<head id="Head1" runat="server">
    <title>Staff</title>
    <link rel="Stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
    <form id="form1" runat="server">
        <div id="wrapper">
            <div id="header">
            </div>
        </div>
    </form>
</body>
</html>
```

Build and run the ***staffLogin*** page. This should display the header image. Close the web browser and stop debugging.

We will now add screen components to allow members of staff to enter a username and password:

```
<div id="wrapper">
  <div id="header">
  </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>
```

Build and run the staffLogin page. Entry boxes should appear. Notice that the password box is set to conceal the text which is being entered.



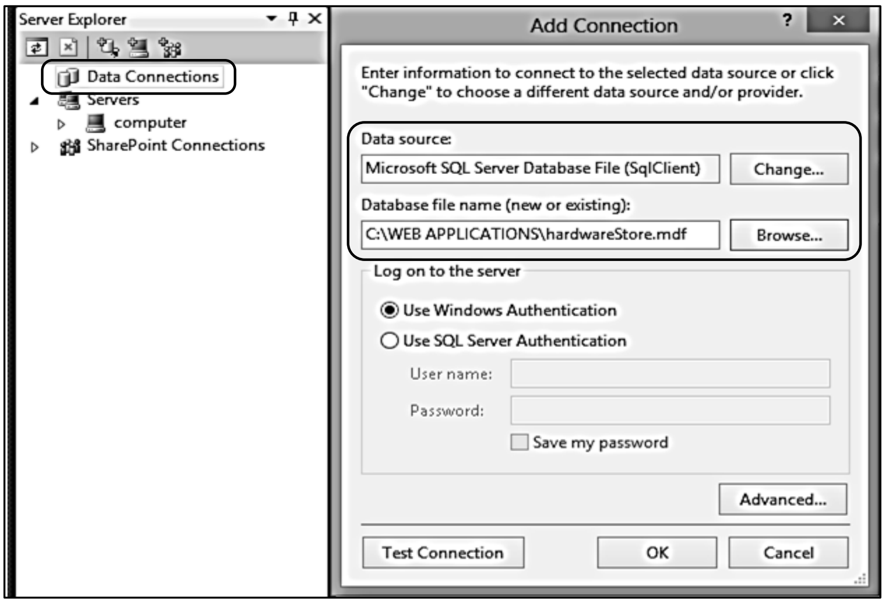
Staff Log-in

User name jones2

Password •••••••••

Close the browser and stop debugging. Before proceeding further with the web pages, we will require a database. This will have tables to contain the staff usernames and passwords, and details of the products on sale.

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 '***hardwareStore.mdf***' for the database file.



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server Database File (SqlClient)

Database file name (new or existing): C:\WEB APPLICATIONS\hardwareStore.mdf

Log on to the server

(● Use Windows Authentication) ○ Use SQL Server Authentication

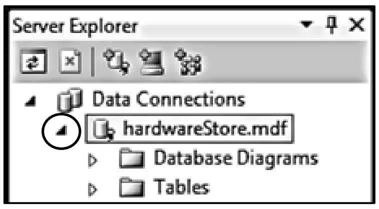
User name:

Password:

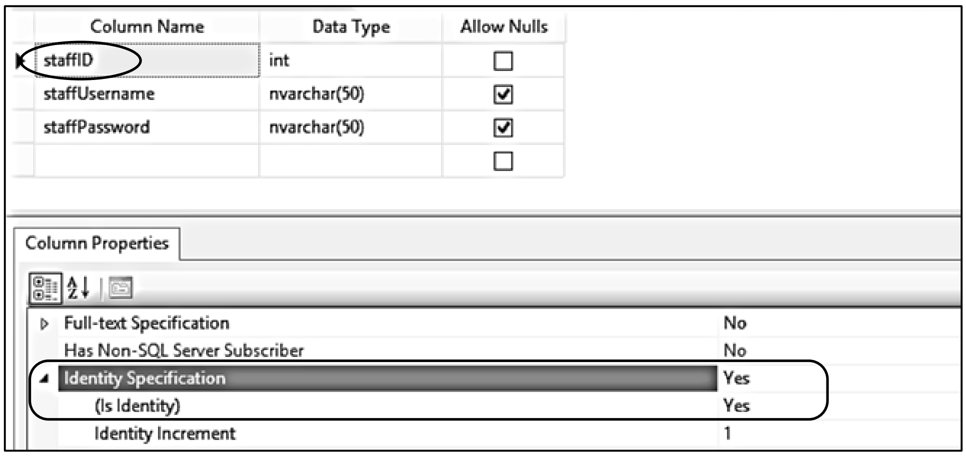
☐ Save my password

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

An icon for the newly created hardwareStore database should appear in the Server Explorer window. Click the small arrow to the left of the ***hardwareStore.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:



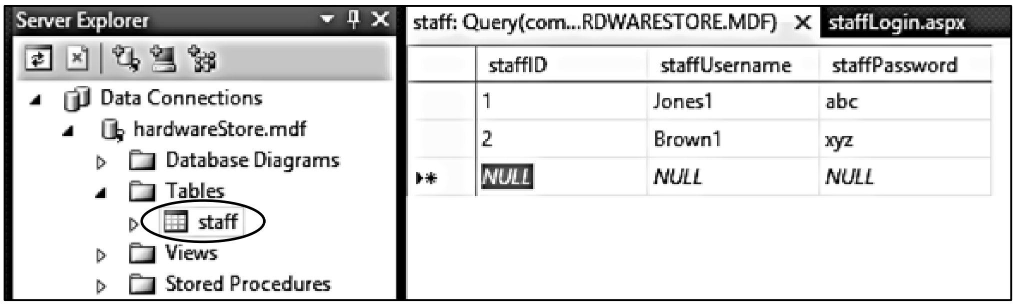
Column Name	Data Type	Allow Nulls
staffID	int	☐
staffUsername	nvarchar(50)	✓
staffPassword	nvarchar(50)	✓
		☐

Column Properties

Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1

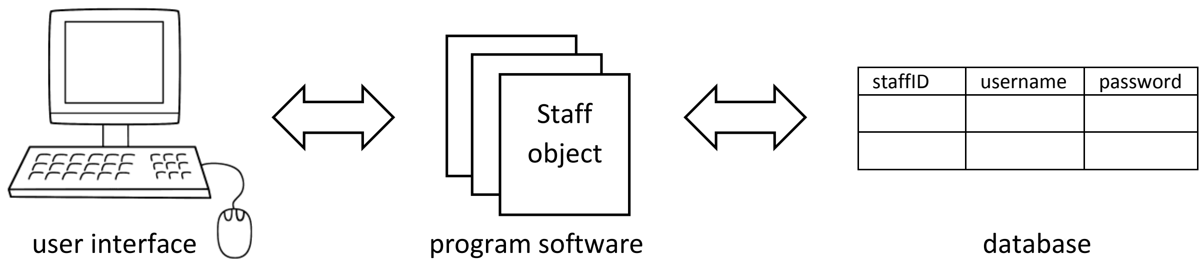
Close the table using the small ‘**X**’ icon at the top right of the form, then save this with the name ‘**staff**’.

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.

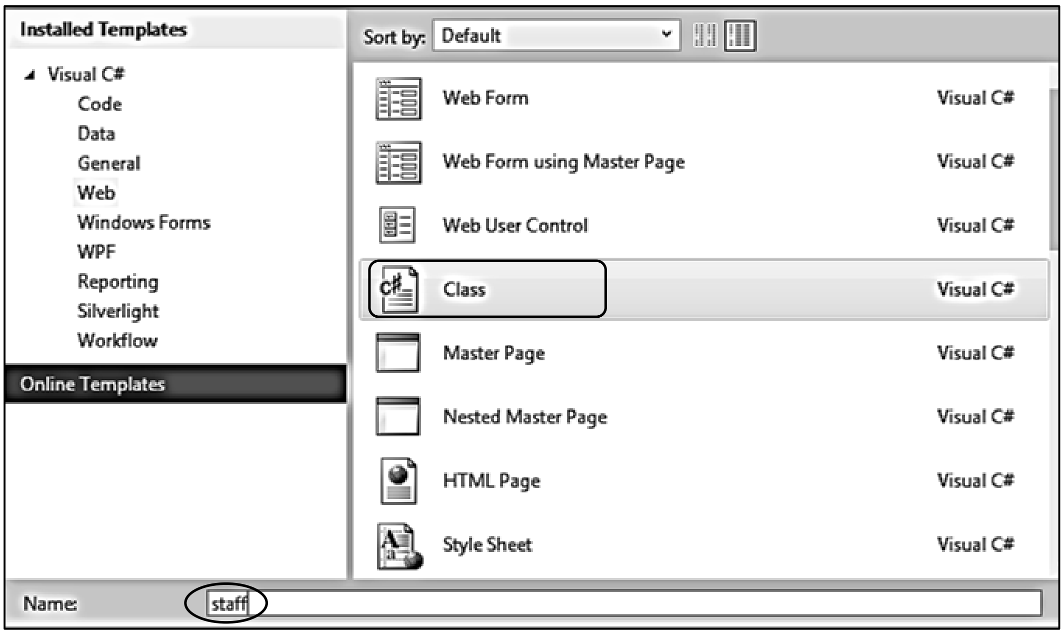


	staffID	staffUsername	staffPassword
	1	Jones1	abc
	2	Brown1	xyz
▶*	NULL	NULL	NULL

When developing web applications involving a databse, there can be considerable advantages in using an object oriented approach. Records are loaded from the database and held in the RAM memory as a set of objects. It is then quick to search for required objects or display the object data without the need for further database operations. We will use this approach in the current project:



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



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 Hardware_store
{
    public class staff
    {
        public static int staffCount = 0;
        public static staff[] staffObject = new staff[10];
        public static string databaseLocation =
                              "C:\\WEB APPLICATIONS\\hardwareStore.mdf;";
        public int staffID { get; set; }
        public string staffUsername { get; set; }
        public string staffPassword { get; set; }

        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:

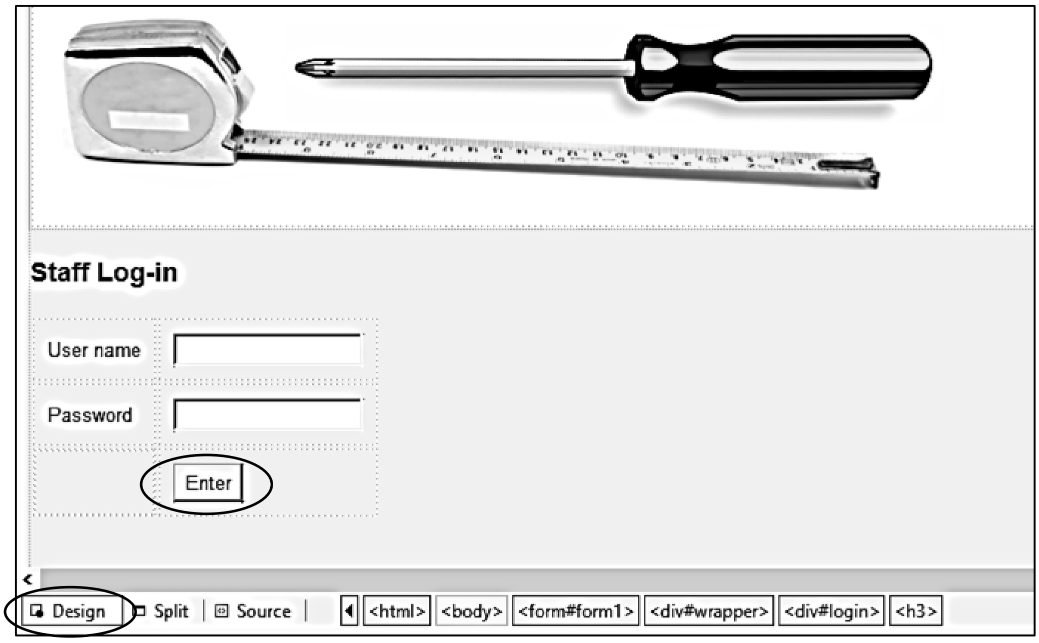
```
try
{
    cnTB.Open();
    SqlCommand cmStaff = new SqlCommand();
    cmStaff.Connection = cnTB;
    cmStaff.CommandType = CommandType.Text;
    cmStaff.CommandText = "SELECT * FROM staff";
    SqlDataAdapter daStaff = new SqlDataAdapter(cmStaff);
    daStaff.Fill(dsStaff);
    cnTB.Close();

    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.



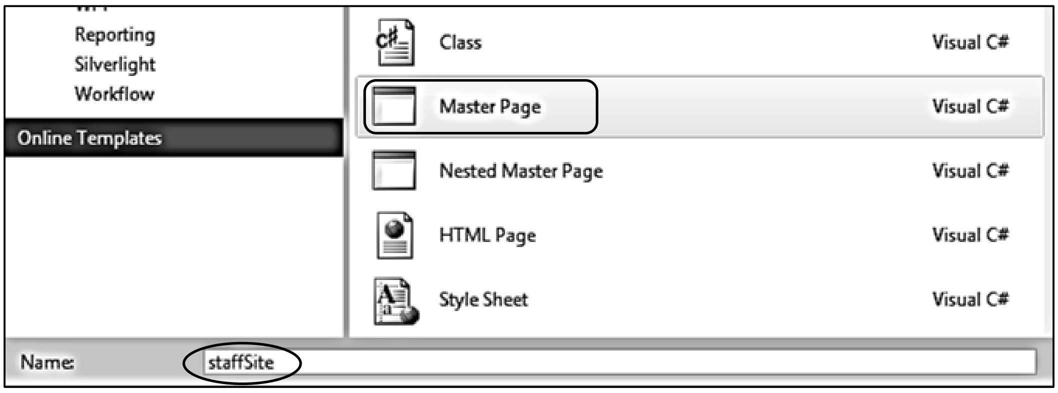
The staff usernames and passwords are held in the set of ***staff objects***. We can add a section of code to check through each of the objects to see if there is a match with the data entered on the log-in screen.

```
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)
    {
        Response.Redirect("editStockItem.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 ***editStockItem.aspx.*** We will work on this page next…

The functions required by the stock system are: to add new items, edit existing items, and to delete items no longer required.

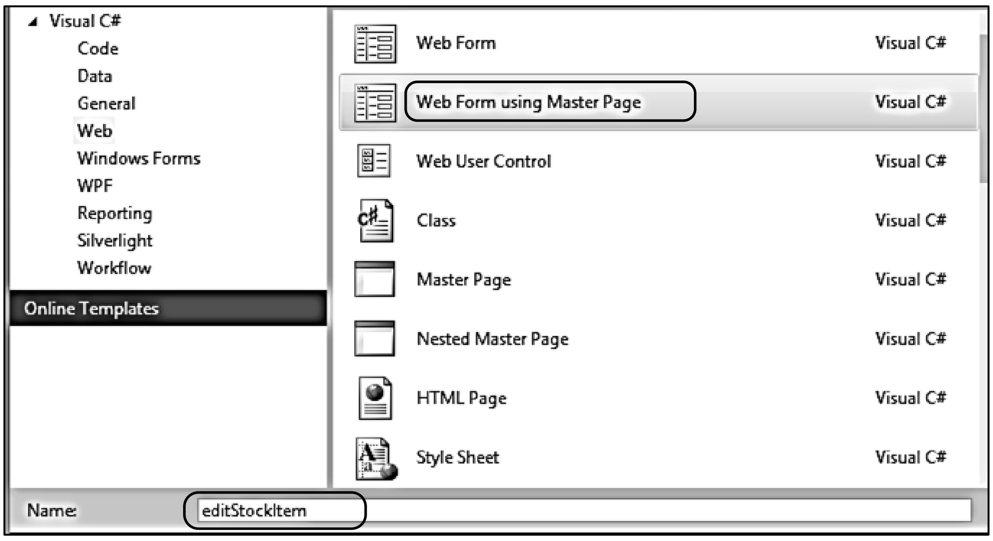
We will set up a menu for the stock system in a similar way to the main web site. Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Master Page***. Give the name '**staffSite**'.



Add code to the staffSite.Master file to link to the style sheet and create a menu bar:

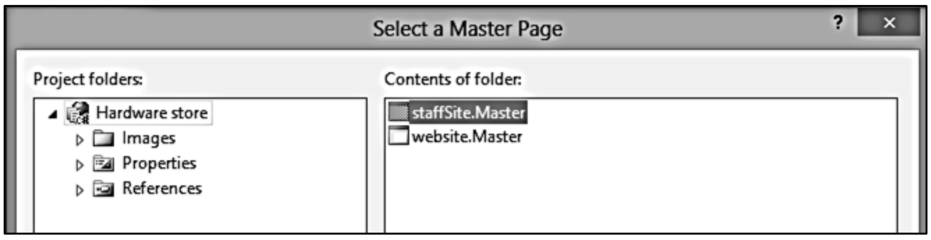
```
<head runat="server">
    <title>Staff</title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
    <link rel="Stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
    <form id="form1" runat="server">
    <div id="wrapper">
        <div id="header">
        </div>

        <div id="navigation">
            <ul id="nav">
            <li><a href="addStockItem.aspx">Add stock item</a></li>
            <li><a href="editStockItem.aspx">Edit stock item</a></li>
            <li><a href="deleteStockItem.aspx">Delete stock item</a></li>
            <li><a href="homepage.aspx">Return to homepage</a></li>
            </ul>
        </div>
        <div id="content">
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

            </asp:ContentPlaceHolder>
        </div>
      </div>
     </form>
```

We can now produce the pages which are linked to the staff menu. Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give the name '**editStockItem**'.



Name: editStockItem

Click the '***Add***' button, but this time choose ***staffSite.Master*** as the master page:

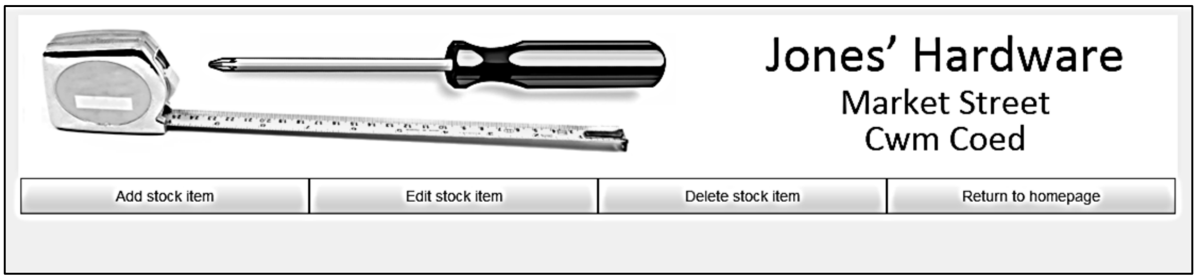


Create two further web pages:

addStockItem.aspx
deleteStockItem.aspx

by selecting ***Add / New Item*** and choosing ***Web Form using Master Page***. In each case, select ***staffSite.Master*** as the master page.

Build and run the ***staffLogin*** webpage again. Enter a correct username and password. The ***editStockItem.aspx*** page should appear, with the staff menu options displayed.



Check that the ***Add stock item*** and ***Delete stock item*** buttons load the corresponding web pages, and that the ***Return to homepage*** button takes the user back to the public website.

Before continuing, we need to attend to one further security issue. If an unauthorised user guessed the URL for a page within the staff system, such as ***editStockItem.aspx***, they would be able to by-pass the log-in procedure and load the page directly. It is simple to close this loophole.

Go to the ***staff.cs*** class file and add a line to the code:

```
public class staff
{
    public static bool login = false;

    public static int staffCount = 0;
    public static staff[] staffObject = new staff[10];
    public static string databaseLocation =
                                 "C:\\WEB APPLICATIONS\\hardwareStore.mdf;";
```

We have created a Boolean (true/false) variable to record whether a valid log-in has been made. Initially we set this to false.

Go now to the C# code page for ***staffLogin.aspx***. Add a line of code to the ***enter_Click*** method:

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

            . . . . . . . . . . . . .

    if (found == true)
    {
        staff.login = true;
        Response.Redirect("editStockItem.aspx");
    }
}
```

If a valid log-in is made, the ***login*** variable is now set to true. Go now to the ***editStockItem.aspx*** page and open the C# window. Add lines of code to the ***Page_Load*** method.

```
public partial class editStockItem : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (staff.login == false)
        {
            Response.Redirect("staffLogin.aspx");
        }
    }
}
```

When an attempt is made to open the ***editStockItem.aspx*** page, the computer first checks the ***login*** variable to see whether a valid log-in has been made. If not, the user is redirected back to the log-in screen.

Build and run the website homepage. Make an attempt to load the editStockItem.aspx page by entering its URL in the browser. You should be redirected instead to the staff log-in screen.

Add the same lines of code to the Page_Load methods of the pages addStockItem.aspx and deleteStockItem.aspx.

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

We can now continue work on the stock system. We will begin by setting up a screen to enter product details. Open the ***StyleSheet*** file and add three further blocks of formatting commands:

```
#entryForm
{
    margin: 20px;
    padding: 20px;
    border: 1px solid #000000;
    height: 500px;
    background-color: #FFFFFF;
}

#itemEntry
{
    float: left;
}

#imageEntry
{
    float: right;
    width: 500px;
}
```

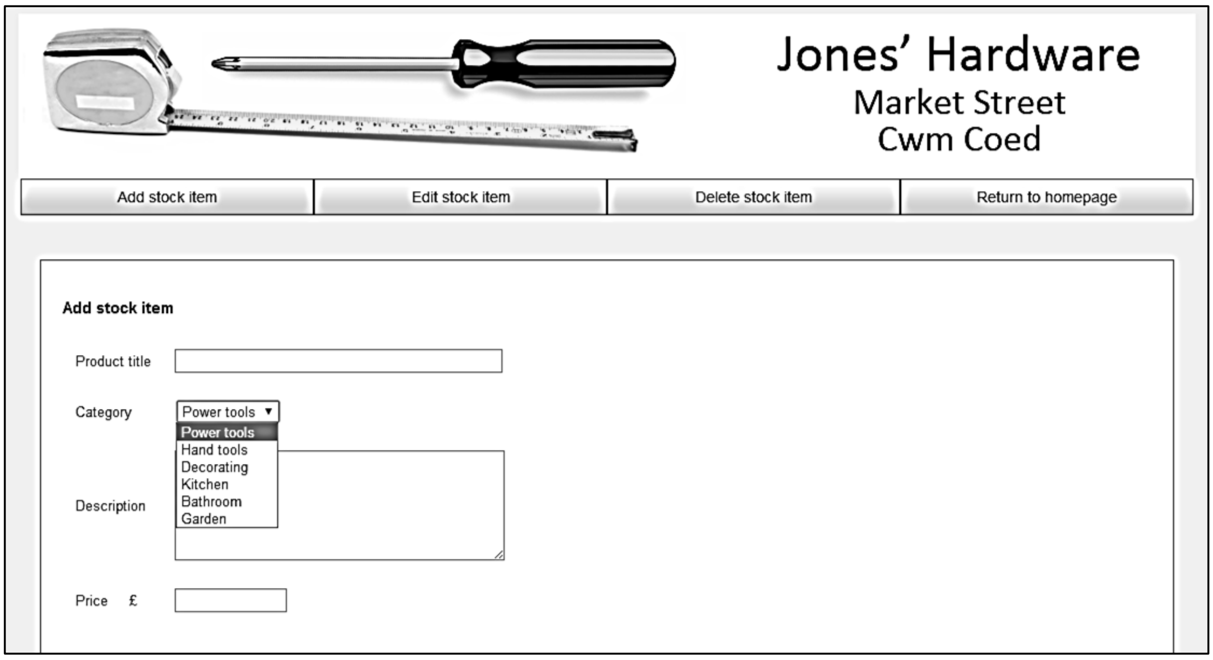
Go to the ***addStockItem.aspx*** page and set up divisions within the ***Content2*** section:

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="entryForm">
    <h3>Add stock item</h3>
    <div id="itemEntry">
    </div>
  </div>
</asp:Content>
```

Go to the ***itemEntry*** division and create a table. Inside this we will put a text box for entry of the product name, a drop down list for selecting a product category, and text boxes for entering a product description and price.

```
<div id="itemEntry" >
  <table border="0" cellpadding="10">
    <tr>
       <td>
          Product title
        </td>
        <td>
          <asp:TextBox
                ID="txtTitle" runat="server" Width="300px"></asp:TextBox>
        </td>
    </tr>
    <tr>
       <td>
         Category
        </td>
        <td>
            <asp:DropDownList ID="lstCategory" runat="server">
                <asp:ListItem Value="power">Power tools</asp:ListItem>
                <asp:ListItem Value="hand">Hand tools</asp:ListItem>
                <asp:ListItem Value="decorating">Decorating</asp:ListItem>
                <asp:ListItem Value="kitchen">Kitchen</asp:ListItem>
                <asp:ListItem Value="bathroom">Bathroom</asp:ListItem>
                <asp:ListItem Value="garden">Garden</asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>
     <tr>
       <td>
          Description
        </td>
        <td>
          <asp:TextBox ID="txtDescription" runat="server" Width="300px"
                                 TextMode="MultiLine" Rows="6"></asp:TextBox>
        </td>
    </tr>
    <tr>
       <td>
          Price &nbsp;&nbsp;&nbsp;  £
        </td>
        <td>
         <asp:TextBox ID="txtPrice" runat="server" Width="100px"></asp:TextBox>
        </td>
    </tr>
  </table>
</div>
```

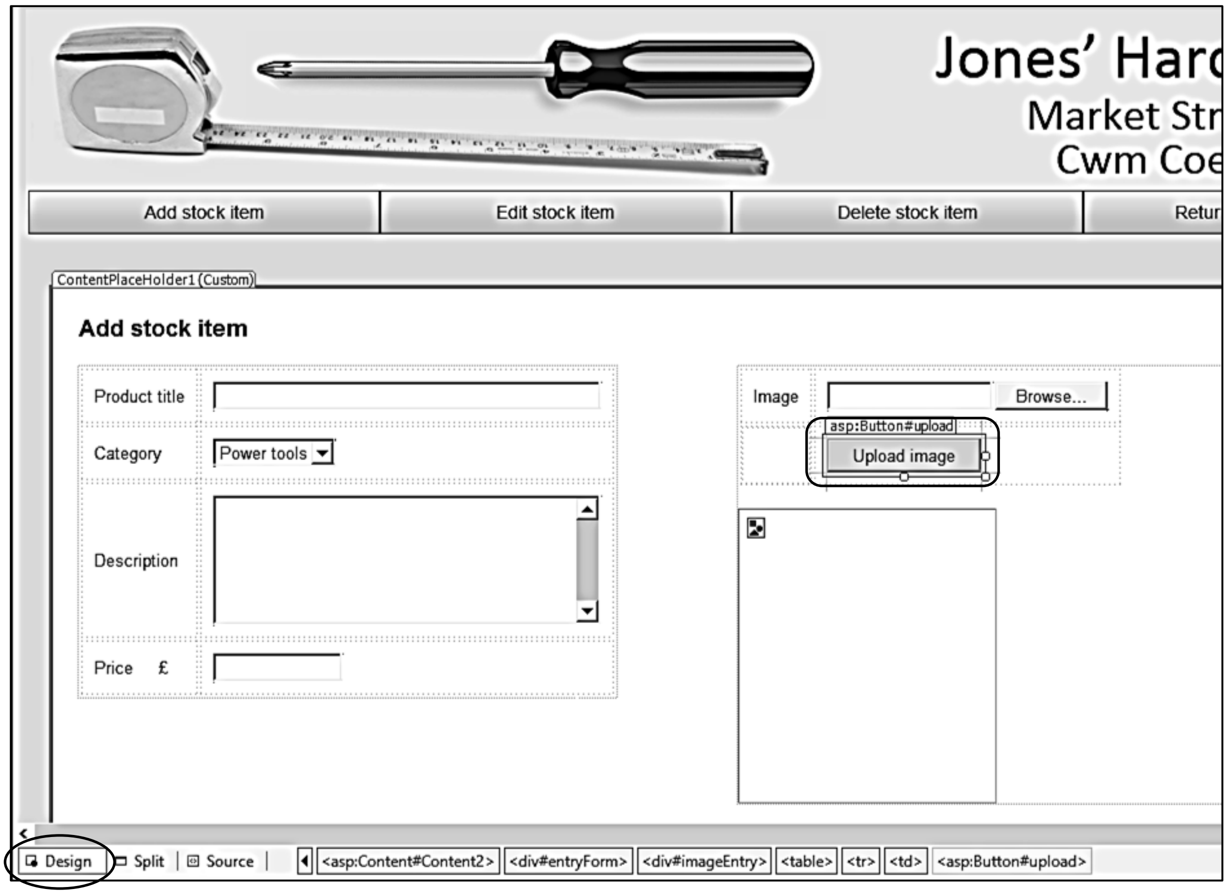
Build and run the staff website. After logging-in, select '***Add stock item***' from the menu bar. Check that text boxes and a drop down list are displayed correctly. Close the browser and stop debugging.



In addition to text data, we will upload a photograph of the new stock item. Add a division to the ***addStockItem.aspx*** page called '***imageEntry***' beneath the '***itemEntry***' division:

```
        </table>
      </div>

      <div id="imageEntry">
        <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='200' />
       </div>

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

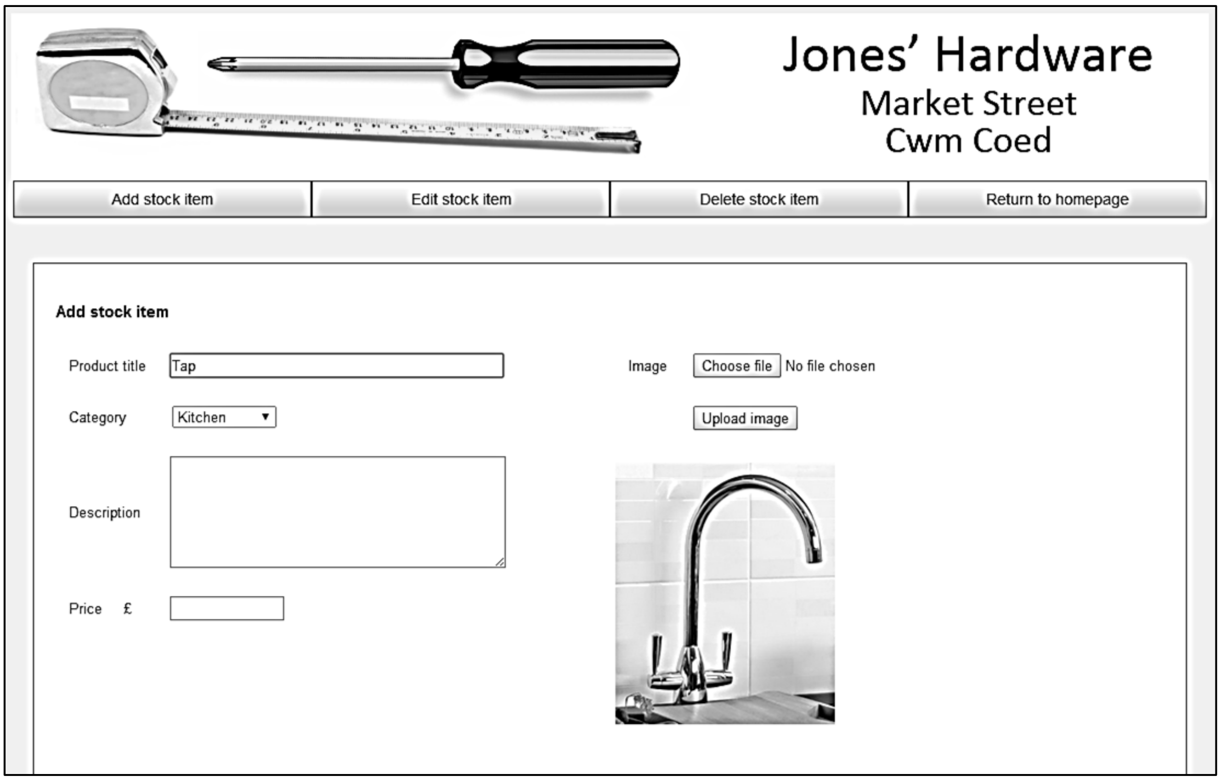
On the ***addStockItem.aspx*** page, click the Design button to change to the design view. Double click the ***Upload Image*** button to create a button click method.



Add lines of code to ***upload_Click( )***:

```
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;
        }

    }
    catch
    {

    }
}
```

Build and run the staff website. Log-in and select '***Add stock item***' from the menu bar. Use the '***Choose file***' option to select a photograph, then click the '***Upload image***' button. The photograph should be displayed. Note: this may not work in the ***Firefox*** browser due to security restrictions.

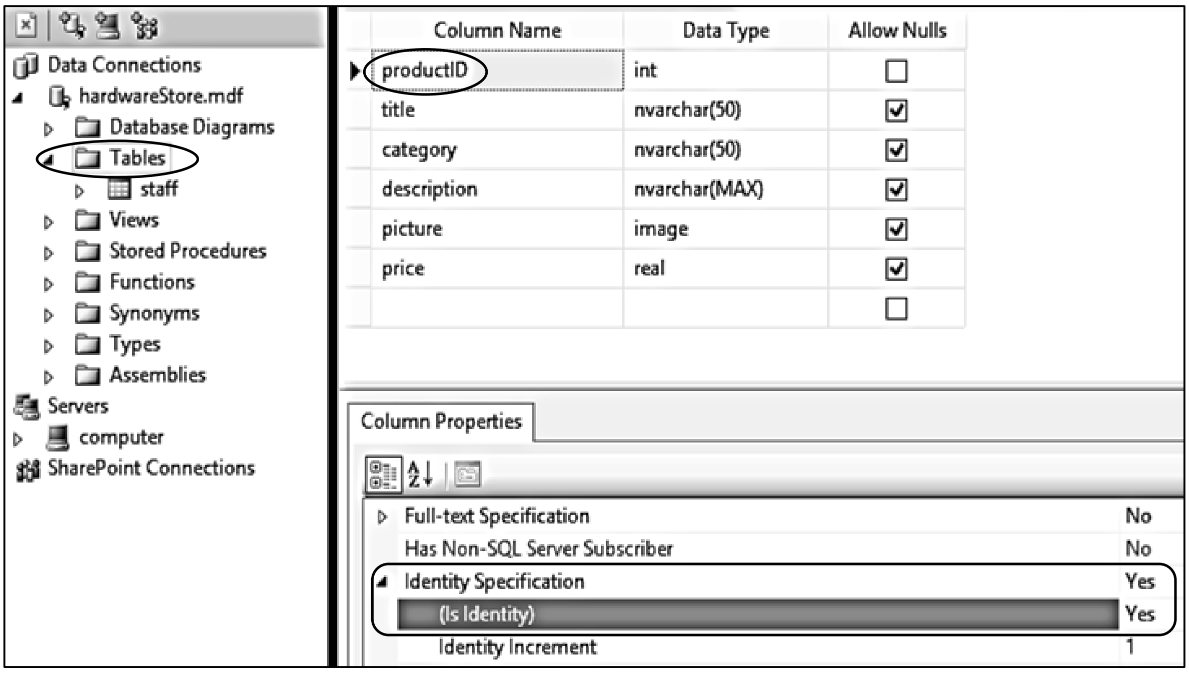


Add stock item

Product title	Tap	Image	Choose file No file chosen
Category	Kitchen		Upload image
Description			
Price £			

Now that the user interface for input of product data has been constructed, we will turn our attention to storing this data in a database table and the creation of ***objects*** to represent stock items.

Go to the Server Explorer window, right click the ***Tables*** icon for the ***hardwareStore*** database and select '***Add New Table***'. Enter fields as shown below. The ***productID*** field should be set as an auto-number by selecting a '***Yes***' value ***Identity Specification / (Is Identity)***.



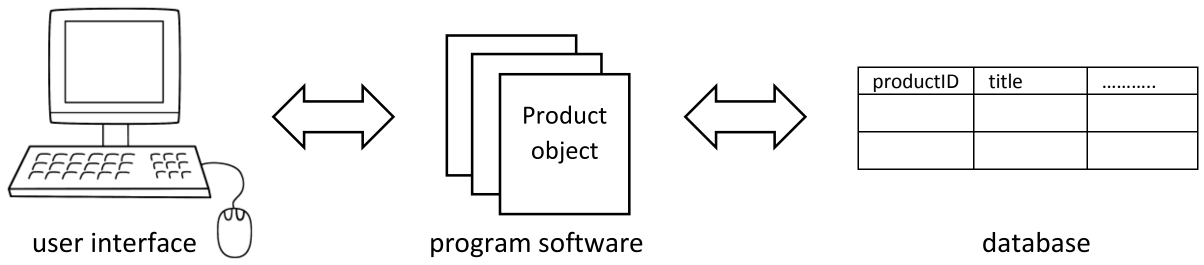
Column Name	Data Type	Allow Nulls
productID	int	☐
title	nvarchar(50)	✓
category	nvarchar(50)	✓
description	nvarchar(MAX)	✓
picture	image	✓
price	real	✓
		☐

Column Properties

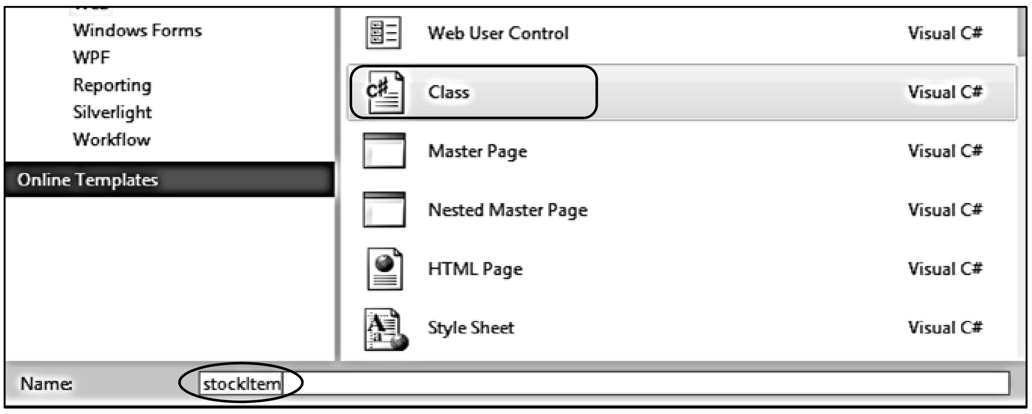
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1

Close the table and specify the table name as ‘**product**’.

We next need to create an ***object class*** to link the user interface and database:



Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Class***. Give the name ‘**stockItem**’.



Add lines of code to the stockItem class file to record the number of stock items, define the properties of the objects, and provide an array to link to the objects.

```
public class stockItem
{
    public static int stockCount = 0;
    public static stockItem[] stockObject = new stockItem[100];
    public static string databaseLocation =
                  "C:\\WEB APPLICATIONS\\hardwareStore.mdf;";
    public int productID { get; set; }
    public string category { get; set; }
    public string title { get; set; }
    public string description { get; set; }
    public double price { get; set; }
}
```

Add directives to the list at the start of the class file to include ***System.Data.SqlClient*** and ***System.Data***:

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

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

The next step is to add a method to upload a new stock item to the database table. This uses similar code to the Mountain Bike Club application in chapter 6 for uploading the photograph. The method will return a message to say whether or not the upload has been successful.

Please note that the instructions beginning:

public static string addItem(...

SqlConnection cnTB = new SqlConnection(...

string query = "INSERT INTO product(...

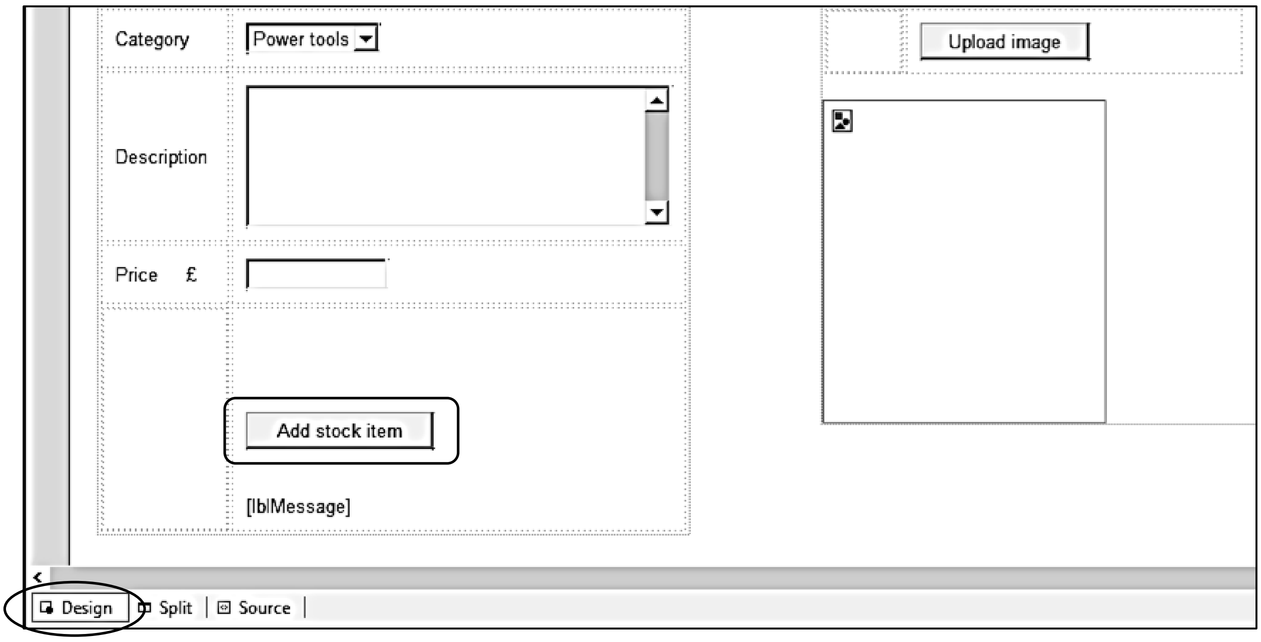
should each be entered as a single line of code without any line break.

```
public string description { get; set; }
public double price { get; set; }

public static string addItem(byte[] picbyte, string description,
                             string title, string category, double price)
{
    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 product(title,category,description,picture,price)
             VALUES ('"+ title + "','" + category + "','" + description +
            "', @pic ,'" + Convert.ToString(price) + "')";
        SqlCommand cmd = new SqlCommand(query, cnTB);
        cmd.Parameters.Add(picparameter);
        cmd.ExecuteNonQuery();
        cnTB.Close();
        message = "Record saved";
    }
    catch
    {
        message = "File error";
    }
    return message;
}
```

We can now return to the ***addStockItem.aspx*** page. We will need to add a ***button*** for uploading the stock item to the database, and also a ***label*** to display the status message from the upload method.

```
      <tr>
         <td>
            Price &nbsp;&nbsp;&nbsp;  £
          </td>
          <td>
           <asp:TextBox ID="txtPrice" runat="server" Width="100px"></asp:TextBox>
          </td>
      </tr>

       <tr>
         <td>
         </td>
         <td>
         <br /><br /><br /><br />
             <asp:Button ID="enter" runat="server" Text="Add stock item"
                 onclick="enter_Click"  />
             <br /> <br /> <br />
             <asp:Label ID="lblMessage" runat="server"></asp:Label>
         </td>
       </tr>

    </table>
  </div>

  <div id=imageEntry>
    <table border=0 cellpadding=10>
      <tr>
         <td>
            Image
          </td>
```

Change to the design view by clicking the ***Design*** button, then double click the ***Add stock item*** button to create an on_click method.



Scroll to the top of the ***addStockItem*** C# code. Add an array variable called ***picbyte*** at the start of the class. This will store the photograph in a digitised form, ready to upload to the database.

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

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

Move now to the ***upload_Click*** method. Add a line of code to transfer the photograph data into the ***picbyte*** array.

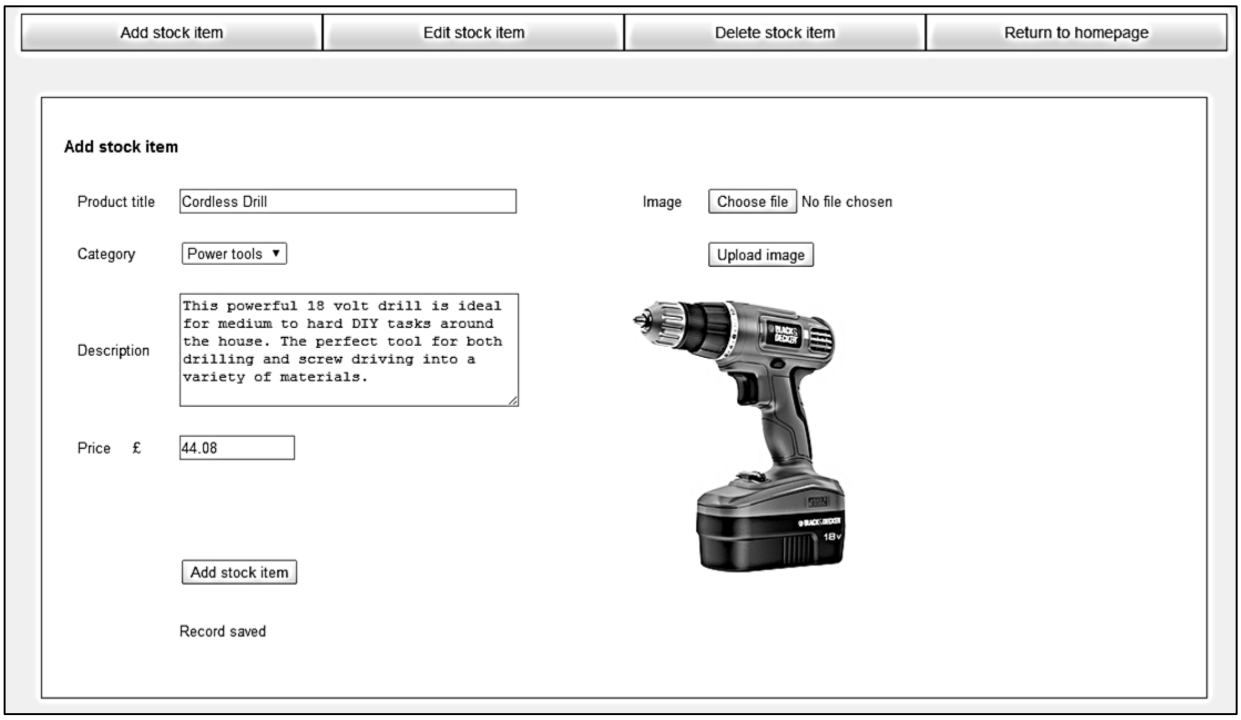
```
protected void upload_Click(object sender, EventArgs e)
{
    try
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(Server.MapPath("Images").ToString() + @"\" +
                      FileUpload1.FileName);
            Image1.ImageUrl = @"Images\" + FileUpload1.FileName;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

Finally, add lines of code to the ***enter_Click*** method. These collect the entries from the text boxes and pass them, along with the digitised photograph, to the ***addItem( )*** method in the ***stockItem*** class.

```
protected void enter_Click(object sender, EventArgs e)
{
    string description = txtDescription.Text;
    string title = txtTitle.Text;
    string category = lstCategory.Text;
    double price = Convert.ToDouble(txtPrice.Text);

    string message = stockItem.addItem(picbyte,description,title,category,price);

    lblMessage.Text = message;
}
```

After the ***addItem( )*** method is run, a message is returned which we display using the ***label*** component.

Build and run the staff website. Log-in and select the '***Add stock item***' option from the menu bar. Enter details of a hardware product, including a photograph, and click the '***Add stock item***' button.

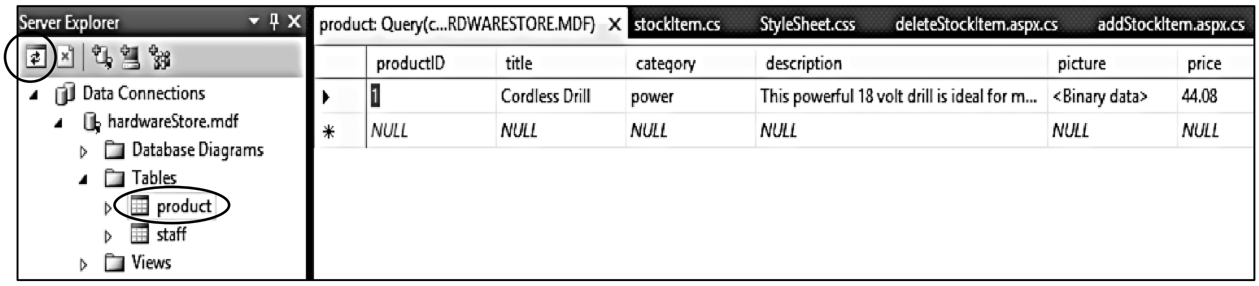


Add stock item

Product title	Cordless Drill	Image	No file chosen
Category	Power tools		
Description	This powerful 18 volt drill is ideal for medium to hard DIY tasks around the house. The perfect tool for both drilling and screw driving into a variety of materials.		
Price £	44.08		

Record saved

Close the browser and stop debugging. Go to the Server Explorer window and click the left icon to refresh the data. Right click the ***product*** table icon and select '***Show Table Data***'. Check that the information for the stock item has been stored in the table correctly. Note that the picture image just appears as a message: ***<Binary data>***

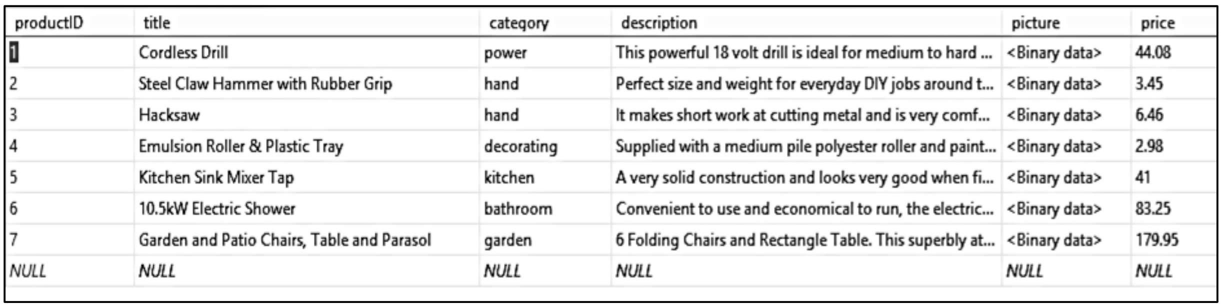


productID	title	category	description	picture	price
1	Cordless Drill	power	This powerful 18 volt drill is ideal for m...	<Binary data>	44.08
NULL	NULL	NULL	NULL	NULL	NULL

It will be convenient if the edit boxes and image are cleared after entering a stock item, ready for the form to be used to enter another item. Add lines to the ***enter_Click*** method of the ***addStockItem*** C# code page to do this:

```
    string category = lstCategory.Text;
    double price = Convert.ToDouble(txtPrice.Text);
    string message = stockItem.addItem(picbyte, description, title, category, price);
    lblMessage.Text = message;
    txtTitle.Text = "";
    txtDescription.Text = "";
    txtPrice.Text = "";
    Image1.ImageUrl = null;
  }
```

Run the staff website and add further test data for stock items. Include at least one item for each of the product categories. At the end of the data entry, close the browser and stop debugging. Go to the Server explorer window and refresh the data. Open the products table and ensure that items are shown correctly.



productID	title	category	description	picture	price
1	Cordless Drill	power	This powerful 18 volt drill is ideal for medium to hard ...	<Binary data>	44.08
2	Steel Claw Hammer with Rubber Grip	hand	Perfect size and weight for everyday DIY jobs around t...	<Binary data>	3.45
3	Hacksaw	hand	It makes short work at cutting metal and is very comf...	<Binary data>	6.46
4	Emulsion Roller & Plastic Tray	decorating	Supplied with a medium pile polyester roller and paint...	<Binary data>	2.98
5	Kitchen Sink Mixer Tap	kitchen	A very solid construction and looks very good when fi...	<Binary data>	41
6	10.5kW Electric Shower	bathroom	Convenient to use and economical to run, the electric...	<Binary data>	83.25
7	Garden and Patio Chairs, Table and Parasol	garden	6 Folding Chairs and Rectangle Table. This superbly at...	<Binary data>	179.95
NULL	NULL	NULL	NULL	NULL	NULL

We need to return to the public section of the website and complete the page to display details of the stock items for sale. Firstly, however, we must write a method within the ***stockItem.cs*** class file to create a set of ***objects*** to represent each of the stock items.

Open the ***stockItem.cs*** class file and create a new ***loadStock( )*** method below the list of properties. This will load the product records and store them temporarily in the ***dsStock*** dataset.

```
public string title { get; set; }
public string description { get; set; }
public double price { get; set; }

public static void loadStock()
{
    DataSet dsStock = new DataSet();

    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;
        cmStock.CommandText = "SELECT * FROM product";
        SqlDataAdapter daStock = new SqlDataAdapter(cmStock);
        daStock.Fill(dsStock);
        cnTB.Close();
    }
    catch
    {

    }
}
```

Add lines to the ***loadStock( )*** method to create the set of stock objects:

```
        cmStock.CommandText = "SELECT * FROM product";
        SqlDataAdapter daStock = new SqlDataAdapter(cmStock);
        daStock.Fill(dsStock);
        cnTB.Close();

        int countRecords = dsStock.Tables[0].Rows.Count;
        stockItem.stockCount = 0;
        for (int i = 0; i < countRecords; i++)
        {
            DataRow drStock = dsStock.Tables[0].Rows[i];
            int itemStockID = (int)drStock[0];
            string itemTitle = Convert.ToString(drStock[1]);
            string itemCategory = Convert.ToString(drStock[2]);
            string itemDescription = Convert.ToString(drStock[3]);
            double itemPrice = Convert.ToDouble(drStock[5]);

            stockItem.stockObject[stockItem.stockCount] = new stockItem();

            stockItem.stockObject[stockItem.stockCount].productID = itemStockID;
            stockItem.stockObject[stockItem.stockCount].category = itemCategory;
            stockItem.stockObject[stockItem.stockCount].title = itemTitle;
            stockItem.stockObject[stockItem.stockCount].description = itemDescription;
            stockItem.stockObject[stockItem.stockCount].price = itemPrice;
            stockItem.stockCount++;
        }
    }
    catch
    {
```

Go now to the ***displayItems.aspx*** page and add a ***label*** component to the '***content2***' section.

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

</asp:Content>
```

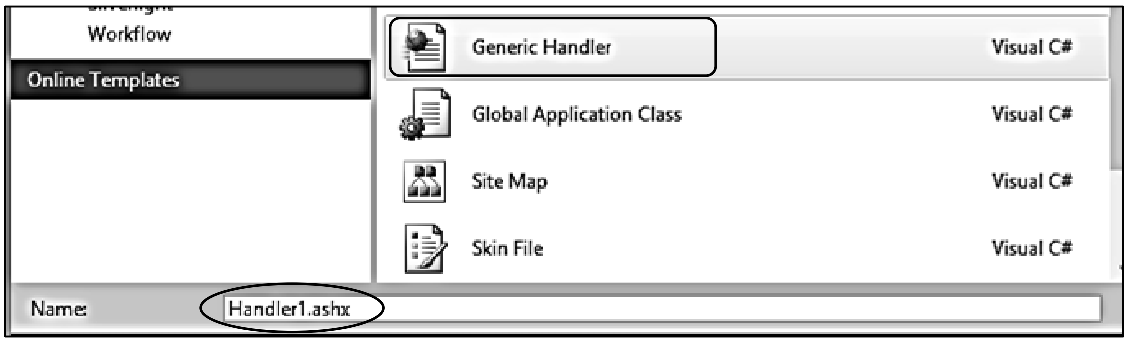
Open the C# code for the ***displayItems.aspx*** page and add lines Page_load to call the method to create stock objects from the database table:

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (stockItem.stockCount == 0)
        {
            stockItem.loadStock();
        }
    }
```

Once the stock item objects have been created, we can search for items in the correct product category and display these on the web page. Notice that we are building up lines of HTML code to create output through the label component, using a similar technique to the ***Bus Timetable*** project in chapter 4.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (stockItem.stockCount == 0)
    {
        stockItem.loadStock();
    }

    string productCategory = Request.QueryString["category"];
    string s = "";
    s = s += "<table cellpadding=8 cellspacing =8 border=0>";
    int photoID;

    for (int i = 0; i < stockItem.stockCount; i++)
    {
        if (productCategory == stockItem.stockObject[i].category)
        {
            s += "<tr>";
            s += "<td border=1 bgcolor=white>";
            s += "<h3>" + stockItem.stockObject[i].title + "</h3>";
            s += stockItem.stockObject[i].description + "<br>";
            double price = stockItem.stockObject[i].price;
            string priceString = String.Format("{0:.##}", price);
            s += "<h3>£" + priceString + "</h3>";
            s += "<td border=1 bgcolor=white>";
            photoID = stockItem.stockObject[i].productID;
            s += "<img src='Handler1.ashx?imgid=" + photoID +
            "' width='200' border='0' >";
            s += "</td>";
            s += "</tr>";
        }
    }
    s += "</table>";
    Label1.Text = s;

}
```

You may remember from the ***Mountain Bike Club*** project in chapter 6 that a ***Handler*** file is needed to load the digitised images and convert them into .JPG format for display on the web page. We will set up the Handler file now. Go to the Solution Explorer window and right click the **Hardware store** project icon. Select ***Add / New Item*** and choose ***Generic Handler***. Accept the name '**Handler1**'.



Name: Handler1.ashx

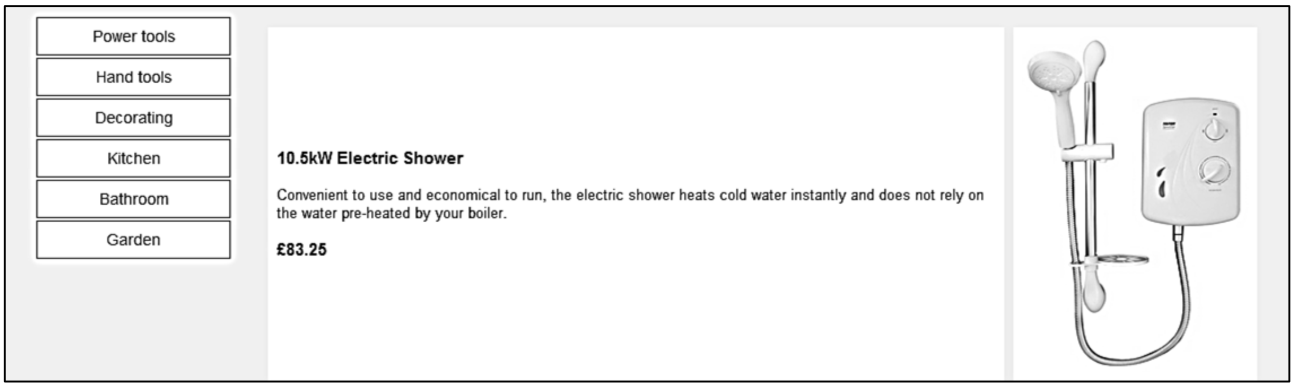
Open the Handler file and replace the code in the ProcessRequest( ) method with the lines shown below . Note that the commands beginning:

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

should be entered as single lines without line breaks.

```
public class Handler1 : IHttpHandler
{
 string databaseLocation = "C:\\WEB APPLICATIONS \\hardwareStore.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 picture FROM product
              WHERE productID=" + context.Request.QueryString["imgid"], conn);
        conn.Open();
        rdr = sqlcmd.ExecuteReader();
        while (rdr.Read())
        {
            context.Response.ContentType = "image/jpg";
            context.Response.BinaryWrite((byte[])rdr["picture"]);
        }
        if (rdr != null)
        rdr.Close();
    }
    finally
    {
        if (conn != null)
            conn.Close();
    }
 }
```

Build and run the web site Home Page. Check that all product categories are displayed correctly.



10.5kW Electric Shower

Convenient to use and economical to run, the electric shower heats cold water instantly and does not rely on the water pre-heated by your boiler.

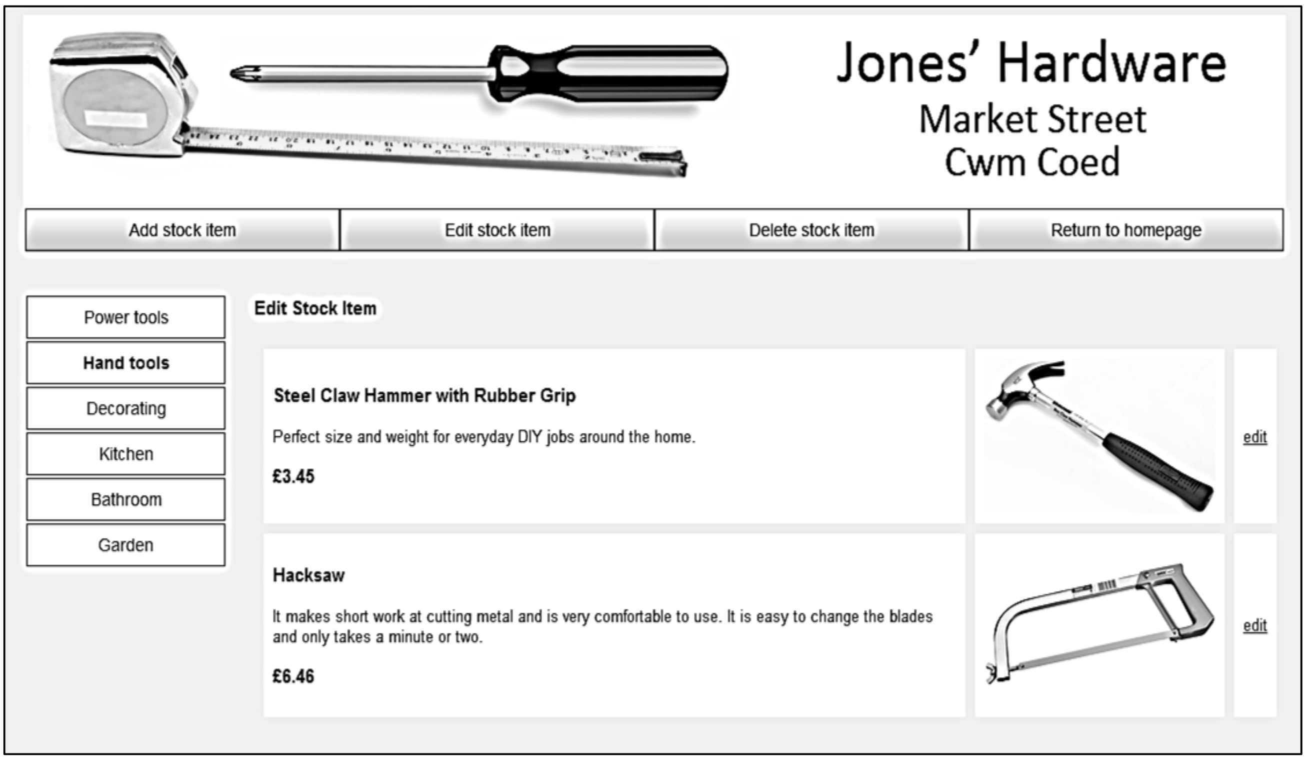
£83.25

Close the web browser and stop debugging.

It just remains to complete the ***Edit Item*** and ***Delete Item*** functions of the staff section. Go to the ***editStockItem.aspx*** page and add code to the '***Content2***' section to produce a menu of product categories.

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

  <div id="products">
        <ul id="cat">
            <li><a href="editStockItem.aspx?category=power">Power tools</a></li>
            <li><a href="editStockItem.aspx?category=hand">Hand tools</a></li>
            <li><a href="editStockItem.aspx?category=decorating">Decorating</a></li>
            <li><a href="editStockItem.aspx?category=kitchen">Kitchen</a></li>
            <li><a href="editStockItem.aspx?category=bathroom">Bathroom</a></li>
            <li><a href="editStockItem.aspx?category=garden">Garden</a></li>
            </ul>
   </div>
   <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

</asp:Content>
```

Open the C# code page for ***editStockItem.aspx*** and add lines to the ***Page_load( )*** method as shown on the next page. This section is almost identical to the product display of the public web page. The only addition is an ***edit*** link provided alongside each item. You may find it quickest to copy and paste the code and make the few changes necessary.



Steel Claw Hammer with Rubber Grip

Perfect size and weight for everyday DIY jobs around the home.

£3.45

edit

Hacksaw

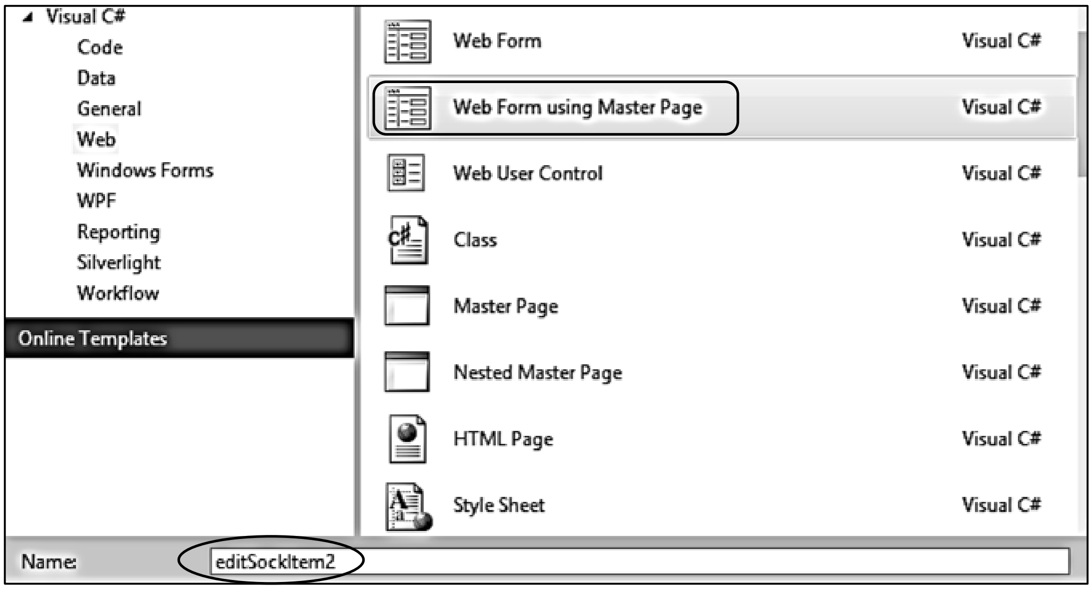
It makes short work at cutting metal and is very comfortable to use. It is easy to change the blades and only takes a minute or two.

£6.46

edit

```
public partial class editStockItem : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (staff.login == false)
        {
            Response.Redirect("staffLogin.aspx");
        }

        if (stockItem.stockCount == 0)
        {
            stockItem.loadStock();
        }

        string productCategory = Request.QueryString["category"];

        string s = "";
        s += "<h3>Edit Stock Item</h3>";
        s = s += "<table cellpadding=8 cellspacing =8 border=0>";
        int photoID;

        for (int i = 0; i < stockItem.stockCount; i++)
        {
            if (productCategory == stockItem.stockObject[i].category)
            {
                s += "<tr>";
                s += "<td border=1 bgcolor=white>";
                s += "<h3>" + stockItem.stockObject[i].title + "</h3>";
                s += stockItem.stockObject[i].description + "<br>";
                double price = stockItem.stockObject[i].price;
                string priceString = String.Format("{0:.##}", price);
                s += "<h3>£" + priceString + "</h3>";
                s += "<td border=1 bgcolor=white>";
                photoID = stockItem.stockObject[i].productID;
                s += "<img src='Handler1.ashx?imgid=" + photoID +
                "' width='200' border='0' >";
                s += "</td>";
                s += "<td border=1 bgcolor=white>";
                s += "<a href='editStockItem2.aspx?stockID=";
                s += stockItem.stockObject[i].productID;
                s += "'> edit </a>";
                s += "</td>";

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

    }
}
```

Return to the Solution Explorer window. Select ***Add / New Item*** and choose ***Web Form using Master Page***. Give the name '**editStockItem2**'.



Name: editSockItem2

Select '***staffSite.Master***' as the master file.

The page for editing a stock record will be very similar to the entry page for a new stock item. Open ***editStockItem2.aspx*** and enter the lines shown below. You may find it easiest to copy the code from the ***addStockItem.aspx*** page and just make the necessary changes.

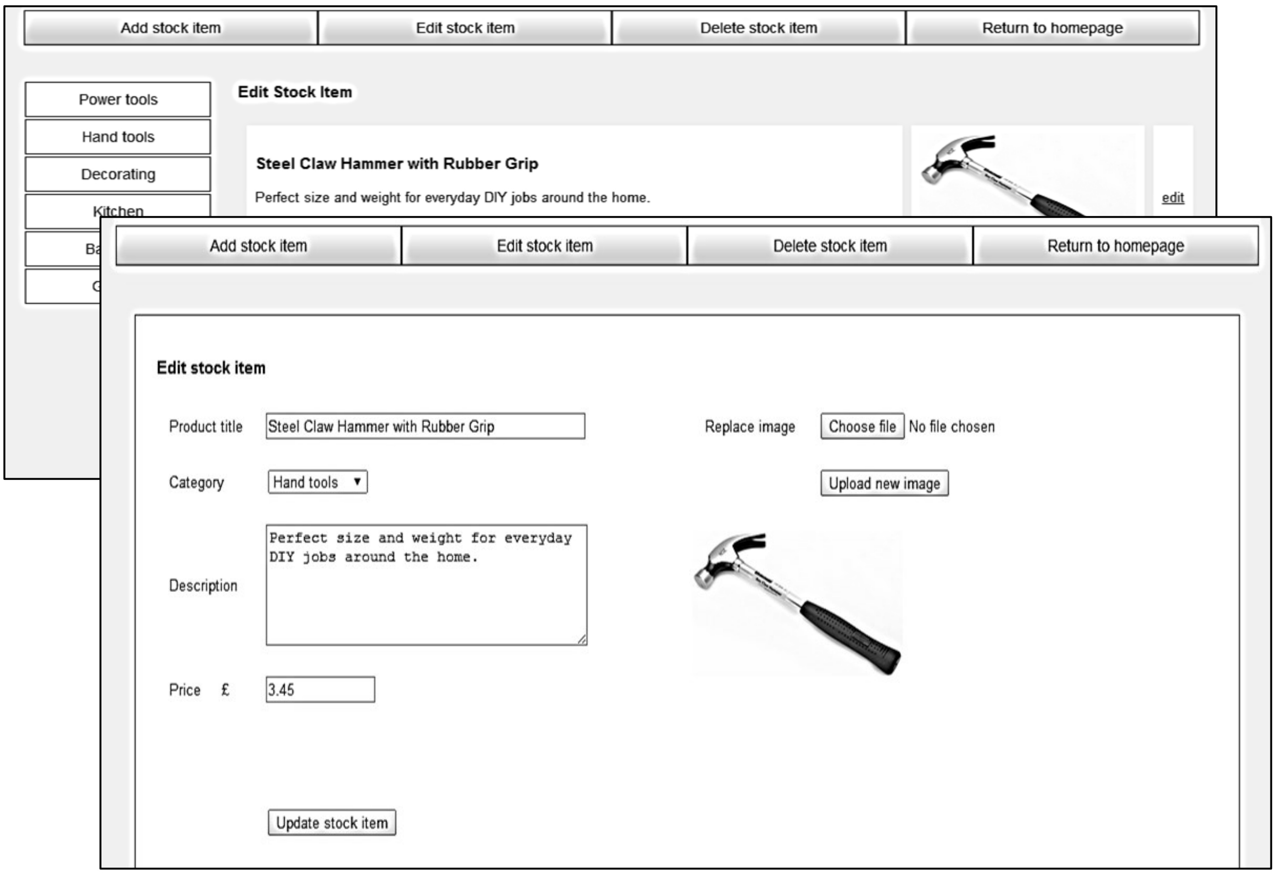
```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id=entryForm>
       <h3>Edit stock item</h3>
       <div id="itemEntry" >
        <table border="0" cellpadding="10">
          <tr>
             <td>
                Product title
              </td>
              <td>
                <asp:TextBox
                      ID="txtTitle" runat="server" Width="300px"></asp:TextBox>
              </td>
          </tr>
          <tr>
             <td>
               Category
              </td>
              <td>
                  <asp:DropDownList ID="lstCategory" runat="server">
                      <asp:ListItem Value="power">Power tools</asp:ListItem>
                      <asp:ListItem Value="hand">Hand tools</asp:ListItem>
                      <asp:ListItem Value="decorating">Decorating</asp:ListItem>
                      <asp:ListItem Value="kitchen">Kitchen</asp:ListItem>
                      <asp:ListItem Value="bathroom">Bathroom</asp:ListItem>
                      <asp:ListItem Value="garden">Garden</asp:ListItem>
                  </asp:DropDownList>
```

```
      </td>
         </tr>
          <tr>
            <td>
               Description
             </td>
             <td>
               <asp:TextBox ID="txtDescription" runat="server" Width="300px"
                                       TextMode="MultiLine" Rows="6"></asp:TextBox>
             </td>
         </tr>
         <tr>
            <td>
               Price &nbsp;&nbsp;&nbsp;  £
             </td>
             <td>
              <asp:TextBox ID="txtPrice" runat="server" Width="100px"></asp:TextBox>
             </td>
         </tr>
         <tr>
            <td></td>
            <td>
                <br /><br /><br /><br />
                <asp:Button ID="enter" runat="server" Text="Update stock item" />
                <br /><br /><br />
                <asp:Label ID="lblMessage" runat="server"></asp:Label>
            </td>
          </tr>
       </table>
      </div>
      <div id=imageEntry>
       <table border=0 cellpadding=10>
         <tr>
            <td>
                Replace image
             </td>
              <td>
               <asp:FileUpload ID="FileUpload1" runat="server" />
             </td>
             </tr>
             <tr>
             <td></td>
            <td>
               <asp:Button ID="upload" runat="server" Text="Upload new image"/>
             </td>
           </tr>
         </table>
           <br />
          <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
      </div>
  </div>
</asp:Content>
```

Open the C# code page for ***editStockItem2.aspx***. Add the variables ***picbyte*** and ***productIDwanted*** at the start of the class, and add lines of code to the ***Page_Load( )*** method.

```
public partial class editStockItem2 : System.Web.UI.Page
{
    public static byte[] picbyte;
    public static int productIDwanted;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (staff.login == false)
        {
            Response.Redirect("staffLogin.aspx");
        }
        string itemCategory = "";
        string itemTitle = "";
        string itemDescription = "";
        double itemPrice = 0;
        if (txtTitle.Text == "")
        {
            productIDwanted = Convert.ToInt16(Request.QueryString["stockID"]);
            for (int i = 0; i < stockItem.stockCount; i++)
            {
                if (stockItem.stockObject[i].productID == productIDwanted)
                {
                    itemCategory = stockItem.stockObject[i].category;
                    itemTitle = stockItem.stockObject[i].title;
                    itemDescription = stockItem.stockObject[i].description;
                    itemPrice = stockItem.stockObject[i].price;
                }
            }
            txtTitle.Text = itemTitle;
            for (int j = 0; j <= 5; j++)
            {
                if (lstCategory.Items[j].Value == itemCategory)
                {
                    lstCategory.Items[j].Selected = true;
                }
            }
            txtDescription.Text = itemDescription;
            string priceString = String.Format("{0:.##}", itemPrice);
            txtPrice.Text = Convert.ToString(priceString);
            string s = "";
            s += "<img src='Handler1.ashx?imgid=" + productIDwanted +
            "' width='200' border='0' >";
            Label1.Text = s;
        }

    }
}
```

Build and run the staff website. Log-in, select '***Edit stock item***', then select a product category. Choose a stock item, then click the '***edit***' link. The ***editStockItem2.aspx*** page should open, with the data for the product displayed.



Edit stock item

Product title: Steel Claw Hammer with Rubber Grip

Category: Hand tools

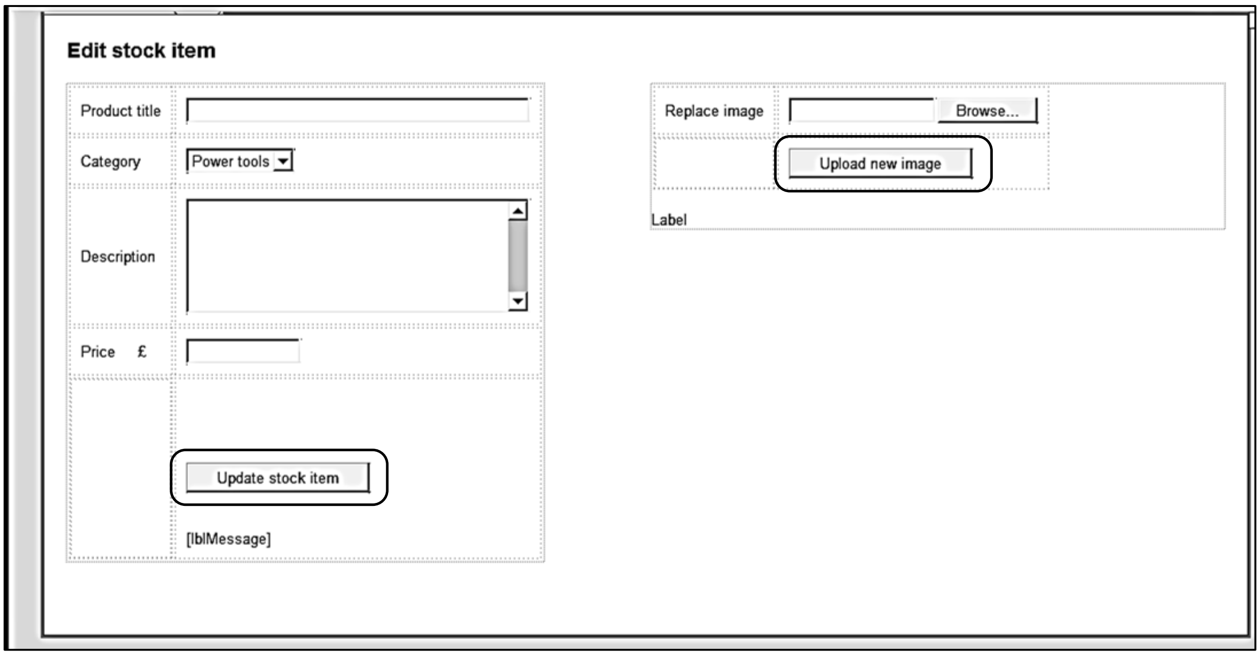
Description: Perfect size and weight for everyday DIY jobs around the home.

Price £: 3.45

Replace image: No file chosen

Close the browser and stop debugging.

Open ***editStockItem2.aspx*** and go to the design screen by clicking the ***Design*** button at the bottom left of the window.
Double click the '***Upload new image***' and '***Update stock item***' buttons to create button_click methods.



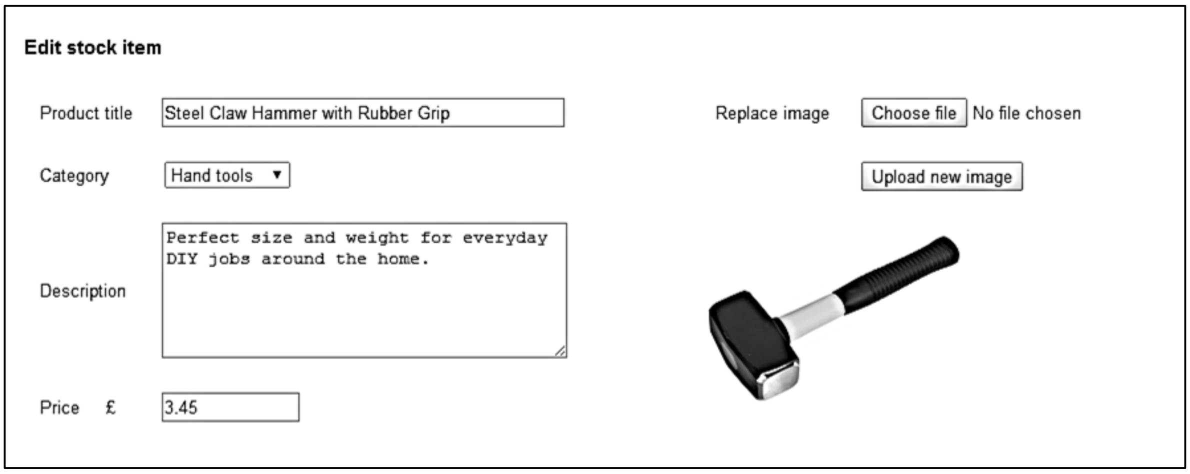
When we update a product record, it will be useful to know whether or not the photograph image needs to be changed in addition to the text data. Go to the ***stockItem.cs*** class file and add the boolean variable ***newPicture***; this will be set to **true** if the photograph needs to be updated:

```
public class stockItem
{
    public static int stockCount = 0;
    public static stockItem[] stockObject = new stockItem[100];
    public static string databaseLocation =
                "C:\\WEB APPLICATIONS\\hardwareStore.mdf;";

    public static bool newPicture;

    public int productID { get; set; }
    public string category { get; set; }
```

Return to the ***editStockItem2.aspx*** C# code page. Go to the end of the ***Page_Load( )*** method and add a line to set the value of ***newPicture*** to **false**.

```
    protected void Page_Load(object sender, EventArgs e)
    {
            . . . .
            txtPrice.Text = Convert.ToString(priceString);
            string s = "";
            s += "<img src='Handler1.ashx?imgid=" + productIDwanted +
            "' width='200' border='0' >";
            Label1.Text = s;
            stockItem.newPicture = false;
        }
    }
```

We will now add lines of code to the ***upload_Click( )*** method of ***editStockItem2.aspx***, as shown on the next page. When you have added this code, build and run the staff website. Select an item for editing. Choose a new image file using the ***Browse*** button, then click '***Upload new image***'. Check that the selected image is displayed.



```
protected void upload_Click(object sender, EventArgs e)
{
    lblMessage.Text = "";
    try
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(Server.MapPath("Images").ToString() + @"\" +
                  FileUpload1.FileName);
            string s = "<img src='Images/" + FileUpload1.FileName +
                  "' width='200' border='0'>";
            Label1.Text = s;
            stockItem.newPicture = true;
            picbyte = FileUpload1.FileBytes;
        }
        else
        {
            lblMessage.Text = "Please load an image";
        }
    }
    catch
    {

    }
}
```

Close the browser and stop debugging.

When the product record has been edited on screen, it will be necessary to update the record in the database. A method will be required to do this. Open the ***stockItem.cs*** class file and add an ***updateItem( )*** method. Note that the lines of code beginning:

public static string updateItem(. . .
SqlConnection cnTB = new SqlConnection(. . .
query = "UPDATE product SET title='" + . . .

should be entered as single commands with no line breaks.

```
public string description { get; set; }
public double price { get; set; }

public static string updateItem(int productID, byte[] picbyte, string description,
                    string title, string category, double price)
{
    string message;
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation +"Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
```

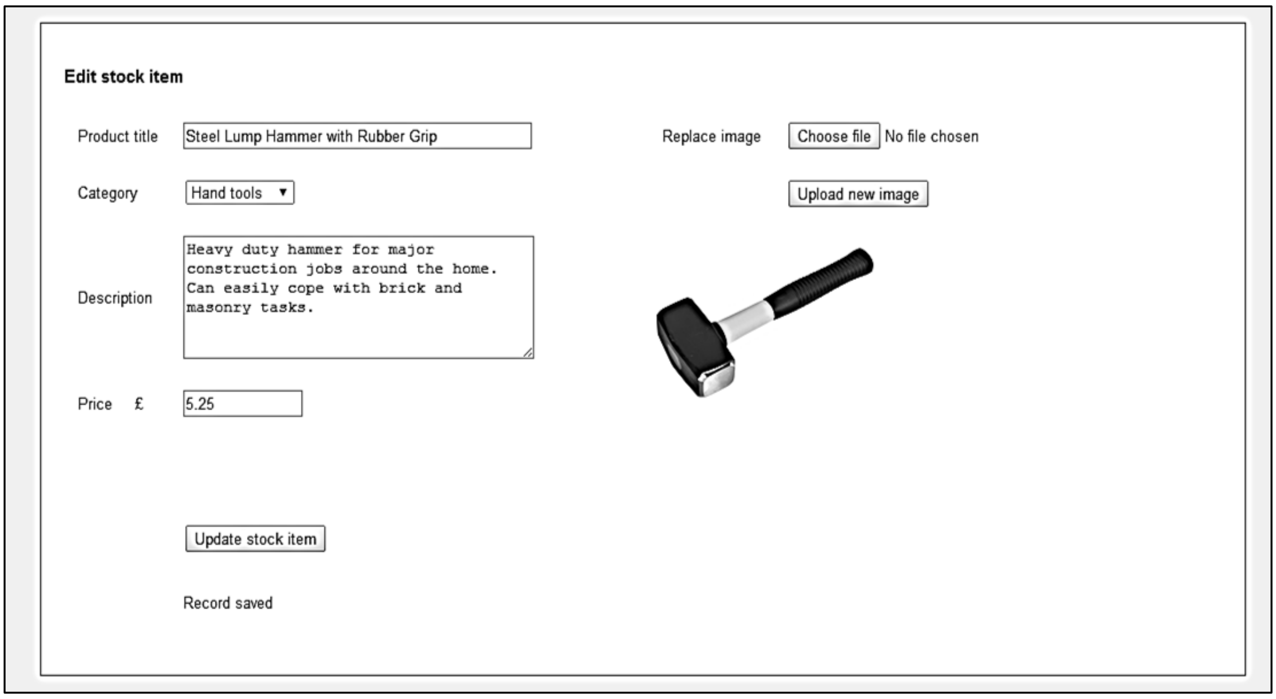
```
            SqlCommand cmStock = new SqlCommand();
            cmStock.Connection = cnTB;
            cmStock.CommandType = CommandType.Text;
            string query;
            if (newPicture == true)
            {
               query = "UPDATE product SET title='" + title + "', category='" +
                   category + "', description='" + description + "', picture =@pic,
                   price='" + Convert.ToString(price) + "' WHERE productID='" +
                   productID + "'";
            }
            else
            {
                query = "UPDATE product SET title='" + title + "', category='" +
                   category + "', description='" + description + "', price='" +
                   Convert.ToString(price) + "' WHERE productID='" + productID + "'";
            }
            SqlCommand cmd = new SqlCommand(query, cnTB);
            if (newPicture == true)
            {
                SqlParameter picparameter = new SqlParameter();
                picparameter.SqlDbType = SqlDbType.Image;
                picparameter.ParameterName = "pic";
                picparameter.Value = picbyte;
                cmd.Parameters.Add(picparameter);
            }
            cmd.ExecuteNonQuery();
            cnTB.Close();
            message = "Record saved";
        }
        catch
        {
            message = "File error";
        }
        return message;
    }

    public static void loadStock()
    {
        DataSet dsStock = new DataSet();
```

Notice that two different SQL queries are provided, depending on whether or not the picture image is to be updated.

Return to the ***editStockItem2.aspx*** C# code page and add lines to the ***enter_Click( )*** method as shown below. These collect the required data items from the text boxes, then call the ***updateItem( )*** method to amend the product record in the database. We finally call the ***loadStock( )*** method to create an updated set of stock item objects.

```
protected void enter_Click(object sender, EventArgs e)
{
    string description = txtDescription.Text;
    string title = txtTitle.Text;
    string category = lstCategory.Text;
    double price = Convert.ToDouble(txtPrice.Text);
    string message = stockItem.updateItem(productIDwanted, picbyte,
                  description, title, category, price);
    lblMessage.Text = message;
    stockItem.loadStock();
}
```

Build and run the staff website. Choose the '***Edit stock item***' option, then select a product to edit. Make changes to each of the data fields and load a new photograph image. Click the '***Update stock item***' button. The message '***Record saved***' should appear.

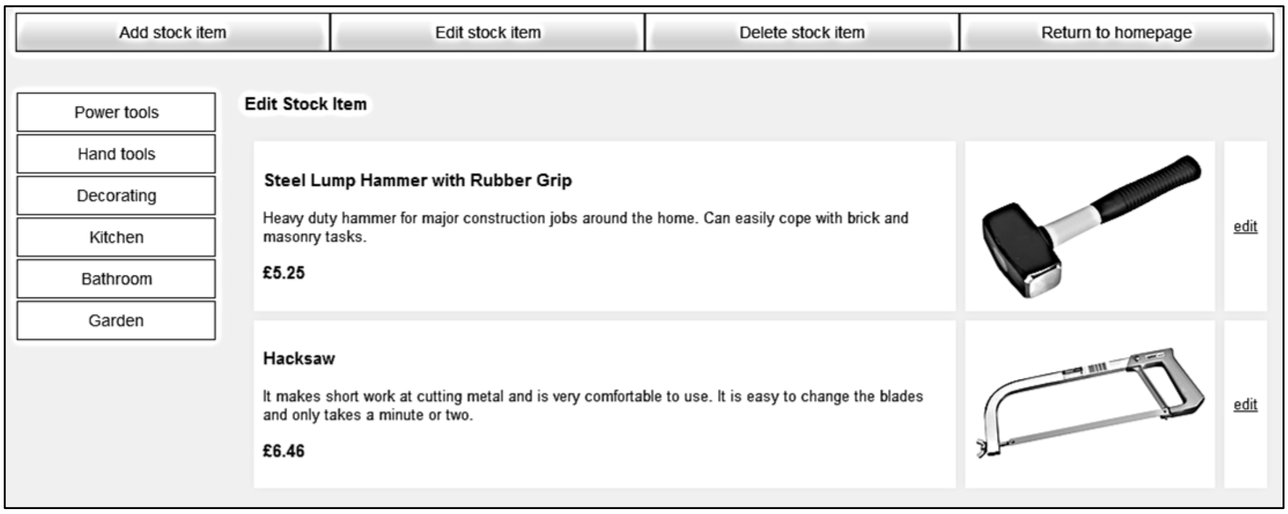


Edit stock item

Product title	Steel Lump Hammer with Rubber Grip
Category	Hand tools
Description	Heavy duty hammer for major construction jobs around the home. Can easily cope with brick and masonry tasks.
Price £	5.25
Replace image	No file chosen

Record saved

Reselect the '***Edit stock item***' menu option and check that the record has been correctly updated.



Edit Stock Item

Steel Lump Hammer with Rubber Grip

Heavy duty hammer for major construction jobs around the home. Can easily cope with brick and masonry tasks.

£5.25

Hacksaw

It makes short work at cutting metal and is very comfortable to use. It is easy to change the blades and only takes a minute or two.

£6.46

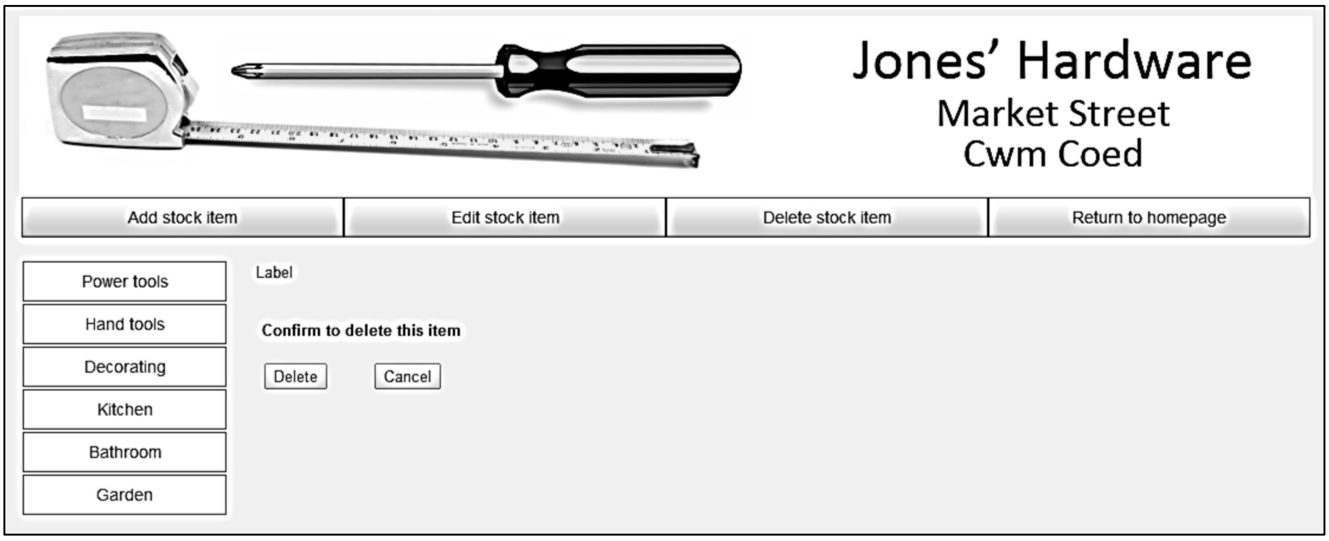
Close the browser and stop debugging.

The final menu option to complete is '***Delete stock item***'. Open the ***deleteStockItem.aspx*** file and add lines of code to the 'Content2' section:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  <div id="products">
      <ul id="cat">
        <li><a href="deleteStockItem.aspx?category=power">Power tools</a></li>
        <li><a href="deleteStockItem.aspx?category=hand">Hand tools</a></li>
        <li><a href="deleteStockItem.aspx?category=decorating">Decorating</a></li>
        <li><a href="deleteStockItem.aspx?category=kitchen">Kitchen</a></li>
        <li><a href="deleteStockItem.aspx?category=bathroom">Bathroom</a></li>
        <li><a href="deleteStockItem.aspx?category=garden">Garden</a></li>
      </ul>
   </div>
   <div id="stockItems">
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
      <br /><br />
      <asp:Panel ID="Panel1" runat="server" Height="72px" style="margin-left:205px">
           <h4>
               Confirm to delete this item
           </h4>
           <asp:Button ID="btnDelete" runat="server" Text="Delete"  />
           &nbsp; &nbsp; &nbsp; &nbsp;
           <asp:Button ID="btnCancel" runat="server" Text="Cancel"  />
       </asp:Panel>
       <br />
   </div>
</asp:Content>
```

Build and run the staff website. Select the '***Delete stock item***' option to examine the page layout so far. As on other pages, we will use a side menu for displaying products.

A ***Panel*** component has been created, which carries the message '***Confirm to delete this item***', plus ***Delete*** and ***Cancel*** buttons. Initially this panel will not be visible. When the user clicks to delete a selected item, the panel will appear.



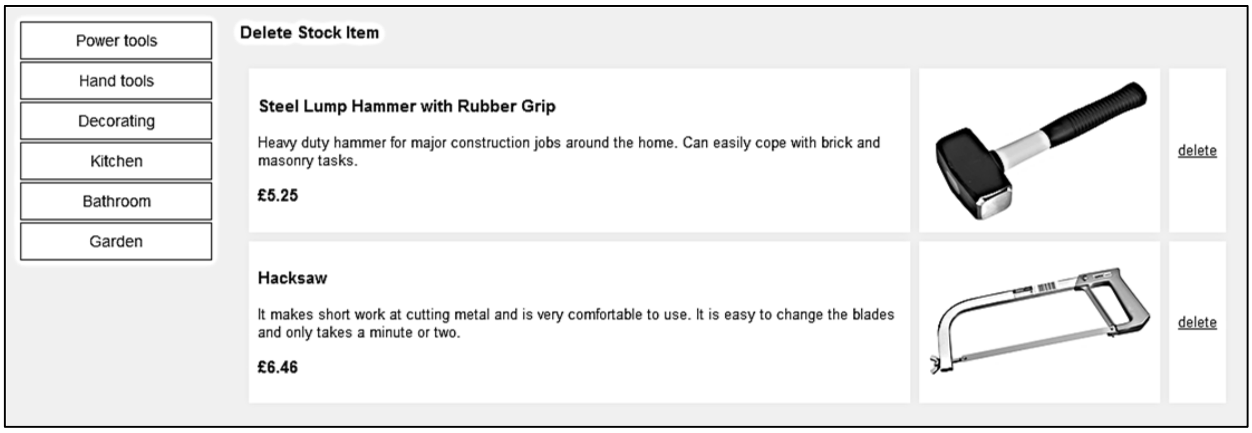
Close the browser and stop debugging.

Return to ***deleteStockItem.aspx*** and open the C# code page. Add lines of program as shown below:

```
public partial class deleteStockItem : System.Web.UI.Page
{
    int productIDwanted;
    string productCategory;

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

        if (stockItem.stockCount == 0)
        {
            stockItem.loadStock();
        }
        Panel1.Visible = false;
        productCategory = Request.QueryString["category"];
        productIDwanted = Convert.ToInt16(Request.QueryString["stockID"]);
        string s = "";
        s += "<h3>Delete Stock Item</h3>";
        s = s += "<table cellpadding=8 cellspacing =8 border=0>";
        int photoID;
        if (productIDwanted == 0)
        {

            for (int i = 0; i < stockItem.stockCount; i++)
            {
                if (productCategory == stockItem.stockObject[i].category)
                {
                    s += "<tr>";
                    s += "<td border=1 bgcolor=white>";
                    s += "<h3>" + stockItem.stockObject[i].title + "</h3>";
                    s += stockItem.stockObject[i].description + "<br>";
                    double price = stockItem.stockObject[i].price;
                    string priceString = String.Format("{0:.##}", price);
                    s += "<h3>£" + priceString + "</h3>";
                    s += "<td border=1 bgcolor=white>";
                    photoID = stockItem.stockObject[i].productID;
                    s += "<img src='Handler1.ashx?imgid=" + photoID +
                    "' width='200' border='0' >";
                    s += "</td>";
                    s += "<td border=1 bgcolor=white>";
                    s += "<a href='deleteStockItem.aspx?category=" +
                                            productCategory + "&stockID=";
                    s += stockItem.stockObject[i].productID;
                    s += "'> delete </a>";
                    s += "</td>";
                    s += "</tr>";
                }
            }
            s += "</table>";
        }
        else
        {
            for (int i = 0; i < stockItem.stockCount; i++)
            {
                if (productIDwanted == stockItem.stockObject[i].productID)
                {
                    s += "<tr>";
                    s += "<td border=1 bgcolor=white>";
                    s += "<h3>" + stockItem.stockObject[i].title + "</h3>";
                    s += stockItem.stockObject[i].description + "<br>";
                    double price = stockItem.stockObject[i].price;
                    string priceString = String.Format("{0:.##}", price);
                    s += "<h3>£" + priceString + "</h3>";
                    s += "<td border=1 bgcolor=white>";
                    photoID = stockItem.stockObject[i].productID;
                    s += "<img src='Handler1.ashx?imgid=" + photoID +
                    "' width='200' border='0' >";
                    s += "</td>";
                    s += "</tr>";
                }
            }
            s += "</table>";
        }
        Label1.Text = s;
        if (productIDwanted > 0)
        {
            Panel1.Visible = true;
        }

    }
}
```

This method is carrying out a series of tasks:

- When the page is first loaded, only the side menu of product categories is displayed.
- On selecting a product category, the page is reloaded, but this time the category name is transferred back as part of the page URL. Products in the selected category are now displayed, along with a **delete** option. This is very similar to the ***edit items*** page.



Steel Lump Hammer with Rubber Grip

Heavy duty hammer for major construction jobs around the home. Can easily cope with brick and masonry tasks.

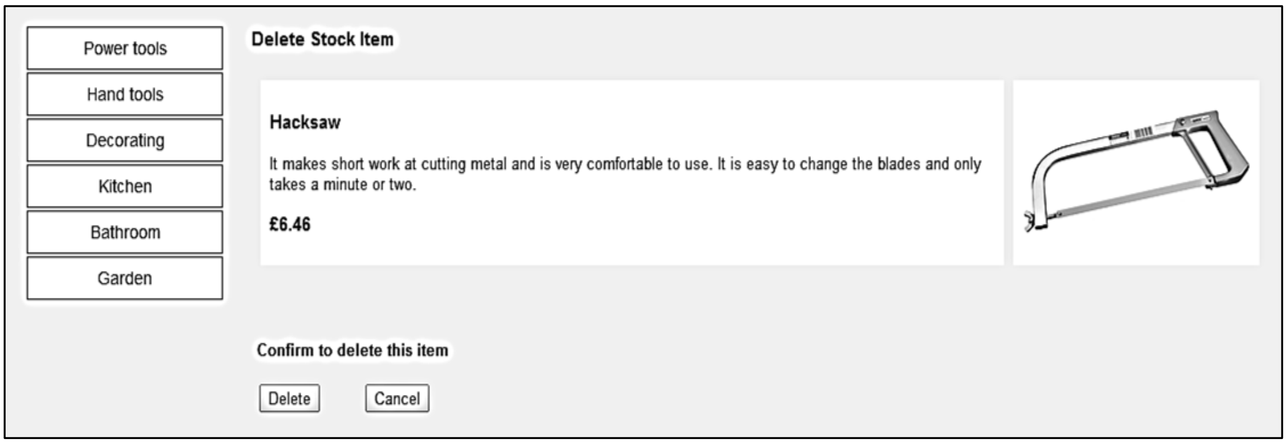
£5.25

Hacksaw

It makes short work at cutting metal and is very comfortable to use. It is easy to change the blades and only takes a minute or two.

£6.46

- If the user clicks to delete an item, the page is again reloaded, but this time the ***productID*** selected for deletion is transferred back as part of the page URL. The program displays only the details for this item, and the '***confirm***' panel becomes visible.



Hacksaw

It makes short work at cutting metal and is very comfortable to use. It is easy to change the blades and only takes a minute or two.

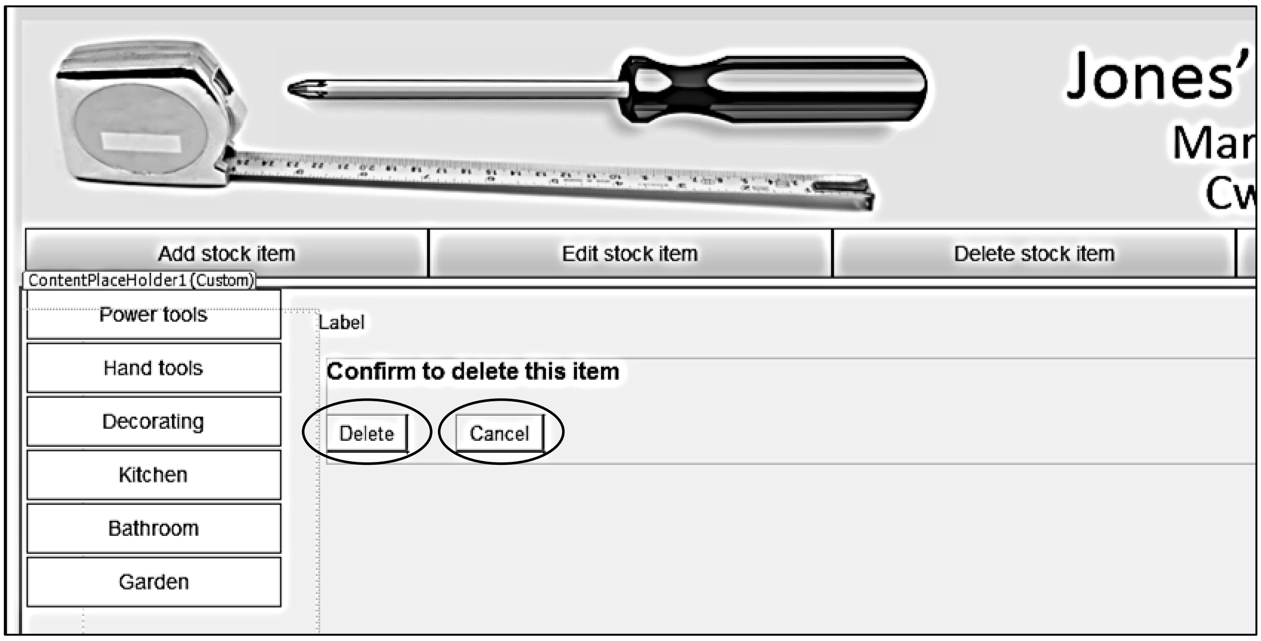
£6.46

Build and run the staff website. Test the '***Delete stock item***' option to check that the screen display operates correctly.

Close the web browser and stop debugging. It simply remains to delete the selected record from the database. Open the ***stockItem.cs*** file to add a method to carry out the deletion.

```
public string description { get; set; }
public double price { get; set; }

public static string deleteItem(int productIDwanted)
{
    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 product WHERE productID='" + productIDwanted + "'";
        message = query;
        SqlCommand cmd = new SqlCommand(query, cnTB);
        cmd.ExecuteNonQuery();
        cnTB.Close();
    }
    catch
    {

    }
    return message;
}
```

Return to ***deleteStockItem.aspx*** and change to the design view by means of the ***Design*** button. Double click the ***Delete*** and ***Cancel*** buttons to create ***button_click*** methods.



Move to the C# code page and add lines to these methods:

```
protected void btnDelete_Click(object sender, EventArgs e)
{
    string message = stockItem.deleteItem(productIDwanted);
    stockItem.loadStock();
    Response.Redirect("deleteStockItem.aspx?category=" + productCategory +
                                                         "&stockID=0");
}

protected void btnCancel_Click(object sender, EventArgs e)
{
    Response.Redirect("deleteStockItem.aspx?category=" + productCategory +
                                                         "&stockID=0");
}
```

The '***Delete***' button calls the ***deleteItem( )*** method and relaods the updated set of stock item objects, whilst the '***Cancel***' button simply re-runs the web page without carrying out any database operations.

The Hardware Store project is now completed. Carefully test both the public and staff web sites to check that all menu options and data operations are working correctly.