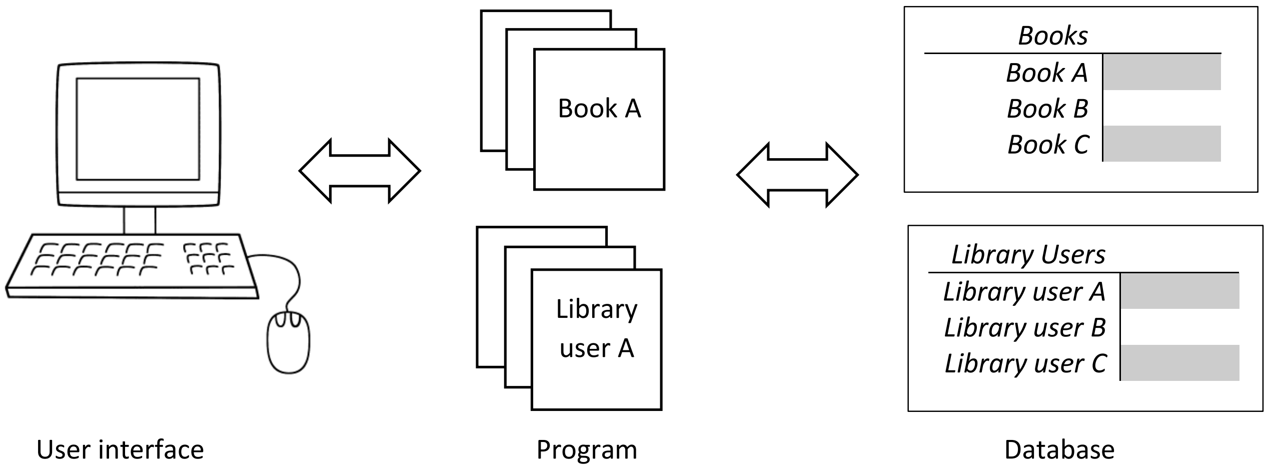
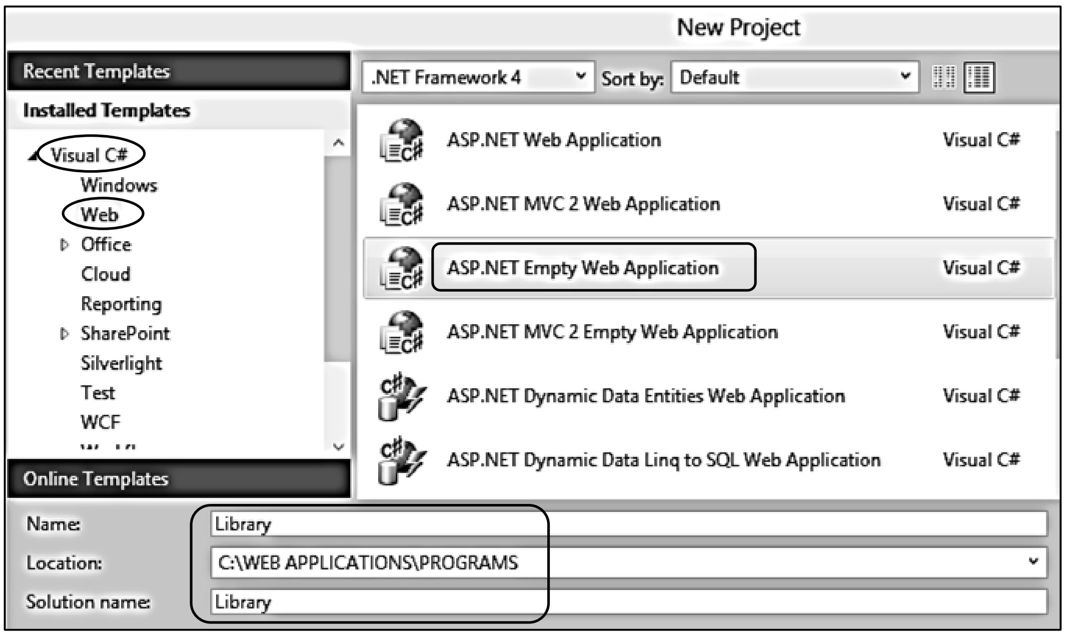
8 Library loan system

In previous programs in this book, we have taken a traditional procedural approach in transferring data directly between web pages and the ASP database. A better approach for large projects is to handle data within the program as ***objects*** which represent real word entities.

In this example program, we will create a simple system for recording book loans from a library. This will require two types of real world object, ***books*** and ***library users***. The objects may be created either by loading data from a database table by inputting data directly from the web page or, and objects can in turn be displayed on screen or saved back to the database.

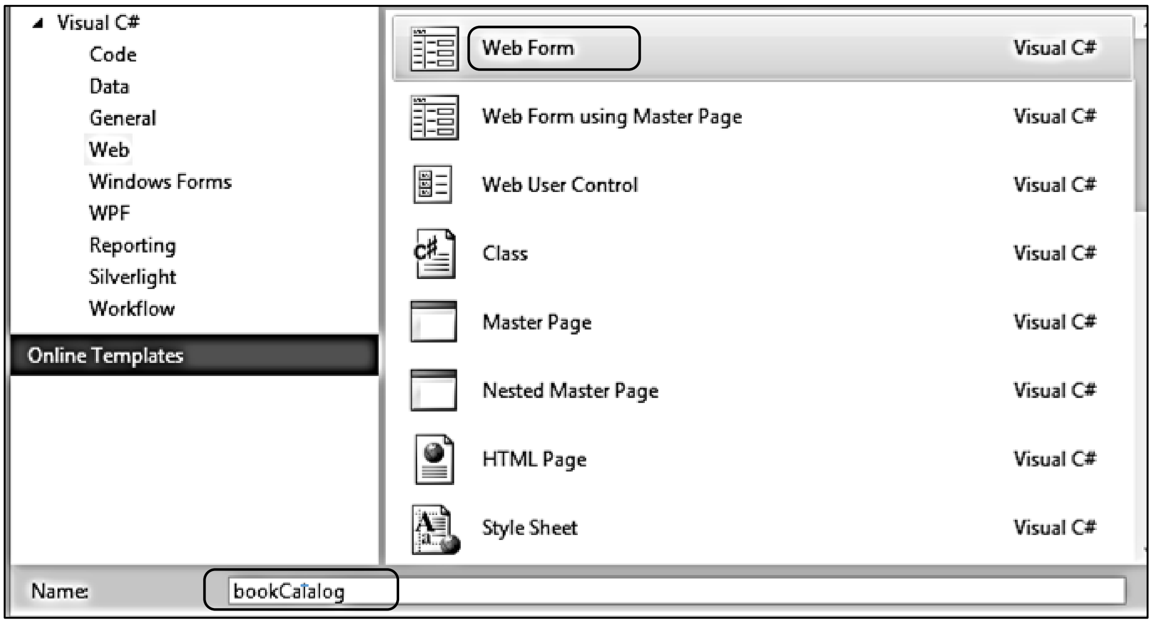


Begin by opening ***Visual Studio*** and clicking ***New Project***. Select ***Visual C# Web*** as the project type, then ***ASP.NET Empty Web Application***. Enter the name Library for the project, and select a location where it will be stored.


New 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: Library
Location: C:\WEB APPLICATIONS\PROGRAMS
Solution name: Library

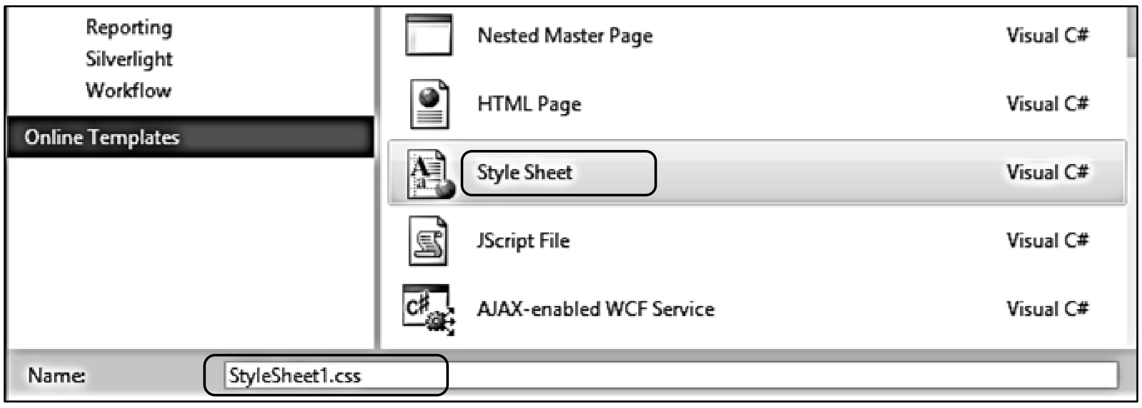
Go to the Solution Explorer window and right click the Library program icon. Select ***Add / New Item***, then choose ***Web Form***. Give the name '**bookCatalog**'.



Open the ***bookCatalog.aspx*** code page and insert the word '**Library**' within the ***<title>*** tags. Add the ID '**content**' to the ***<div>*** tag, and insert a heading '**Library Catalogue**' inside **<h3>** tags.

```
<head runat="server">
    <title>Library</title>
</head>
<body>
    <form id="form1" runat="server">
      <div id="content">
         <h3>Book Catalogue</h3>
      </div>
    </form>
</body>
</html>
```

We will next add some formatting to the page by means of a style sheet. Go to the ***Solution Explorer*** window and right click the ***Library*** program icon. Select ***Add / New Item***, then choose ***Style Sheet***. Accept the name **StyleSheet1**.



Open the style sheet and add formatting code for the ***body*** and ***content*** sections.

```
body
{
    font-family: Arial, Helvetica, sans-serif;
    background-color:#a0a0a0;
    font-size:medium;
}

#content
{
    width: 1080px;
    height: 2000px;
    margin: 0 auto;
    padding: 20px;
    background-color: #FFFFFF;
    color: Black;
}
```

Return to the ***bookCatalog.aspx*** code, go to the ***<head>*** section and add a link to the style sheet.

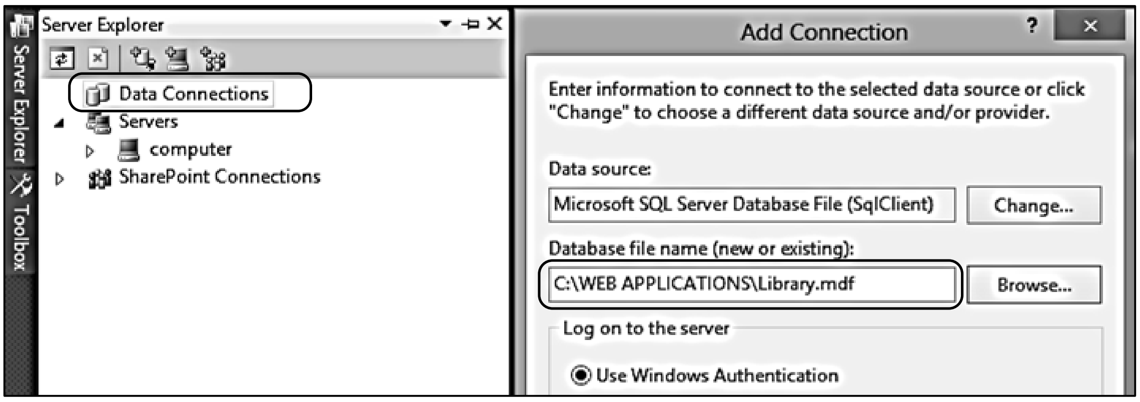
```
<head runat="server">
    <title>Library</title>
    <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
```

Build and run the web page. The heading should be displayed on a white panel.

Book Catalogue

Close the web browser, return to ***Visual Studio*** and stop debugging.

We will now create a database in which ***book*** and ***library user*** records can be stored. Use the main menu ***View*** option to open the ***Server Explorer*** window. Right click on ***Data Connections*** and select ***Add Connection***. Check that the data source is set to ***Microsoft SQL Server Database File***, browse to a suitable folder location for the database, and give the database file name '**Library.mdf**'.



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

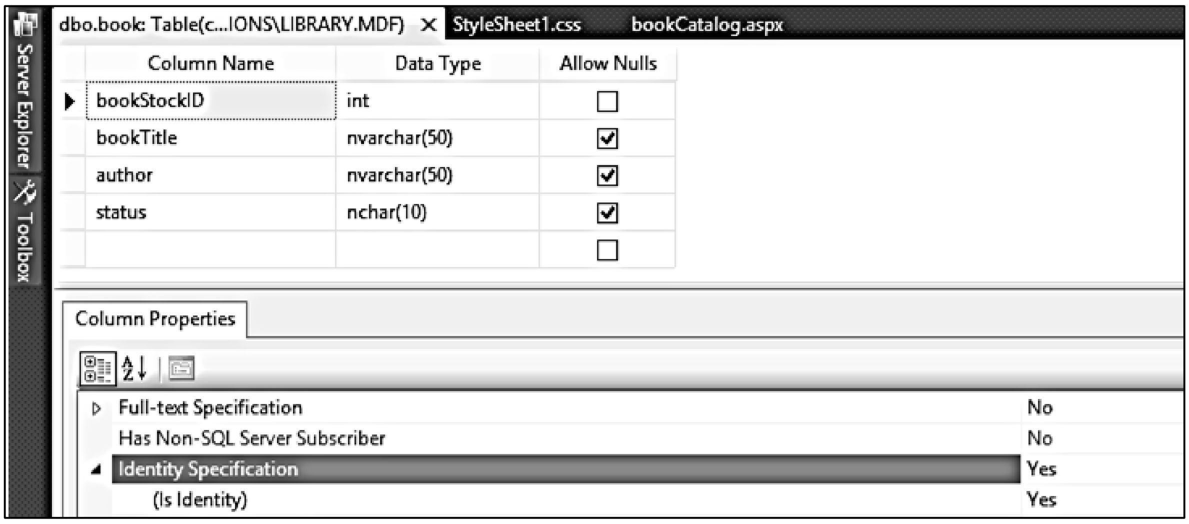
Data source:
Microsoft SQL Server Database File (SqlClient)

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

Log on to the server

● Use Windows Authentication

Go to the ***Server Explorer*** window and click the small arrow to the left of **Library.mdf** to open the database. Click right on the ***Tables*** icon and select ***Add New Table***. Enter fields as shown below. The **bookStockID** field should be set as an auto-number which will be allocated by the computer when records are added. To do this, select the **bookStockID** row, go to the ***properties*** window below and open the ***Identity Specification*** section by clicking the small arrow icon. Set the '***Is Identity***' property to '***Yes***'.

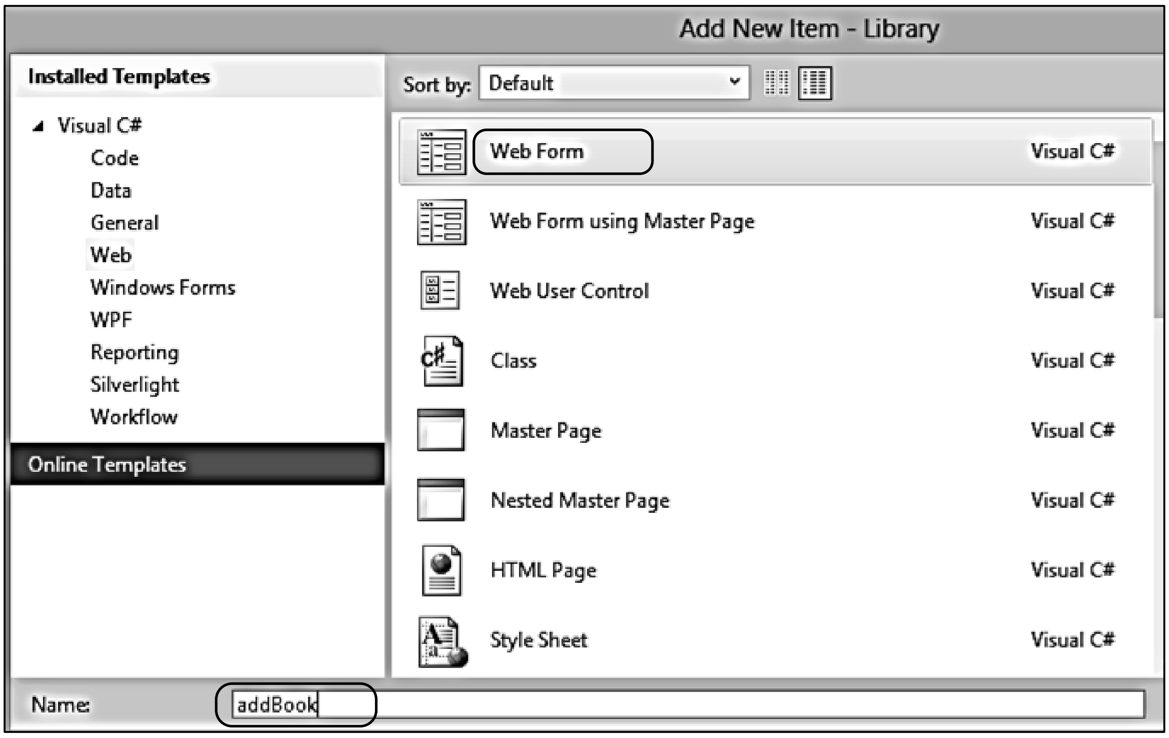


Column Name	Data Type	Allow Nulls
bookStockID	int	☐
bookTitle	nvarchar(50)	✓
author	nvarchar(50)	✓
status	nchar(10)	✓
		☐

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

Close the table by clicking the cross symbol on the tab, and give the name '**book**' to the table.

The next step is to create another web page for the entry of book records. Go to the ***Solution Explorer*** window and right click the ***Library*** program icon. Select ***Add / New Item***, click on ***Web Form*** and give the name '***addBook***'.

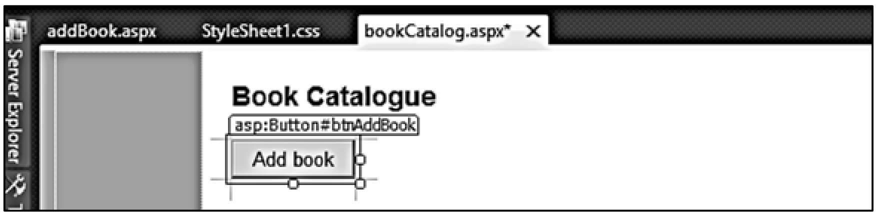


Name: addBook

Before working on the web page to add books to the library, we will make a button link to open this page from the book catalogue. Select the HTML code for ***bookCatalog.aspx*** and add a line of code to create a button.

```
<body>
    <form id="form1" runat="server">
      <div id="content">
         <h3>Book Catalogue</h3>

         <asp:Button ID="btnAddBook" runat="server" Text="Add book" />

      </div>
    </form>
</body>
```

Change to the design view where the button should now appear.



Double click the button to create a C# ***button_click*** method. Add a line of code to open the ***addBook*** page.

```
protected void btnAddBook_Click(object sender, EventArgs e)
{
   Response.Redirect("addBook.aspx");
}
```

Go now to the ***addBook.aspx*** HTML code page. Insert lines of code in the ***<head>*** section to:

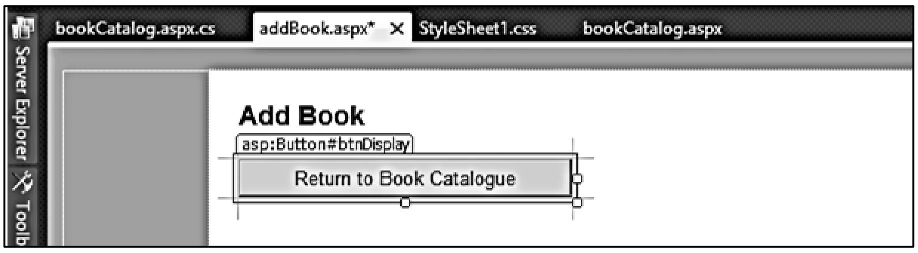
- Link to the stylesheet
- Set the text to '**Library**' on the page tab

and in the ***<body>*** section to:

- Give an ID name to the division
- Set a title 'Add Book' for the page using <h3> heading style
- Insert a button to link back to the book catalogue.

```
<head id="Head1" runat="server">
  <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
  <title>Library</title>
</head>
<body>
   <form id="form1" runat="server">
     <div id="content">
       <h3>Add Book</h3>
       <asp:Button ID="btnDisplay" runat="server" Text="Return to Book Catalogue"/>
     </div>
   </form>
```

Change to the design view and double click the button.

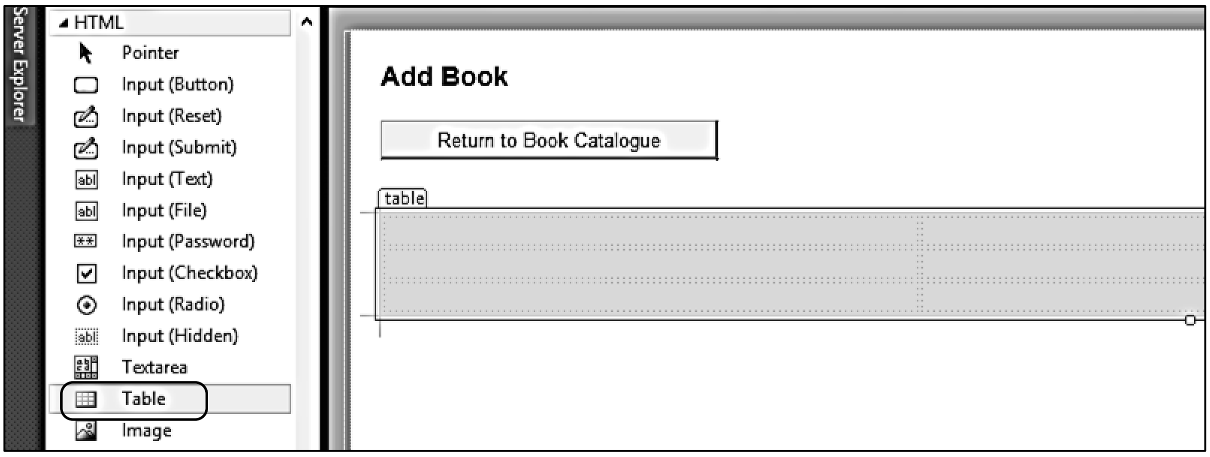


Add a line of C# code to reload the book catalogue page.

```
protected void btnDisplay_Click(object sender, EventArgs e)
{
    Response.Redirect("bookCatalog.aspx");
}
```

Build and run the web site. Check that you can change between the **Book Catalogue** and **Add Book** pages by means of the buttons. Close the web browser, return to ***Visual Studio*** and stop debugging.

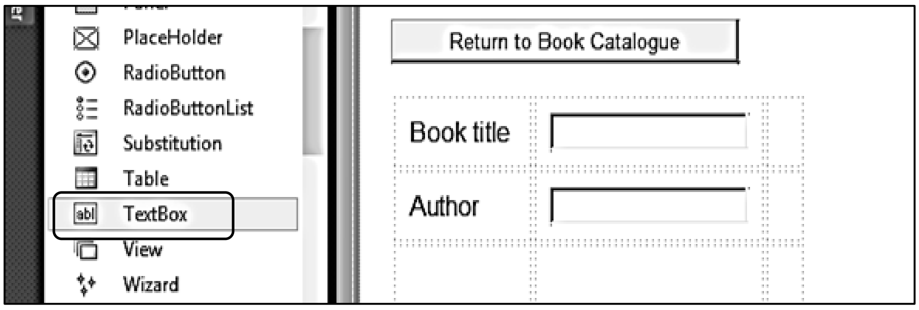
Select the ***addBook.aspx*** page and change to the ***Design*** view. Click to the right of the button and press the ***Enter*** key to move the cursor downwards to a new line. Go to the ***Toolbox*** and scroll down to the ***HTML*** section. Select the ***Table*** component, then drag and drop this onto the form below the button.



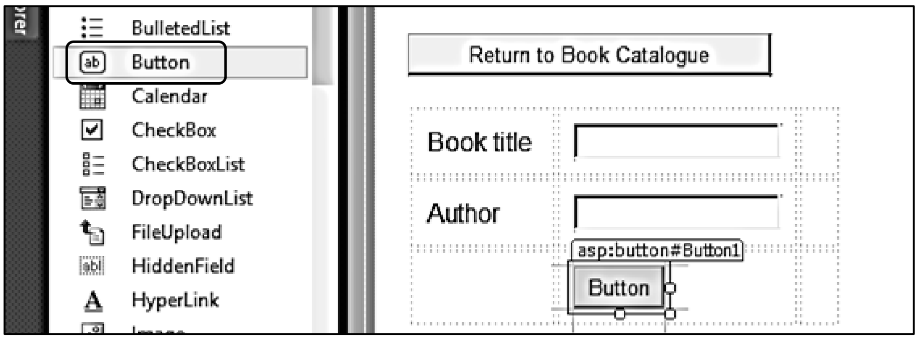
Return to the HTML code by clicking the ***Source*** button below the design window. Check that two ***<br>*** line break tags are present below the button. Delete the 'width: 100%' formatting command from the ***<table>*** tag and replace this with a ***cellpadding = "10"*** command.

```
<asp:Button ID="btnDisplay" runat="server" Text="Return to Book Catalogue"
        onclick="btnDisplay_Click" />
<br />
<br />
<table cellpadding="10">
    <tr>
        <td>
           &nbsp;</td>
```

Return to the ***Design*** view. Type the captions '**Book title**' and '**Author**' into the first cells on the top two rows of the table. Go to the ***Standard*** section of the ***Toolbox*** and drag and drop ***Textboxes*** into cells to the right of the captions.



Add a ***button*** component to the middle column on the bottom row of the table.

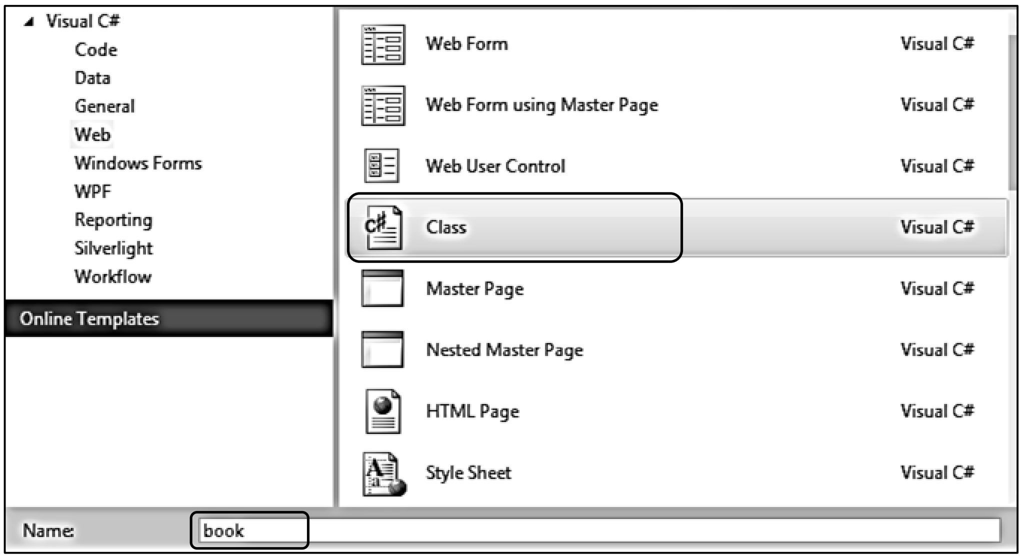


Click the ***Source*** button to return to the HTML code view. We will make a few changes to the ***<table>*** block of code. Set the ***ID***'s and ***widths*** of the text boxes, and set the ***ID*** and ***Text*** value for the button. Note that the <td> sections for the third column on each row of the table are not needed and can be deleted.

```
<table cellpadding="10">
  <tr>
      <td>
          Book title</td>
      <td>
          <asp:TextBox ID="txtTitle" runat="server" Width="300px"></asp:TextBox>
      </td>
  </tr>
  <tr>
      <td>
          Author</td>
      <td>
          <asp:TextBox ID="txtAuthor" runat="server" Width="200px"></asp:TextBox>
      </td>
  </tr>
  <tr>
     <td></td>
     <td>
       <asp:Button ID="btnAdd" runat="server" Text="Add book to Catalogue" />
     </td>
  </tr>
</table>
```

We can now plan how to handle the book data input from the web page. In this and future programs we will use ***classes*** of ***objects*** to store and process data in the RAM memory.

Go to the ***Solution Explorer*** window, right click the ***Library*** program icon, then select ***Add / New Item***. Choose '**Class**', and give the name '**book**'.

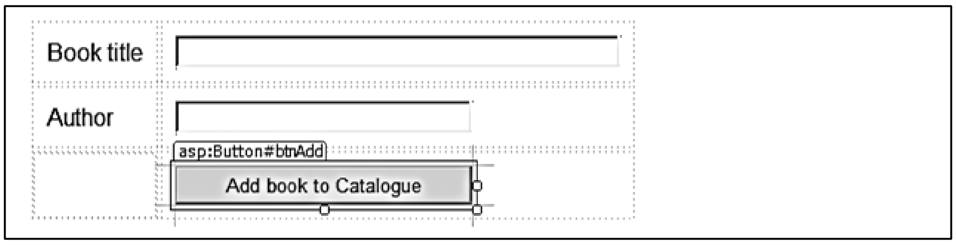


Name: book

The first step in producing an ***object class*** is to create a set of simple methods which will allow data values to be transferred into and out of the objects. These methods are called '***set***' and '***get***' respectively. Add lines to the class file to do this.

```
namespace Library
{
    public class book
    {
        public int stockID { get; set; }
        public string title { get; set; }
        public string author { get; set; }
        public string status { get; set; }
    }
}
```

Return to the ***Design*** view for the **addBook.aspx** page. Double click the 'Add book to Catalogue' button to create a ***button_click*** method.



Add code to the ***button_click*** method to call an **addBook** method which we will create in the ***book*** class file. We will also allow the addBook method to return a message to indicate whether or not the record has been saved successfully. Once the book details have been saved, we will clear the ***txtTitle*** and ***txtAuthor*** text boxes, ready for entry of the next book record.

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    lblMessage.Text = book.addBook(txtTitle.Text, txtAuthor.Text);
    txtTitle.Text = "";
    txtAuthor.Text = "";
}
```

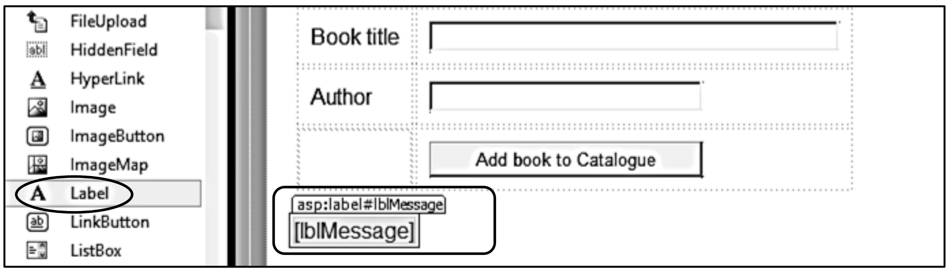
Go now to the ***book.cs*** class file. Begin by adding '**using Data SqlClient**' and '**using Data**' directives at the top of the code page.

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

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

namespace Library
{
    public class book
    {
```

The ***addBook*** method can now be inserted after the block of ***get*** and ***set*** methods. The location of the database also needs to be shown.

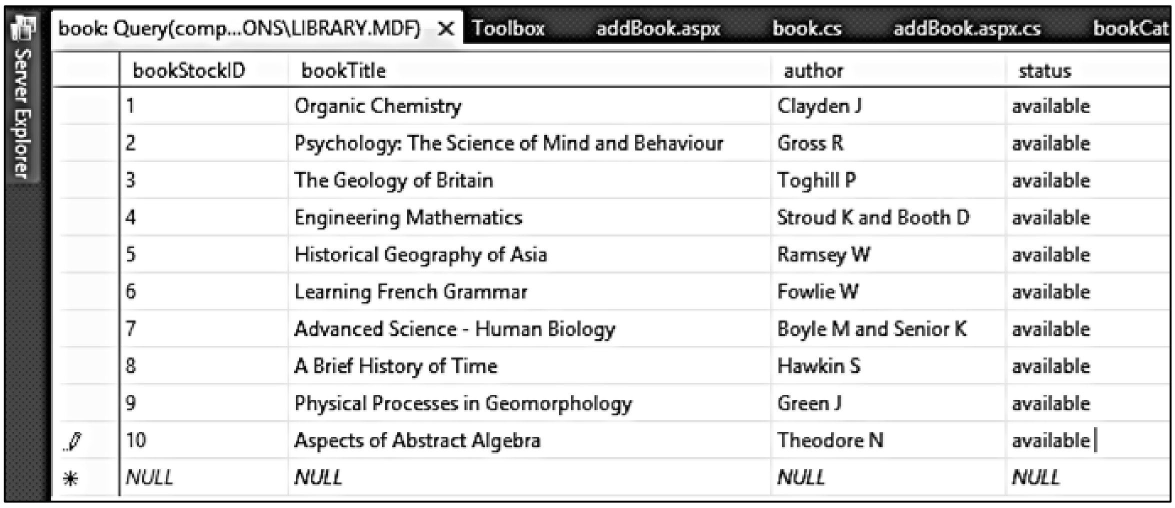
```
public string status { get; set; }

public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

public static string addBook(string title, string author)
{
   string message;
   SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
      AttachDbFilename=" +databaseLocation + "Integrated Security=True;
      Connect Timeout=30; User Instance=True");
   try
   {
       cnTB.Open();
       SqlCommand cmBooks = new SqlCommand();
       cmBooks.Connection = cnTB;
       cmBooks.CommandType = CommandType.Text;
       cmBooks.CommandText = "INSERT INTO book(bookTitle,author,status) VALUES ('"
            + title + "','" + author + "','available')";
       cmBooks.ExecuteNonQuery();
       cnTB.Close();
       message = "Record saved";
   }
   catch
   {
       message = "File error";
   }
   return message;
}
```

Return to the ***Design*** view for the **addBook.aspx** page. Add a label component below the table. Go to the Properties window and set the ***Name*** property to '***lblMessage***'. Clear the ***Text*** property of the label, so it is left blank.



Build and run the web site. Select the ***Add Book*** page and enter a book title and author. Click the '***Add book to Catalogue***' button. If all is well, the message '***Record saved***' should appear.

Continue to add a series of book titles and authors. Close the web browser, return to ***Visual Studio*** and stop debugging. Open the ***Server Explorer*** window and check that the books are listed correctly in the book table of the database. Each book should have been automatically allocated a ***bookStockID***, and its status should be shown as '**available**'.



	bookStockID	bookTitle	author	status
	1	Organic Chemistry	Clayden J	available
	2	Psychology: The Science of Mind and Behaviour	Gross R	available
	3	The Geology of Britain	Toghill P	available
	4	Engineering Mathematics	Stroud K and Booth D	available
	5	Historical Geography of Asia	Ramsey W	available
	6	Learning French Grammar	Fowlie W	available
	7	Advanced Science - Human Biology	Boyle M and Senior K	available
	8	A Brief History of Time	Hawkin S	available
	9	Physical Processes in Geomorphology	Green J	available
	10	Aspects of Abstract Algebra	Theodore N	available
*	NULL	NULL	NULL	NULL

We will now arrange for the list of books to be displayed. Open the ***bookCatalogue.aspx*** page. Add lines of code to insert a label below the button.

```
<body>
    <form id="form1" runat="server">
      <div id="content">
         <h3>Book Catalogue</h3>
         <asp:Button ID="btnAddBook" runat="server" Text="Add book"
              onclick="btnAddBook_Click" />
         <br />
         <br />
         <asp:Label ID="Label1" runat="server"></asp:Label>
      </div>
    </form>
</body>
```

Open the C# code page ***bookCatalogue.aspx.cs.*** (This can be done by right clicking on the HTML page. A pop-up menu will appear. Select the '**View Code**' option.)
Add lines to the ***Page_load*** method which will call a ***loadBooks*** method which we will add to the ***book*** class file. The program will use a property ***bookCount*** to keep a count of the number of book objects created. To make the website operate more efficiently, the **IF** condition ensures that the database is only accessed to load the book data once when the site is first opened.

```
public partial class bookCatalog : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (book.bookCount == 0)
        {
            book.loadBooks();
        }
    }
```

Go now to the book.cs class file. Add the ***bookCount*** property. We will also set up a ***bookObject*** array, ready to link to book objects. Please note, however, that no memory space will actually be allocated to these objects until they are created by loading records from the database while the program is running.

```
public class book
{
    public static int bookCount = 0;
    public static book[] bookObject = new book[100];

    public int stockID { get; set; }
    public string title { get; set; }
```

Create a ***loadBooks*** method below the block of ***get*** and ***set*** methods.

```
    public string author { get; set; }
    public string status { get; set; }

    public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

    public static void loadBooks()
    {

    }

    public static string addBook(string title, string author)
    {
        string message;
```

Code can now be added to the loadBooks method. This carries out a series of tasks:

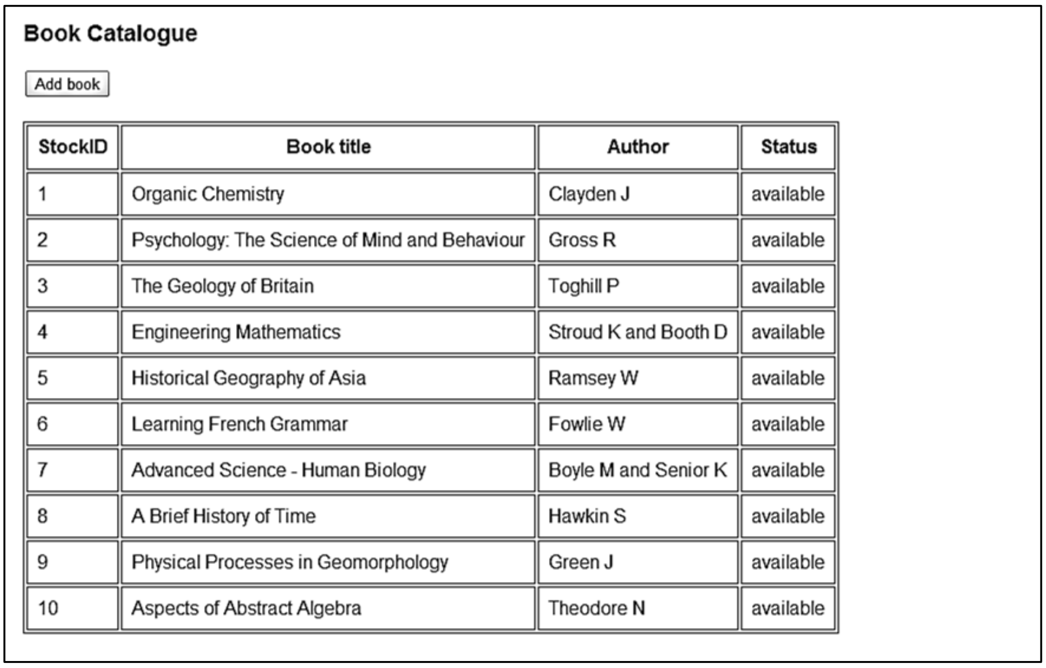
- All the book records are loaded from the database using the SQL command '***SELECT * FROM book***'
- We find the number of book records loaded, and use this value to set the ***countRecords*** variable.
- A loop then collects the **title** and **author** data for each book and uses this to create a ***book*** object.

```
public static void loadBooks()
{
    DataSet dsBooks = new DataSet();

    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
         AttachDbFilename=" + databaseLocation + "Integrated Security=True;
         Connect Timeout=30; User Instance=True");
    try
     {
         cnTB.Open();
         SqlCommand cmBooks = new SqlCommand();
         cmBooks.Connection = cnTB;
         cmBooks.CommandType = CommandType.Text;
         cmBooks.CommandText = "SELECT * FROM book";
         SqlDataAdapter daBooks = new SqlDataAdapter(cmBooks);
         daBooks.Fill(dsBooks);
         cnTB.Close();

         int countRecords = dsBooks.Tables[0].Rows.Count;
         book.bookCount = 0;

         for (int i = 0; i < countRecords; i++)
         {
             DataRow drBooks = dsBooks.Tables[0].Rows[i];
             int bookStockID = (int)drBooks[0];
             string bookTitle = Convert.ToString(drBooks[1]);
             string author = Convert.ToString(drBooks[2]);
             string status = Convert.ToString(drBooks[3]);

             book.bookObject[book.bookCount] = new book();
             book.bookObject[book.bookCount].stockID = bookStockID;
             book.bookObject[book.bookCount].title = bookTitle;
             book.bookObject[book.bookCount].author = author;
             book.bookObject[book.bookCount].status = status;

             book.bookCount++;
         }
     }
     catch
     {

     }
}
```

Return to the C# page ***bookCatalog.aspx.cs*** and add HTML code to build a table to display the list of books. The code will be inserted into the page when the website runs by using the label component. We first create a set of column headings for the table, then use a loop to add details of each book on a separate row.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (book.bookCount == 0)
    {
        book.loadBooks();
    }

    Label1.Text = "";
    string s = "";
    string status;

    s += "<table cellpadding=8 border=1>";
    s += "<tr>";
    s += "<th>";
    s += "StockID";
    s += "</th>";
    s += "<th>";
    s += "Book title";
    s += "</th>";
    s += "<th>";
    s += "Author";
    s += "</th>";
    s += "<th>";
    s += "Status";
    s += "</th>";
    s += "</tr>";
    for (int i = 0; i < book.bookCount; i++)
    {
        s += "<tr>";
        s += "<td>";
        s += book.bookObject[i].stockID;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].title;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].author;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].status;
        s += "</td>";
        s += "</tr>";
    }
    s += "</table>";
    Label1.Text = s;

}
```

Build and run the web page. The list of books should now be displayed.

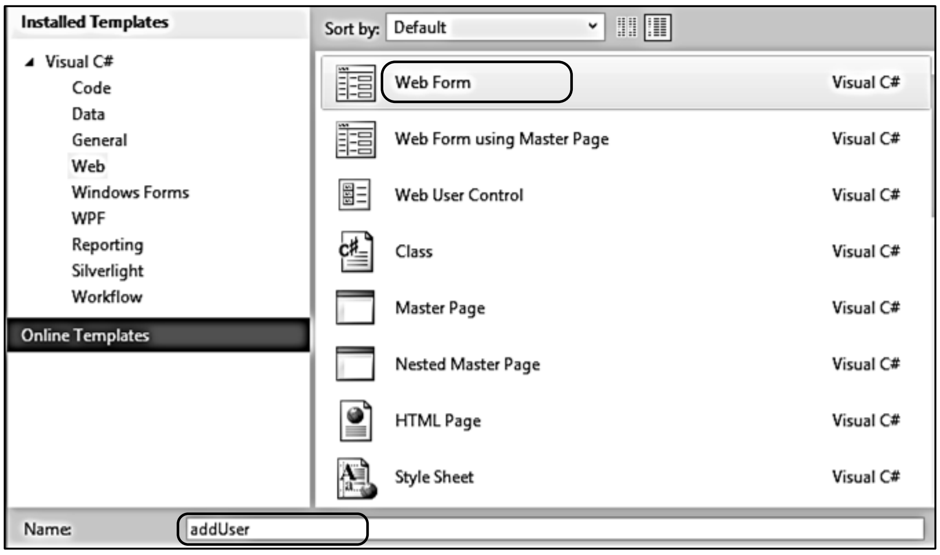


Book Catalogue

Add book

StockID	Book title	Author	Status
1	Organic Chemistry	Clayden J	available
2	Psychology: The Science of Mind and Behaviour	Gross R	available
3	The Geology of Britain	Toghill P	available
4	Engineering Mathematics	Stroud K and Booth D	available
5	Historical Geography of Asia	Ramsey W	available
6	Learning French Grammar	Fowlie W	available
7	Advanced Science - Human Biology	Boyle M and Senior K	available
8	A Brief History of Time	Hawkin S	available
9	Physical Processes in Geomorphology	Green J	available
10	Aspects of Abstract Algebra	Theodore N	available

Close the browser and stop debugging. The next stage is to create a web page where library users can be registered. Go to the ***Solution Explorer*** and right click on the ***Library*** program icon. Select ***Add / New Item***, then choose '***Web Form***'. Give the name '**addUser**'.



Name: addUser

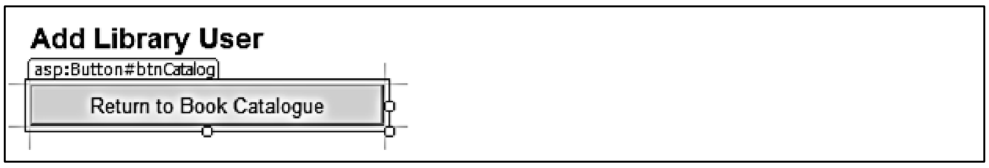
Begin by adding lines to the ***<head>*** section to link to the style sheet, and to display a title '**Library**' on the page tab.

```
<head id="Head1" runat="server">
    <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
    <title>Library</title>
</head>
```

Move now to the ***<body>*** section. Add code to display a page heading '**Add Library User'**, and a button to return to the book catalogue page. We will also give an ID '**content**' to the division.

```
<body>
   <form id="form1" runat="server">
   <div id="content">
     <h3>Add Library User</h3>
     <asp:Button ID="btnCatalog" runat="server" Text="Return to Book Catalogue" />
   </div>
   </form>
</body>
```

Change to the Design view and check that the button has been created correctly.



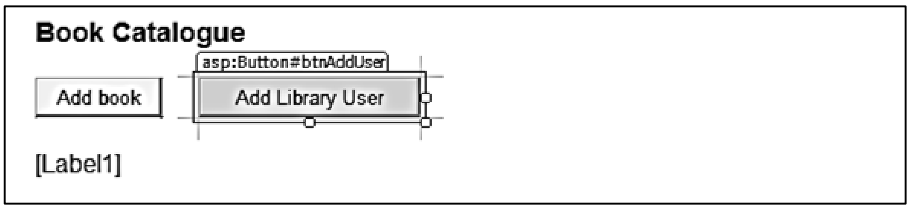
Double click to create a ***button_click*** method. Add code to link to the ***bookCatalog.aspx*** page.

```
protected void btnCatalog_Click(object sender, EventArgs e)
{
    Response.Redirect("bookCatalog.aspx");
}
```

Go now to the ***bookCatalog.aspx*** HTML code page. Add a button which will link back to the **Add Library User** page.

```
<div id="content">
   <h3>Book Catalogue</h3>
   <asp:Button ID="btnAddBook" runat="server" Text="Add book"
        onclick="btnAddBook_Click" />
   &nbsp; &nbsp; &nbsp;
   <asp:Button ID="btnAddUser" runat="server" Text="Add Library User" />
   <br />
   <br />
   <asp:Label ID="Label1" runat="server"></asp:Label>
</div>
```

Change to the Design view and check that the button has been created.



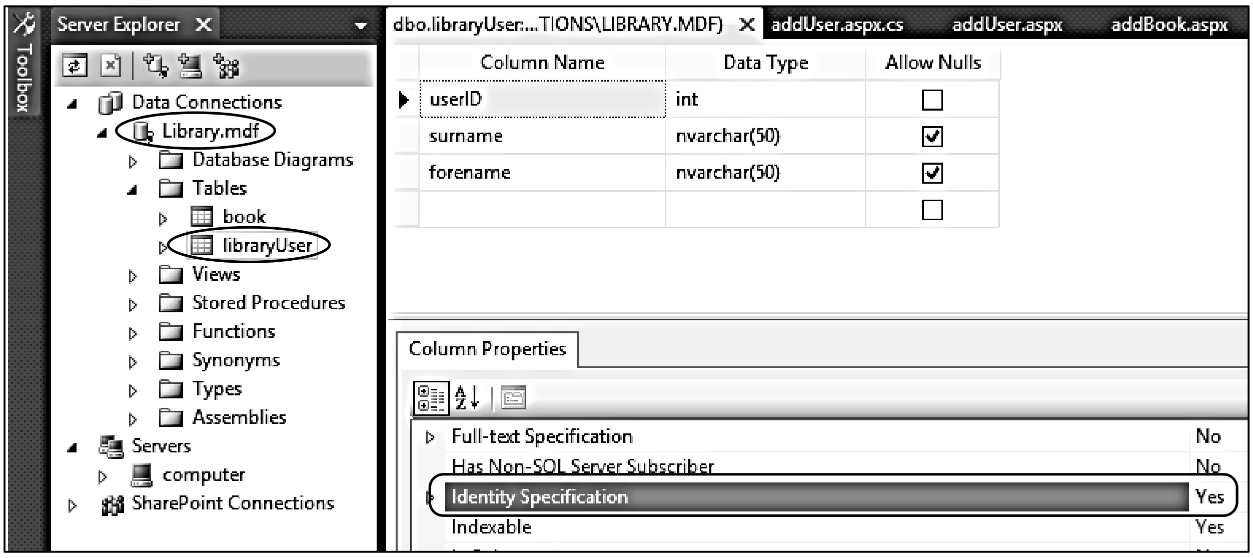
Double click the button to create a method, then add code to link to the addUser.aspx page.

```
protected void btnAddUser_Click(object sender, EventArgs e)
{
    Response.Redirect("addUser.aspx");
}
```

Build and run the website. Check that it is possible to navigate backwards and forwards between the **Book Catalogue** and **Add Library User** pages by means of the buttons. Close the web browser, return to ***Visual Studio*** and stop debugging.

A table needs to be added to the database to record library users. Go to the ***Server Explorer*** window, open the database by clicking the small arrow the left of ***Library.mdf*** icon. Right click ***Tables*** and select ***Add New Table***.

Add fields to the table. We will set the ***userID*** to be an autonumber. To do this, scroll down to the ***Identity Specification*** property, open this option by clicking the small arrow, then set '***IsIdentity***' to '***Yes***'. Close the table with the small cross on the tab, and give the name '***libraryUser***'.

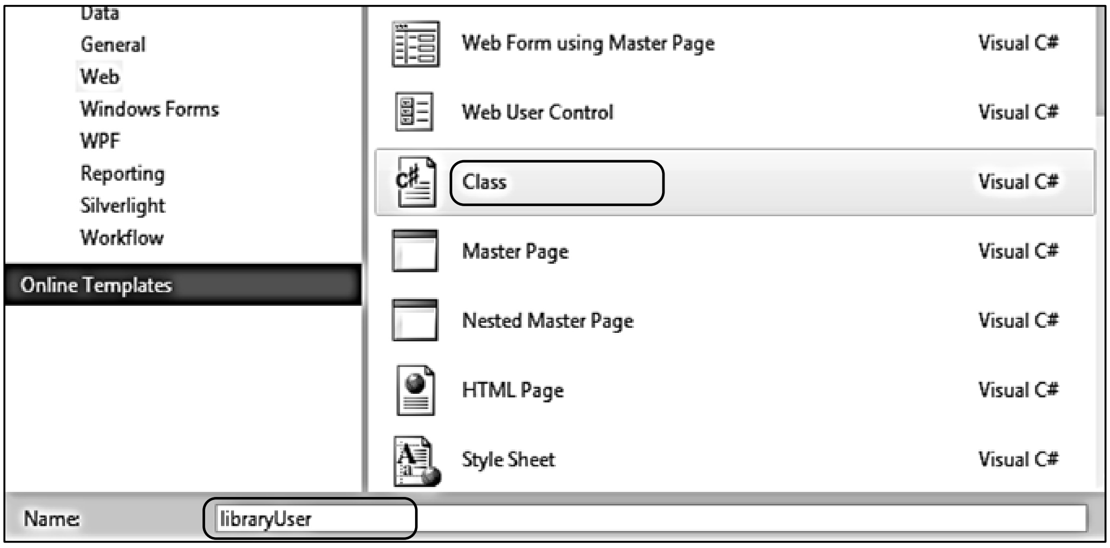


Column Name	Data Type	Allow Nulls
userID	int	☐
surname	nvarchar(50)	☑
forename	nvarchar(50)	☑
		☐

Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
Indexable	Yes

As for books, we will create a class of objects to hold details of each library user.

Go to the ***Solution Explorer*** window. Right click the **Library** program icon and select ***Add / New Item***. Choose '***Class***' and give the name '**libraryUser**'.



As with the book class, we will begin by adding some code to the empty ***libraryUser*** class file which will be needed by the program:

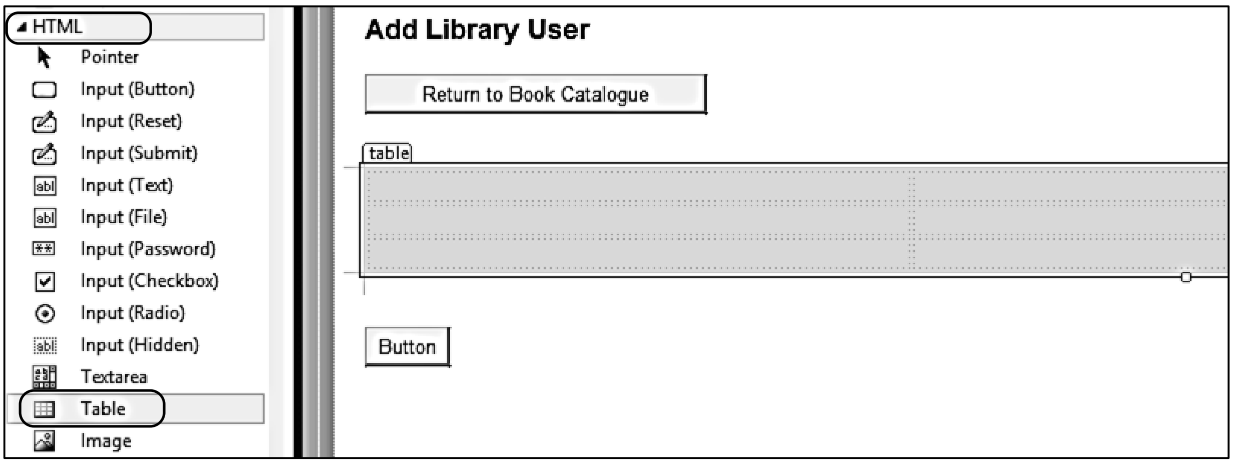
- Insert '**using Data SqlClient**' and '**using Data**' directives.
- Set up a ***userCount*** property to keep a count of the number of library users, and set up a ***userObject*** array to link to the library user objects when they are created.
- Produce ***set*** and ***get*** methods to allow data to be transferred into and out of the library user objects.
- Give the location of the database.

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

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

namespace Library
{

    public class libraryUser
    {
        public static int userCount = 0;
        public static libraryUser[] userObject = new libraryUser[100];

        public int userID { get; set; }
        public string surname { get; set; }
        public string forename { get; set; }

        public static string databaseLocation="C:\\WEB APPLICATIONS\\Library.mdf;";
    }
}
```

Return to the ***addUser.aspx*** page and select the ***Design*** view.

Click to the right of the '**Return to Book Catalogue**' button and press the ***Enter*** key twice to move the cursor downwards. Open the ***Toolbox*** and scroll down to the ***HTML*** section. Select the ***Table*** component, drag and drop onto the form.

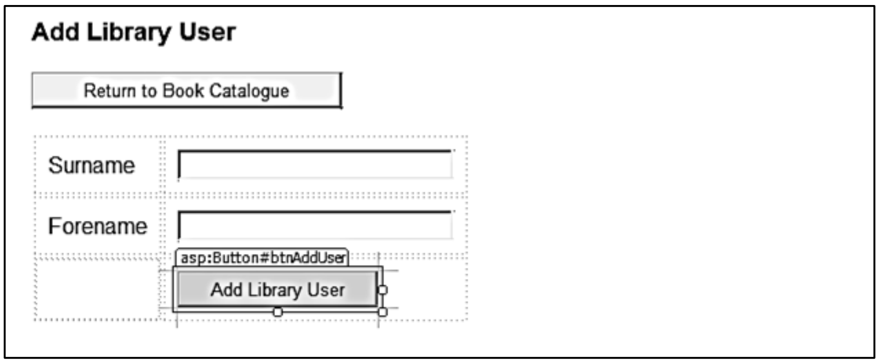
Click to the right of the table and press the ***Enter*** key twice more to move the cursor downwards. Scroll the ***Toolbox*** up to the ***Standard*** section. Select the ***Button*** component, drag and drop onto the form below the table.



Click the ***Source*** button to change to the HTML page. Edit the code which has just been inserted by the ***Design*** view. Drag the line of ***Button*** code into the table, as shown below. Some ***<td></td>*** table data tags and ***<br />*** line break tags are no longer required and can be deleted.

```
    <div id="content">
       <h3>Add Library User</h3>
       <asp:Button ID="btnCatalog" runat="server" Text="Return to Book Catalogue"
           onclick="btnCatalog_Click" />
       <br />
       <br />
       <table cellpadding="10">
          <tr>
             <td>Surname</td>
             <td>
                <asp:TextBox ID="txtSurname" runat="server"
                        Width="200px"></asp:TextBox>
             </td>
         </tr>
         <tr>
             <td>Forename</td>
             <td>
                <asp:TextBox ID="txtForename" runat="server"
                        Width="200px"></asp:TextBox>
             </td>
         </tr>
         <tr>
             <td></td>
             <td>
                <asp:Button ID="btnAddUser" runat="server" Text="Add Library User"/>
             </td>
         </tr>
      </table>
   </div>
```

Return to the ***Design*** view to check that the page now contains captions, text boxes and the button in the correct cells of the table.



Also build and run the web site to check that the ***Add Library User*** page appears correctly. Close the web browser, return to ***Visual Studio*** and stop debugging.

In the Design screen for ***addUser.aspx***, double click the button to create a C# method. Add code which will call a method '***addUser***' which we will add to the ***libraryUser*** class to save the record into the database table. Once the record has been saved, the text boxes can be cleared, ready for the next data entry.

```
protected void btnAddUser_Click(object sender, EventArgs e)
{
    libraryUser.addUser(txtSurname.Text, txtForename.Text);
    txtSurname.Text = "";
    txtForename.Text = "";
}
```

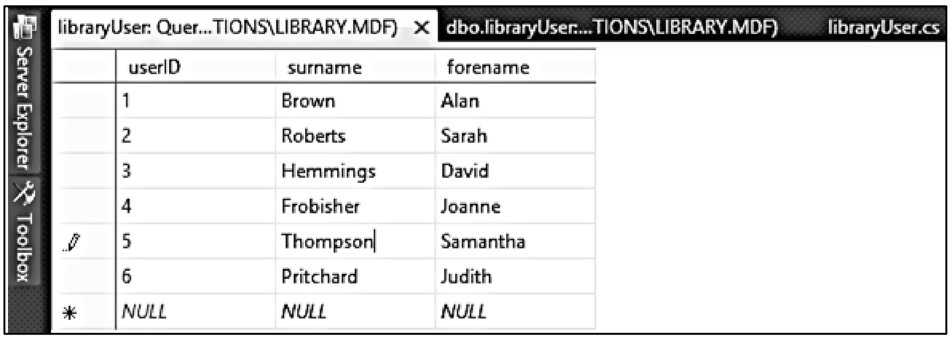
Go now to the ***libraryUser*** class file. Insert the ***addUser( )*** method below the database location line.

```
public int userID { get; set; }
public string surname { get; set; }
public string forename { get; set; }

public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

public static void addUser(string surname, string forename)
{
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmUsers = new SqlCommand();
        cmUsers.Connection = cnTB;
        cmUsers.CommandType = CommandType.Text;
        cmUsers.CommandText = "INSERT INTO libraryUser(surname,forename)
             VALUES ('" + surname + "','" + forename + "')";
        cmUsers.ExecuteNonQuery();
        cnTB.Close();
    }
    catch
    {

    }
 }
```

Build and run the web site. Go to the **Add Library User** page, then enter a series of surnames and forenames, clicking the 'Add Library User' button to save each record into the database.

Close the web browser, return to ***Visual Studio*** and stop debugging.

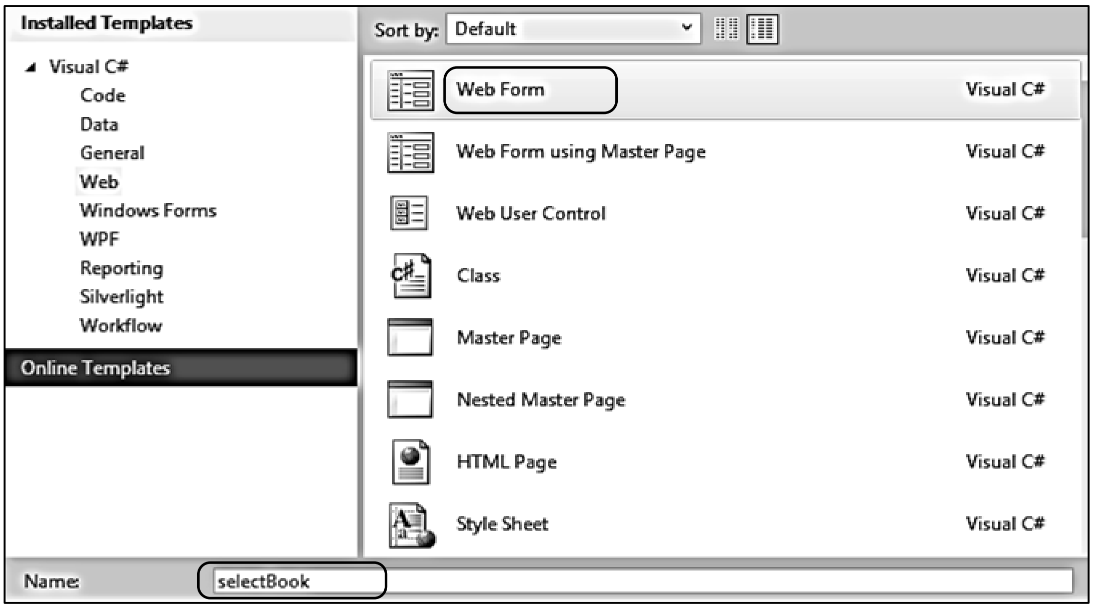
Go to the ***Database Explorer*** window. Right click the ***libraryUser*** table icon and select '**Show Table Data**'. Check that the names you entered have been inserted into the table correctly.


Server Explorer
Toolbox
libraryUser: Quer...TIONS\LIBRARY.MDF)
dbo.libraryUser:...TIONS\LIBRARY.MDF)
libraryUser.cs

userID	surname	forename
1	Brown	Alan
2	Roberts	Sarah
3	Hemmings	David
4	Frobisher	Joanne
5	Thompson	Samantha
6	Pritchard	Judith
NULL	NULL	NULL

The final stage of the project is to arrange for library users to borrow and then return books. To do this we will create another web page.

Go to the ***Solution Explorer*** window and right click the ***Library*** project icon. Select ***Add / New Item***. Choose ***Web Form*** and give the name '**selectBook**'.

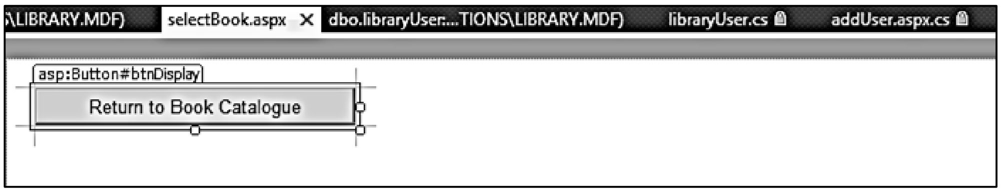


Name: selectBook

Open the ***selectBook.aspx*** code page. In the ***<head>*** section, add a link to the style sheet and a title for the page tab. In the ***<body>*** section give an **ID** for the ***division***, and insert a button showing the text '**Return to Book Catalogue**'.

```
<head id="Head1" runat="server">
   <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
   <title>Library</title>
</head>
<body>
   <form id="form1" runat="server">
   <div id="content">
      <asp:Button ID="btnDisplay" runat="server" Text="Return to Book Catalogue"/>
   </div>
   </form>
</body>
```

Change to the ***Design*** screen and check that the button is displayed.



Double click the button to create a ***button_click*** method, then add code to link to the **Book Catalogue** page.

```
protected void btnDisplay_Click(object sender, EventArgs e)
{
    Response.Redirect("bookCatalog.aspx");
}
```

Build and run the web page. Click the '***Return to Book Catalogue***' button and check that the book list page is then displayed. Close the web browser to return to ***Visual Studio***, then stop debugging.

We will now add buttons alongside each book record to allow a loan to be entered. Before doing this, open the ***StyleSheet*** file and add formatting which will be needed for the ***<table>*** and ***<a href..>*** tags.

```
#content
{
    width: 1080px;
    height: 2000px;
    margin: 0 auto;
    padding: 20px;
    background-color: #FFFFFF;
    color: Black;
}

table
{
    border:1px;
}

a
{
    color: White;
    text-decoration:none;
}
```

Go to the Solution Explorer window and select the C# code for the Book Catalogue page, ***bookCatalog.aspx.cs***. Add the string variable '**userIDwanted**' at the start of the ***bookCatalog*** class.

```
public partial class bookCatalog : System.Web.UI.Page
{
    string userIDwanted;

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

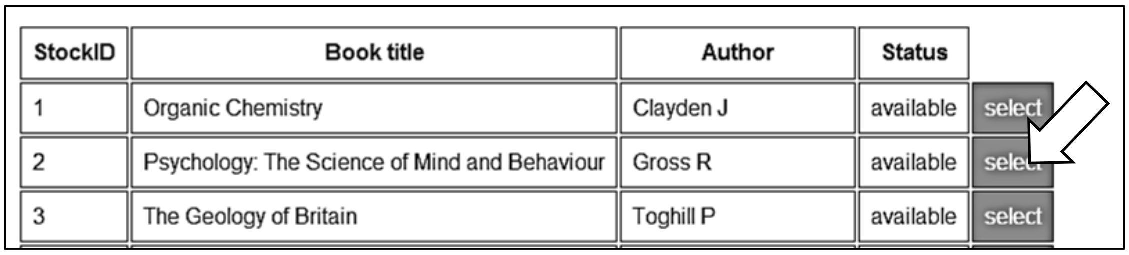
Code can now be added to the loop in the ***PageLoad( )*** method. This will carry out several tasks:

- A rectangle is created alongside the book record by setting the background colour of a cell to pale blue using the colour code ***#00AAEE***. This will act as a button showing the text '**select**'.

- If the rectangle is clicked, the ***Select Book*** page will be loaded. The ID numer of the selected book is transferred to the ***Select Book*** page by adding this as a ***parameter*** to the web page URL.
  For example, if the stockID is 3, the program will build up the command string:
  < a href = 'selectBook.aspx ?stockID = 3'>

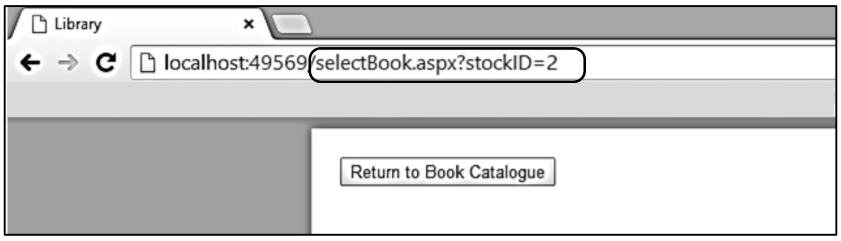
```
    for (int i = 0; i < book.bookCount; i++)
    {
        s += "<tr>";
        s += "<td>";
        s += book.bookObject[i].stockID;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].title;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].author;
        s += "</td>";
        s += "<td>";
        s += book.bookObject[i].status;

        status=book.bookObject[i].status;
        status = status.Trim();
        s += "</td>";
        s += "<td bgcolor=#00AAEE>";
        s += "<a href='selectBook.aspx?stockID=";
        s += book.bookObject[i].stockID;
        s += "'>select</a>";

        s += "</td>";
        s += "</tr>";
    }
    s += "</table>";

    Label1.Text = s;
}
```

Build and run the web page. '**Select**' buttons should appear alongside each book record. Click one of the buttons:



StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	available	select
2	Psychology: The Science of Mind and Behaviour	Gross R	available	select
3	The Geology of Britain	Toghill P	available	select

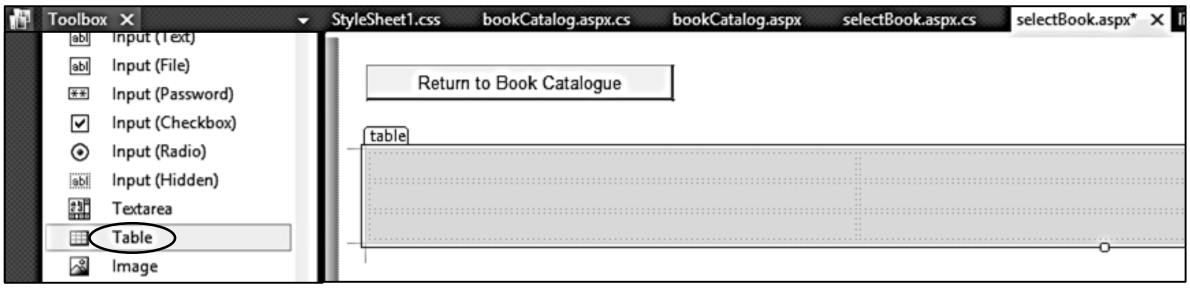
The ***selectBook*** page should open. Examine the address window at the top of the screen. In addition to the page URL ***selectBook.aspx***, the ***stockID*** of the selected book should be shown.



Close the web browser, return to ***Visual Studio*** and stop debugging.

Go to the ***Design*** view of the ***selectBook.aspx*** page. Click to the right of the '**Return to Book Catalogue**' button and press enter twice to move the cursor downwards to miss an empty line.

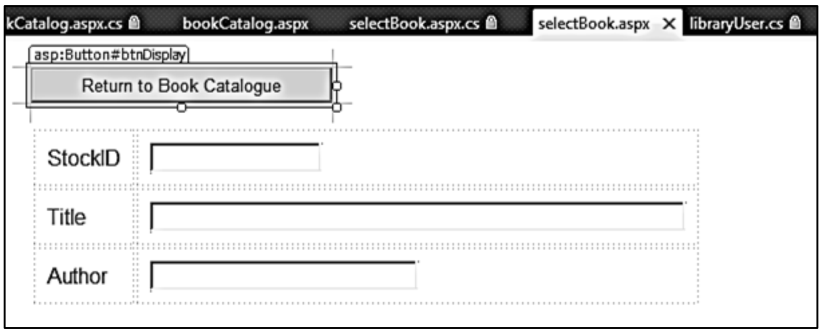
Open the ***Toolbox*** and scroll down to the ***HTML*** section. Select the ***Table*** component and drag and drop onto the web page below the button.



Click the ***Source*** button to change to the HTML code. Edit the ***<table>*** section by adding captions and text boxes for display of the book details. Remove any ***<td></td>*** tags no longer required.

```
<table cellpadding="10">
   <tr>
      <td>StockID</td>
      <td>
          <asp:TextBox ID="txtStockID" runat="server"></asp:TextBox>
      </td>
   </tr>
   <tr>
      <td>Title</td>
      <td>
          <asp:TextBox ID="txtTitle" runat="server" Width="400px">
          </asp:TextBox>
      </td>
   </tr>
   <tr>
      <td>Author</td>
      <td>
         <asp:TextBox ID="txtAuthor" runat="server" Width="200px">
         </asp:TextBox>
      </td>
   </tr>
</table>
```

Return to the ***Design*** view and check that the table has been set up correctly.



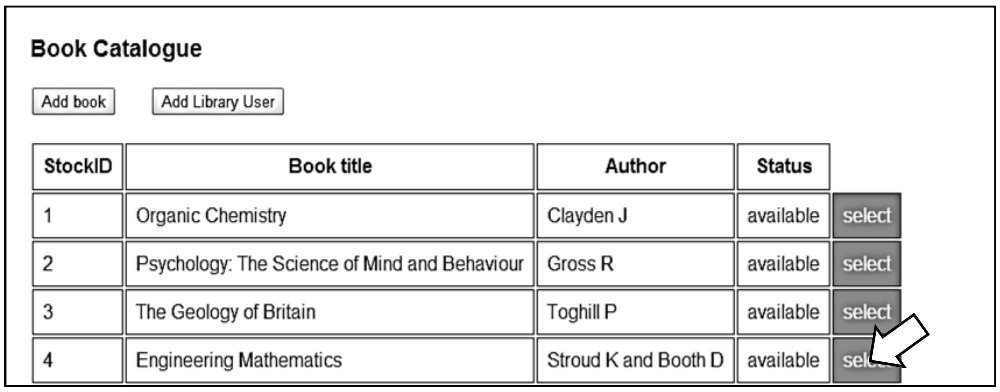
Right click on the design page to open a pop-up menu. Select the '**View Code**' option.

Create a **stockID** string variable in the ***selectBook*** class near the top of the C#page.
Add lines of code to the ***Page_Load( )*** method. This code carries out several tasks:

- The **stockID** value is extracted from the URL page address. This is stored as an integer variable '***n***'.
- Book records are loaded by calling the ***loadBooks( )*** method in the ***book*** class file.
- A loop is used to find and display the required book details in the text boxes.

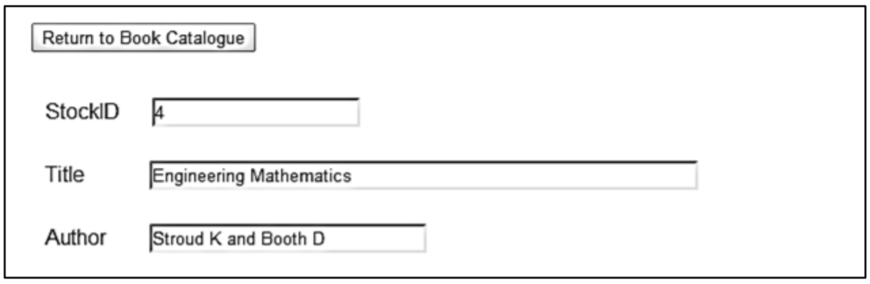
```
public partial class selectBook : System.Web.UI.Page
{
    String stockID;

    protected void Page_Load(object sender, EventArgs e)
    {

        stockID = Request.QueryString["stockID"];
        book.loadBooks();
        txtStockID.Text = stockID;
        int n = Convert.ToInt16(stockID);
        String status = "";
        for (int i = 0; i < book.bookCount; i++)
        {
            if (book.bookObject[i].stockID == n)
            {
                txtTitle.Text = book.bookObject[i].title;
                txtAuthor.Text = book.bookObject[i].author;
                status = book.bookObject[i].status;
            }
        }

    }
```

Build and run the web site. Go to the **Book Catalogue** page and click to select a book.


Book Catalogue

StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	available	select
2	Psychology: The Science of Mind and Behaviour	Gross R	available	select
3	The Geology of Britain	Toghill P	available	select
4	Engineering Mathematics	Stroud K and Booth D	available	select

Check that the ***selectBook*** page opens and the correct book details are displayed.



StockID	4
Title	Engineering Mathematics
Author	Stroud K and Booth D

Close the web browser, return to Visual Studio and stop debugging.

Go to the HTML code page for ***selectBook.aspx***. After the table section, add a Panel component with captions, a drop down list and button. These will allow the borrower to be selected from the list of names of library users.

```
            <td>
               <asp:TextBox ID="txtAuthor" runat="server" Width="200px">
               </asp:TextBox>
            </td>
         </tr>
      </table>

      <br />
      <br />
      <asp:Panel ID="Panel1" runat="server">
           Available for loan.
           <br />
           <br />
           Select borrower:
           <asp:DropDownList ID="DropDownList1" runat="server" Visible="True">
           </asp:DropDownList>
           <br />
           <br />
           <asp:Button ID="btnLoan" runat="server" Text="Record book loan"/>
           <br />
      </asp:Panel>

   </div>
```

Before running the web page, the names of library users must be loaded into the drop down list. To do this, return to the C# page ***selectBook.aspx.cs***. Insert code into the ***Page_Load( )*** method. This will:

- Load the library user records by means of a method ***loadUsers*** which we will add to the ***libraryUser*** class file.
- Clear any previous entries from the drop down list.
- Use a loop to build up a string consisting of the **userID**, **surname** and **forename** of each library user, then add this to the drop down list.

```
protected void Page_Load(object sender, EventArgs e)
{
   stockID = Request.QueryString["stockID"];

   if (DropDownList1.Items.Count < 1)
   {
      libraryUser.loadUsers();
      string s;
      DropDownList1.Items.Clear();
      for (int i = 0; i < libraryUser.userCount; i++)
      {
          s = "";
          s += libraryUser.userObject[i].userID + ": ";
          s += libraryUser.userObject[i].surname + ", ";
          s += libraryUser.userObject[i].forename;
          DropDownList1.Items.Add(s);
      }
    }

    book.loadBooks();
    txtStockID.Text = stockID;
    int n = Convert.ToInt16(stockID);
```

Move now to the ***libraryUser.cs*** class file. Add the ***loadUsers( )*** method after the database location line.

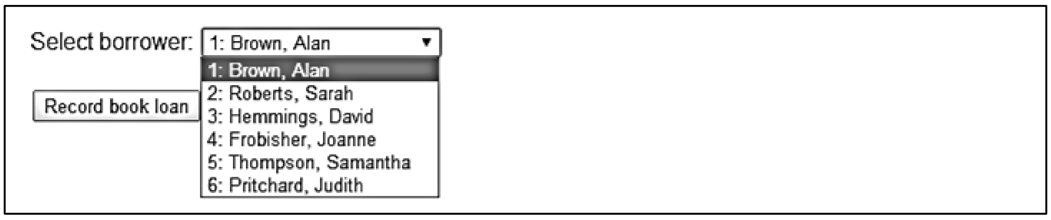
```
public class libraryUser
{
    public static int userCount = 0;
    public static libraryUser[] userObject = new libraryUser[100];

    public int userID { get; set; }
    public string surname { get; set; }
    public string forename { get; set; }

    public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

    public static void loadUsers()
    {

    }
```

Add code to the ***loadUsers( )*** method. This will carry out several tasks:

- The database is opened and all ***libraryUser*** records are loaded into a data set.
- A loop is used to access data from each libraryUser record and create a library user ***object***. The number of ***userObjects*** created is stored as the variable ***userCount***.

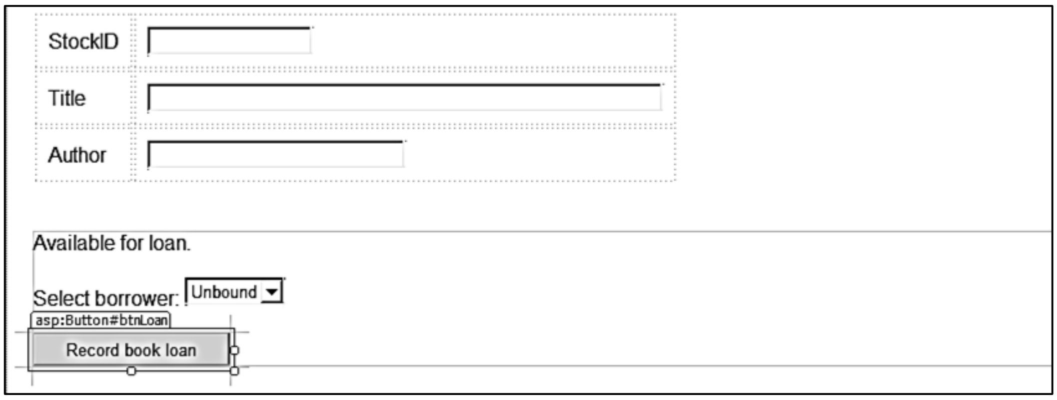
```
public static void loadUsers()
{
    DataSet dsUsers = new DataSet();

    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
       AttachDbFilename=" + databaseLocation + "Integrated Security=True;
       Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmUsers = new SqlCommand();
        cmUsers.Connection = cnTB;
        cmUsers.CommandType = CommandType.Text;
        cmUsers.CommandText = "SELECT * FROM libraryUser";
        SqlDataAdapter daUsers = new SqlDataAdapter(cmUsers);
        daUsers.Fill(dsUsers);
        cnTB.Close();

        int countRecords = dsUsers.Tables[0].Rows.Count;
        libraryUser.userCount = 0;
        for (int i = 0; i < countRecords; i++)
        {
            DataRow drUsers = dsUsers.Tables[0].Rows[i];
            int userID = (int)drUsers[0];
            string surname = Convert.ToString(drUsers[1]);
            string forename = Convert.ToString(drUsers[2]);

            libraryUser.userObject[libraryUser.userCount] = new libraryUser();
            libraryUser.userObject[libraryUser.userCount].userID = userID;
            libraryUser.userObject[libraryUser.userCount].surname = surname;
            libraryUser.userObject[libraryUser.userCount].forename = forename;
            libraryUser.userCount++;
        }
    }
    catch
    {

    }
}
```

Build and run the ***Book Catalogue*** page, then select a book. When the ***selectBook*** page opens, check that the library users are displayed correctly in the drop down list.



Close the web browser, return to ***Visual Studio*** and stop debugging.

We will now work on the program code needed to record book loans. Go to the ***Design*** view of the ***selectBook.aspx*** page. Double click the '**Record book loan**' button to create a ***button_click*** method.



Add code to the method to carry out a series of tasks:

- A string variable '**s**' is used to store details of the borrower selected from the drop down list.
- The variable '**s**' is split into three parts: the **userID**, **surname** and **forename**.
- The **stockID** of the book borrowed is obtained from the ***txtStockID*** text box.
- Details of the loan will be saved using a ***recordLoan( )*** method which we will add to the ***book*** class file.
- Book records are reloaded, so that the new loan is included, then the program returns to the ***Book Catalogue*** page.

```
protected void btnLoan_Click(object sender, EventArgs e)
{
    string s = DropDownList1.SelectedItem.Text;
    string[] words = s.Split(':');
    int userID = Convert.ToInt16(words[0]);
    int stockID = Convert.ToInt16(txtStockID.Text);
    book.recordLoan(stockID, userID);
    book.loadBooks();
    Response.Redirect("bookCatalog.aspx");
}
```

Open the ***book.cs*** class file and add the ***recordLoan( )*** method after the database location line. Notice the parameters which pass the **stockID** and **userID** values to the method.

```
public string author { get; set; }
public string status { get; set; }
public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

public static void recordLoan(int stockID, int userID)
{

}
```

Code can now be added to the ***recordLoan( )*** method. This opens the database in the normal way, then uses an SQL command to change the status of the required book from '**available**' to the **userID** number of the borrower.

```
public static void recordLoan(int stockID, int userID)
{
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename=" + databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmBooks = new SqlCommand();
        cmBooks.Connection = cnTB;
        cmBooks.CommandType = CommandType.Text;
        cmBooks.CommandText = "UPDATE book SET  status ='" + userID +
            "' WHERE bookStockID='" + stockID + "'";
        cmBooks.ExecuteNonQuery();
        cnTB.Close();
    }
    catch
    {

    }
}
```

Go next to the Book Catalogue C# code page, ***bookCatalog.aspx.cs***. Locate the loop within the ***Page_Load( )*** method and add lines to the code as shown below. These carry out several tasks:

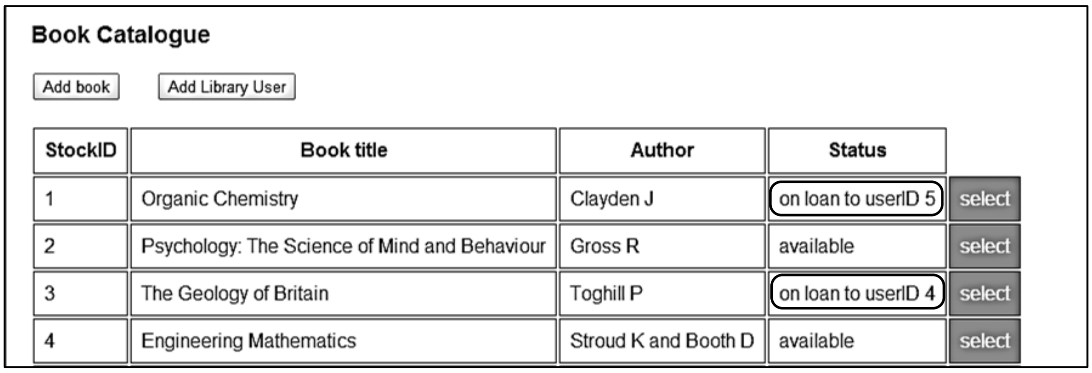
- If a book is on loan, a message is displayed to give the ID number of the borrower. The ID number is then stored as a variable '**userIDwanted**'.

- When a button is clicked to select a book, the **userID** of the borrower (if on loan) is transferred to the ***selectBook*** page in addition to the **stockID** of the book.

```
for (int i = 0; i < book.bookCount; i++)
{
    s += "<tr>";
    s += "<td>";
    s += book.bookObject[i].stockID;
    s += "</td>";
    s += "<td>";
    s += book.bookObject[i].title;
    s += "</td>";
    s += "<td>";
    s += book.bookObject[i].author;
    s += "</td>";
    s += "<td>";

    s += book.bookObject[i].status;

    status=book.bookObject[i].status;
    status = status.Trim();

    if (status == "available")
    {
        s += status;
        userIDwanted = "";
    }
    else
    {
        s += "on loan to userID " + status;
        userIDwanted = status;
    }

    s += "</td>";
    s += "<td bgcolor=#00AAEE>";
    s += "<a href='selectBook.aspx?stockID=";
    s += book.bookObject[i].stockID;

    s += "&userID=";
    s += userIDwanted;

    s += "'>select</a>";
    s += "</td>";
    s += "</tr>";
}
```

remove this line of code

Build and run the web page. Select several books from the ***Book Catalogue*** and use the drop down list of library users on the ***selectBook*** page to record loans to different users. Check that the userID values of the borrowers are displayed correctly.



Book Catalogue

Add book | Add Library User

StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	on loan to userID 5	select
2	Psychology: The Science of Mind and Behaviour	Gross R	available	select
3	The Geology of Britain	Toghill P	on loan to userID 4	select
4	Engineering Mathematics	Stroud K and Booth D	available	select

Close the browser and stop debugging. Our final task is to record the return of books by borrowers.

Go to the HTML code page of selectBook.aspx and add a second panel. If a book is on loan, this will display details of the borrower and has a button to record the book's return.

```
     <asp:Panel ID="Panel1" runat="server">
         Available for loan.
         <br />
         <br />
         Select borrower:
         <asp:DropDownList ID="DropDownList1" runat="server" Visible="True">
         </asp:DropDownList>
         <br />
         <br />
         <asp:Button ID="btnLoan" runat="server" Text="Record book loan"
             onclick="btnLoan_Click"/>
         <br />
    </asp:Panel>

     <asp:Panel ID="Panel2" runat="server">
        On loan to&nbsp;&nbsp;
        <asp:TextBox ID="txtBorrower" runat="server" Width="400px"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="btnReturnBook" runat="server" Text="Record book return"/>
    </asp:Panel>

</div>
```

We will arrange that only one of the panels is displayed at any time – ***Panel1*** in the case of a book which is available for borrowing, and ***Panel2*** in the case of a book which is already on loan.

Go to the C# code page ***selectBook. aspx.cs***. Add a string variable '**userIDwanted**' and a line of code to obtain the value for this variable, which was attached to the page URL as an extra parameter.

```
public partial class selectBook : System.Web.UI.Page
{
    String stockID;
    String userIDwanted;

    protected void Page_Load(object sender, EventArgs e)
    {
        stockID = Request.QueryString["stockID"];
        userIDwanted = Request.QueryString["userID"];

       if (DropDownList1.Items.Count < 1)
       {
            libraryUser.loadUsers();
```

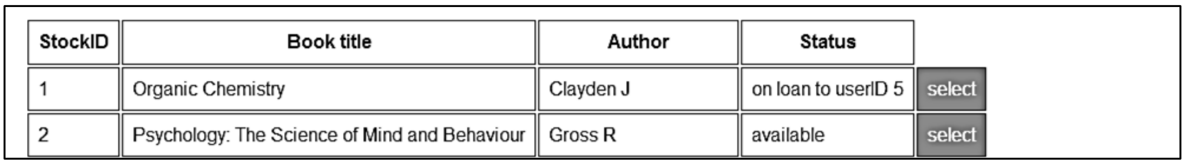
Move down the ***Page_Load( )*** method to locate the loop which finds the title, author and loan status of the selected book. After this loop, insert lines of code to check whether the book is available and then display the appropriate panel.

```
    for (int i = 0; i < book.bookCount; i++)
    {
        if (book.bookObject[i].stockID == n)
        {
            txtTitle.Text = book.bookObject[i].title;
            txtAuthor.Text = book.bookObject[i].author;
            status = book.bookObject[i].status;

        }
    }

    status = status.Trim();
    if (status == "available")
    {
        Panel1.Visible = true;
        Panel2.Visible = false;
    }
    else
    {
        Panel1.Visible = false;
        Panel2.Visible = true;
    }
```

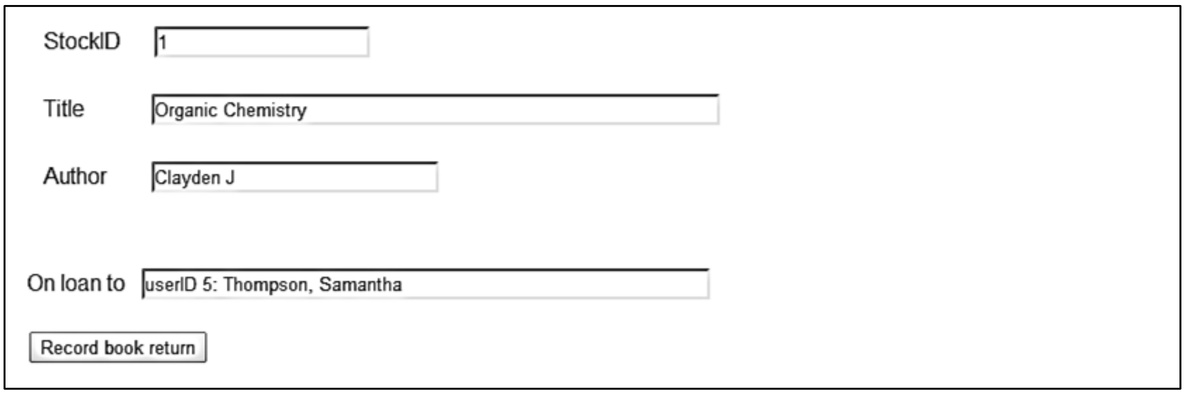
Code can then be added to the ***else*** block of the conditional structure, which operates in the case of a book on loan. The lines of code carry out a loop to find the correct user object, then transfer the **userID**, **surname** and **forename** to the text box.

```
if (status == "available")
{
    Panel1.Visible = true;
    Panel2.Visible = false;
}
else
{
    Panel1.Visible = false;
    Panel2.Visible = true;
    int userID;
    for (int i = 0; i < libraryUser.userCount; i++)
    {
        userID = libraryUser.userObject[i].userID;
        if (userID == Convert.ToInt16(userIDwanted))
        {
            string surname = libraryUser.userObject[i].surname;
            string forename = libraryUser.userObject[i].forename;
            string s = "userID " + userIDwanted + ": " + surname + ", "
                 + forename;
            txtBorrower.Text = s;
        }
    }
}
```

Build and run the web site. Go to the Book Catalogue page and select a book which is on loan. Check that the selectBook page opens with only the 'Record book return' panel visible, and the details of the borrower displayed correctly.

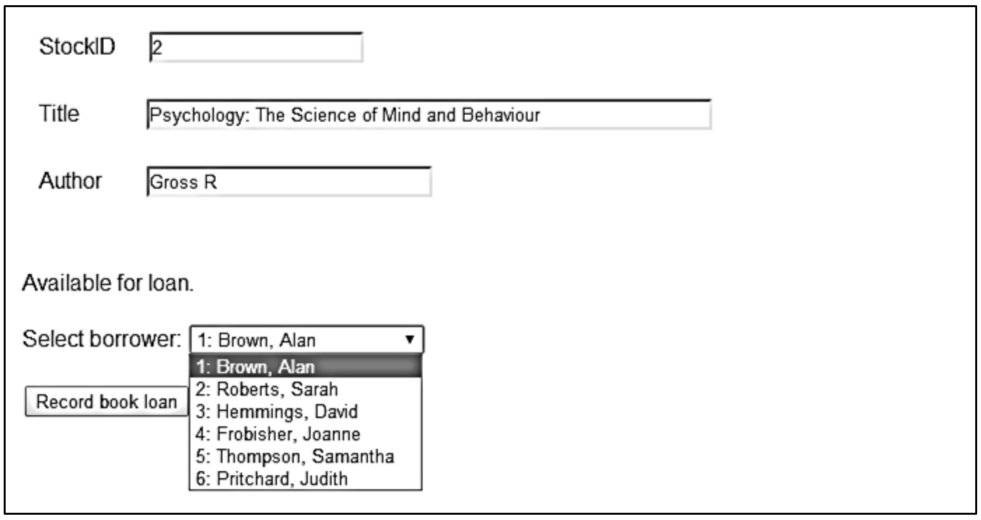


StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	on loan to userID 5	select
2	Psychology: The Science of Mind and Behaviour	Gross R	available	select



StockID	1
Title	Organic Chemistry
Author	Clayden J
On loan to	userID 5: Thompson, Samantha

Check also that the correct panel is displayed for a book not on loan.



StockID 2

Title Psychology: The Science of Mind and Behaviour

Author Gross R

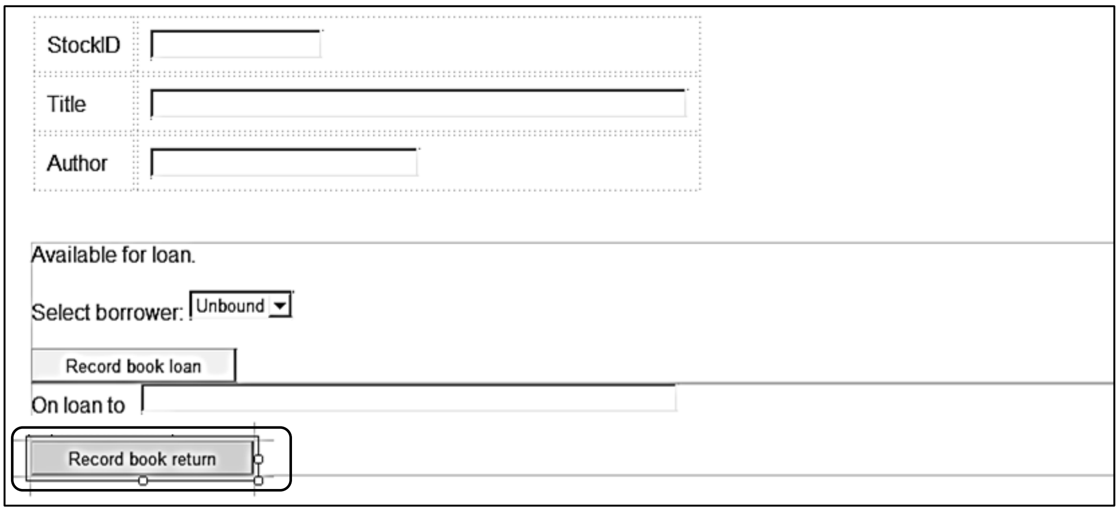
Available for loan.

Select borrower: 1: Brown, Alan

1: Brown, Alan
2: Roberts, Sarah
3: Hemmings, David
4: Frobisher, Joanne
5: Thompson, Samantha
6: Pritchard, Judith

Close the web browser, return to Visual Studio and stop debugging.

It simply remains to set up program code for the '***Record book return***' button. Go to the Design view of ***selectBook.aspx*** and double click the button to create a ***button_click*** method.



StockID

Title

Author

Available for loan.

Select borrower: Unbound

On loan to

Add lines of code to the button_click method which will:

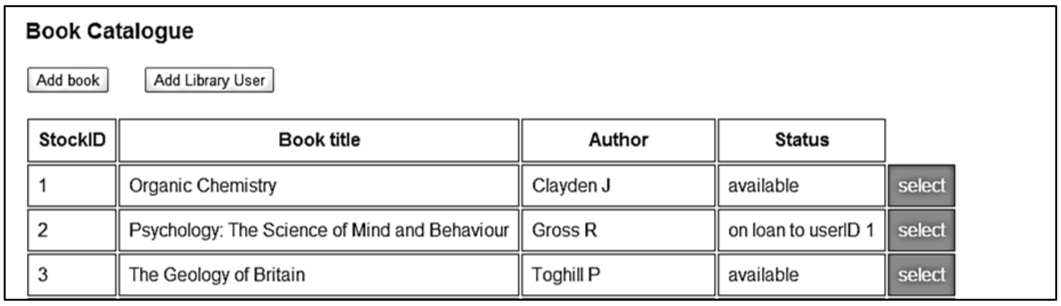
- Call a ***recordReturn( )*** method which we will add to the ***book*** class file.
- Reload the book records, so that the returned book is displayed as '**available**'.
- Go to the ***Book Catalogue*** page.

```
protected void btnReturnBook_Click(object sender, EventArgs e)
{
    book.recordReturn(Convert.ToInt16(stockID));
    book.loadBooks();
    Response.Redirect("bookCatalog.aspx");
}
```

Move to the book.cs class file and add a ***recordReturn( )*** method. This opens the database, then uses an SQL command to reset the loan status of the selected book record to '**available**'.

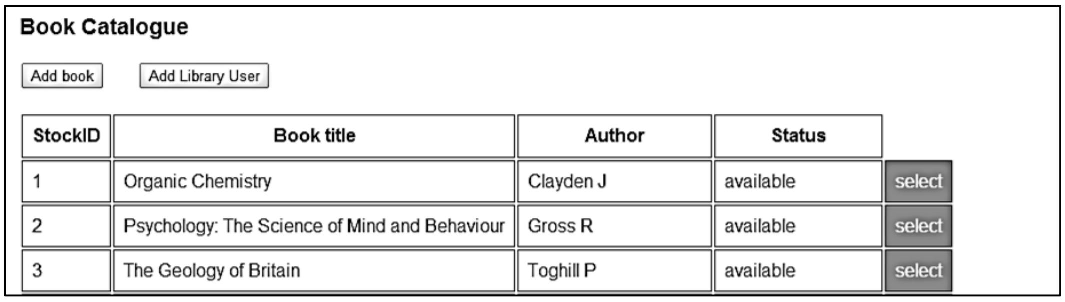
```
public string author { get; set; }
public string status { get; set; }

public static string databaseLocation = "C:\\WEB APPLICATIONS\\Library.mdf;";

public static void recordReturn(int stockID)
{
    SqlConnection cnTB = new SqlConnection(@"Data Source=.\SQLEXPRESS;
         AttachDbFilename=" + databaseLocation + "Integrated Security=True;
         Connect Timeout=30; User Instance=True");
    try
    {
        cnTB.Open();
        SqlCommand cmBooks = new SqlCommand();
        cmBooks.Connection = cnTB;
        cmBooks.CommandType = CommandType.Text;
        cmBooks.CommandText = "UPDATE book SET  status ='available'
              WHERE bookStockID='" + stockID + "'";
        cmBooks.ExecuteNonQuery();
        cnTB.Close();
    }
    catch
    {

    }
}
```

Build and run the web site. Systematically test the program by issuing books on loan to different library users, then checking that the return of the books is recorded correctly.



Book Catalogue

StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	available	select
2	Psychology: The Science of Mind and Behaviour	Gross R	on loan to userID 1	select
3	The Geology of Britain	Toghill P	available	select



Book Catalogue

StockID	Book title	Author	Status	
1	Organic Chemistry	Clayden J	available	select
2	Psychology: The Science of Mind and Behaviour	Gross R	available	select
3	The Geology of Britain	Toghill P	available	select