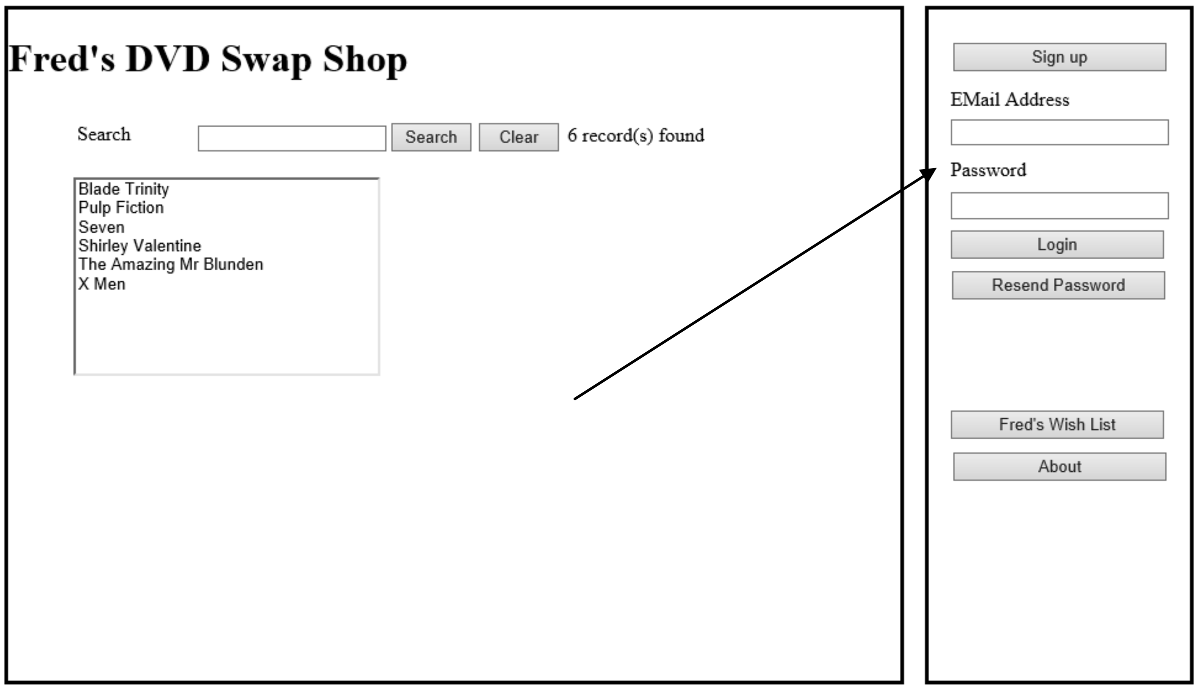
HTML 5 Form Processing

In this session we will explore the way that data is passed from an HTML 5 form to a form processor and back again.

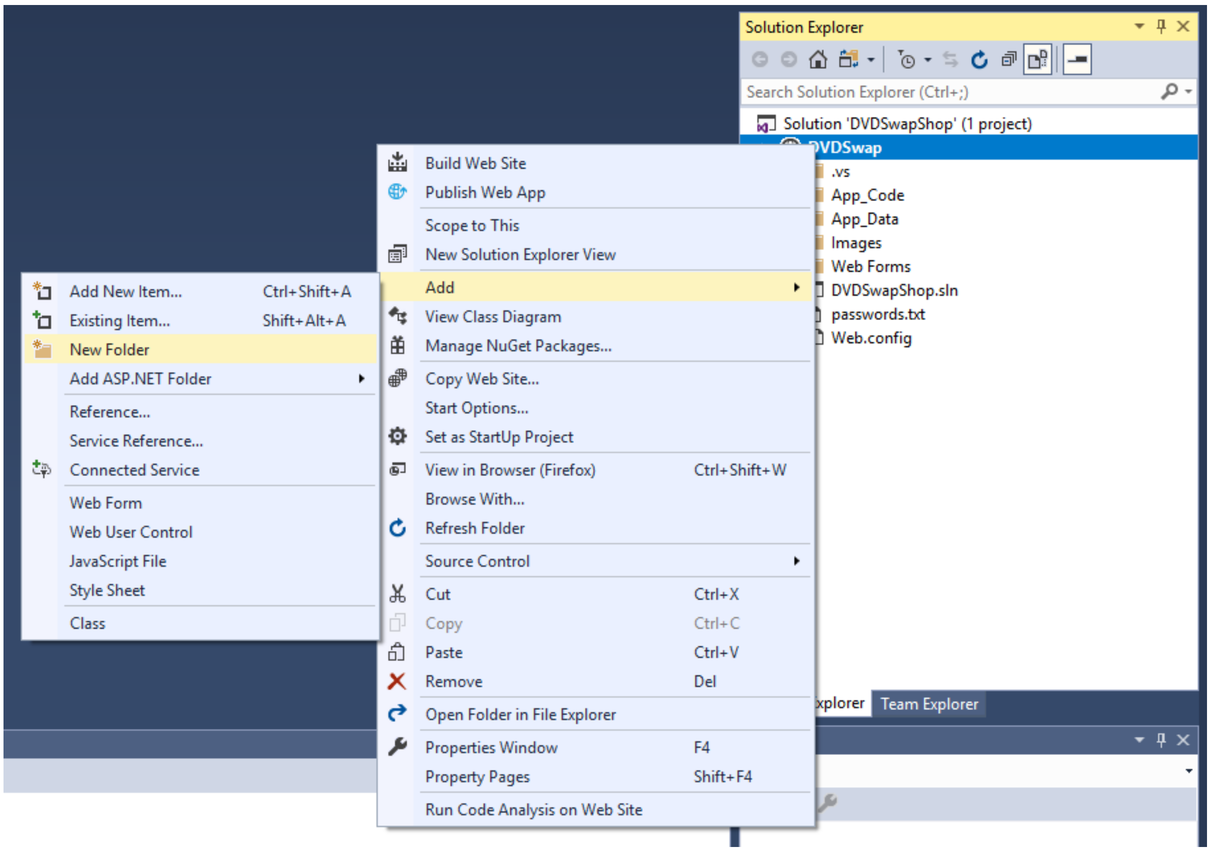
We are going to start by looking at the functionality of part of the DVD Swap and then revise the way that this feature is implemented.

This is the start of a process where we will review the technology that drives the DVD Swap Shop and also look at ways we may use alternatives and also improve the design.

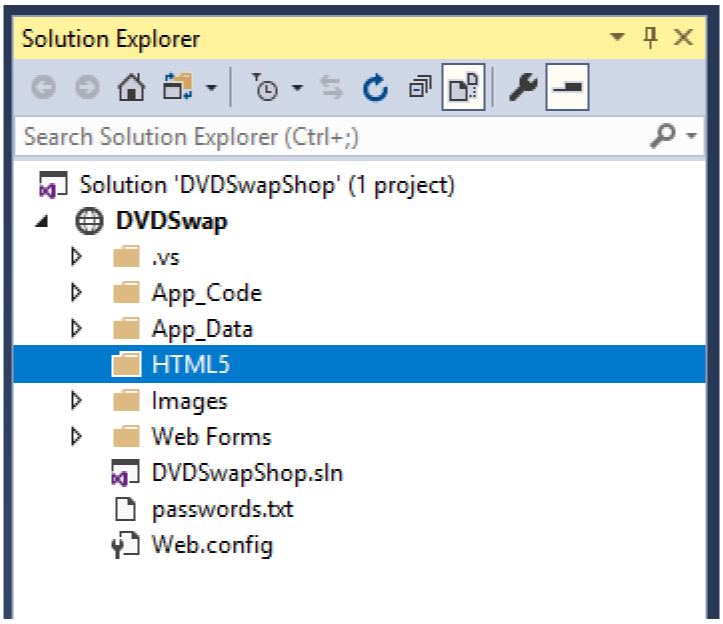
On the main page of the application we find a login form in the top right corner.



To replicate the functionality we are going to create a second presentation layer within the web site. Within your copy of the DVD Swap Shop create a new folder called HTML5…



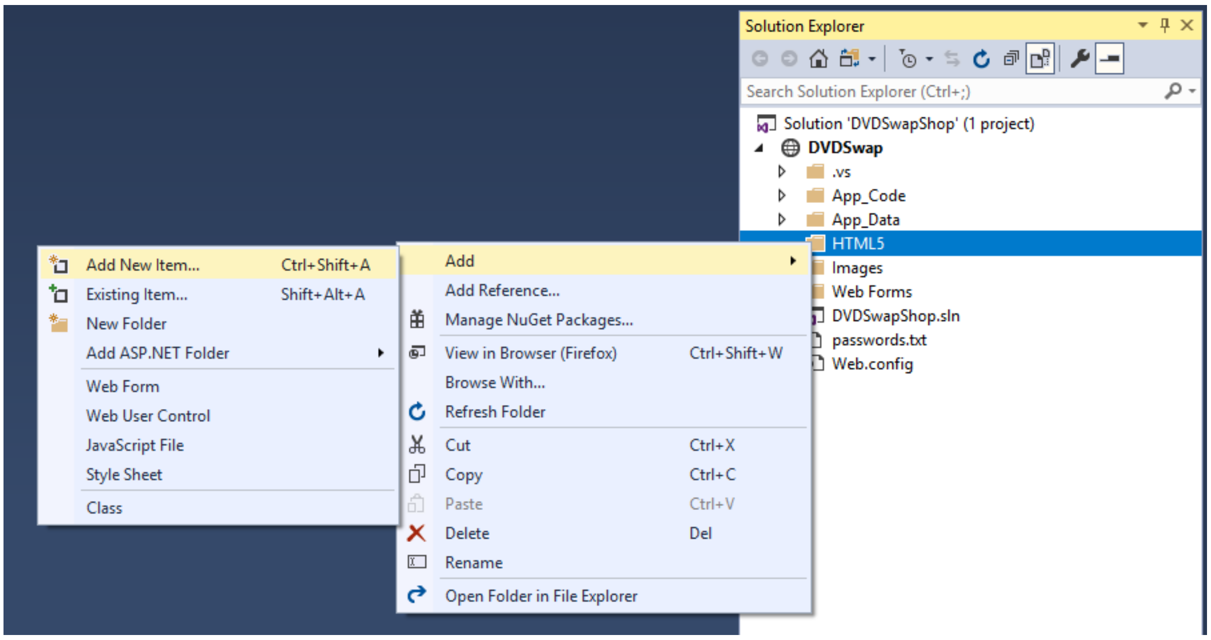
Name the new folder HTML5



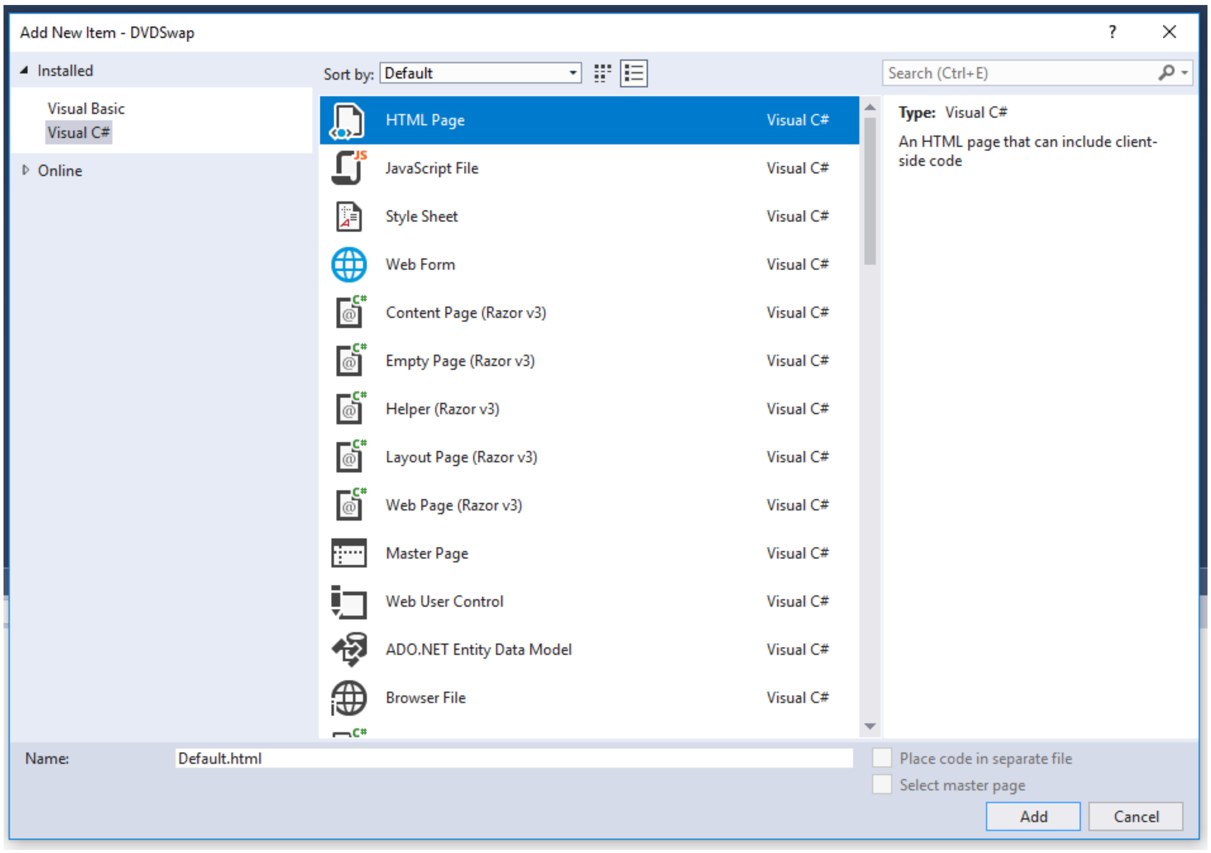
There are two components we want to look at for this example.

1. An HTML 5 page allowing the user to enter their user name and password
2. An ASPX form processor allowing us to capture the data and display some output

The first step in Visual Studio is to create a new HTML 5 page. In the Solution explorer right click on the new folder and select Add New Item…



We shall name the page Default.html…



Add New Item - DVDSwap
Installed
Visual Basic
Visual C#
Online
Sort by: Default
Search (Ctrl+E)
HTML Page Visual C#
JavaScript File Visual C#
Style Sheet Visual C#
Web Form Visual C#
Content Page (Razor v3) Visual C#
Empty Page (Razor v3) Visual C#
Helper (Razor v3) Visual C#
Layout Page (Razor v3) Visual C#
Web Page (Razor v3) Visual C#
Master Page Visual C#
Web User Control Visual C#
ADO.NET Entity Data Model Visual C#
Browser File Visual C#
Type: Visual C#
An HTML page that can include client-side code
Name: Default.html
Place code in separate file
Select master page
Add
Cancel

The new page is called Default.html for a good reason. When a user first navigates to a site having entered the URL the server will typically open up the web site's home page.

If we are using web forms, the home-page would be called Default.aspx. In the case of using a Microsoft server using HTML the home-page is called Default.html. If we were using a non Microsoft server such as Apache, the home-page would be called index.html.

Having created the home-page let's have a look at the resulting HTML 5.

```html
<!DOCTYPE html>
<html>
<head>
    <title></title>
       <meta charset="utf-8" />
</head>
<body>

</body>
</html>
```

Let's break this down step by step.

The first thing to notice are the tags that define what is where within the page.

Tags are the items of text contained within the braces <>.

Tags enclose mark-up which is used to identify different sections of the web page.

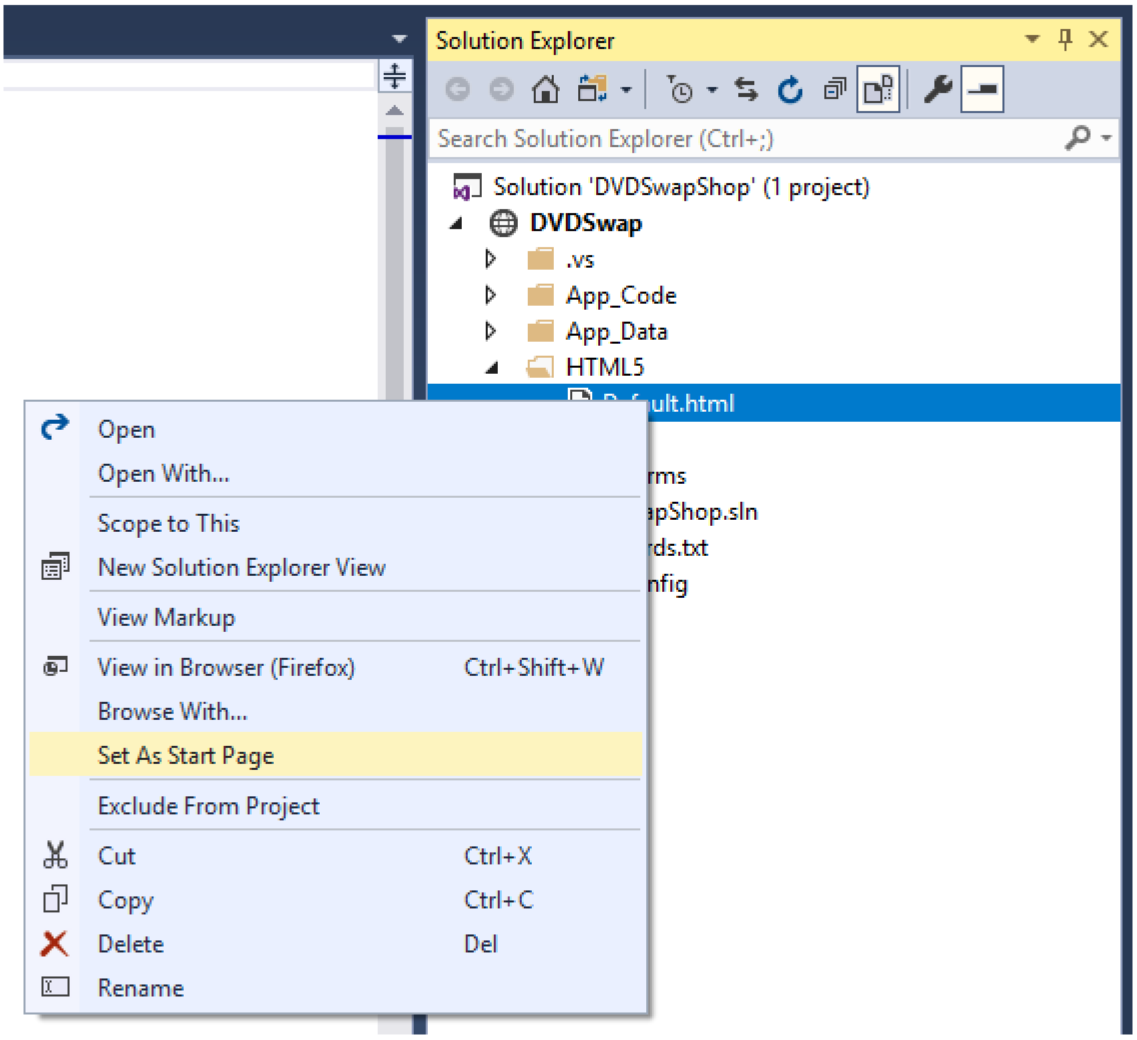
For example the title tag may be used to add a title to your web-page.

```html
    <title></title>
```

Alter the page title to DVD Swap Shop like so…

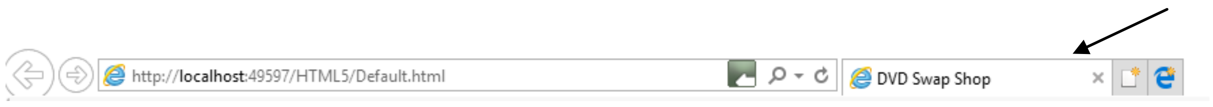
```html
<title>DVD Swap Shop</title>
```

To make this new HTML5 page the start up page for the program, right click on the page and select "set as start page".



This will make the program use the new HTML5 presentation rather than the Web Form one (you may switch back in the same way.

Run the program to view the page in the browser…



Not terribly exciting but it makes the point.

In HTML 5, tags are case sensitive, so <TITLE> is a different tag to <Title> and <title>!

If a tag is opened in HTML 5 then it must be closed.

We can see the opening and closing tags for title.

```
<title>
```

Opens the tag.

```
</title>
```

Closes the tag.

There are also some other important parts of the page structure.

The first line called the doctype, `<!DOCTYPE html>` tells the browser opening the page that this is an HTML 5 page.

Historically there have been a number of versions of HTML and all behave in slightly different ways.

You can switch between different versions by changing the doctype

For example we could set the version of HTML to HTML 4.01 strict…

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Much of the mark-up would be the same but there would be differences.

We are interested in the newest version though so won't worry about other versions but you can see the range of doctypes here…

https://www.w3schools.com/tags/tag_doctype.asp

The next tag of interest is the html tag…

```
<html></html>
```

The html tag is used to indicate where the start and end of the HTML page is. The html tag must follow the doctype and enclose all the other content in the document like so...

```
<!DOCTYPE html>
<html>
<head>
    <title>My Web Page</title>
       <meta charset="utf-8" />
</head>
<body>

</body>
</html>
```

The `<head></head>` tag marks the header information for the page. As we saw above one tag that lives here is the title tag. In this example we see another tag that tells the browser what character set the page will be using.

```
<meta charset="utf-8" />
```

The character set encoding is an important consideration in making sure that the file is read correctly.

As you are probably aware everything we see on a computer is internally represented as binary data. When it comes to representing text data we normally associate a numeric value with the letter from the character set.

For example the ASCII character set uses the following codes to represent the letters A to F

65	Uppercase A
66	Uppercase B
67	Uppercase C

68	Uppercase D
69	Uppercase E
70	Uppercase F

In UNICODE (Another system for representing characters we use the following codes to represent A to F)

41	Uppercase A
42	Uppercase B
43	Uppercase C
44	Uppercase D
45	Uppercase E
46	Uppercase F

So if our browser received the following character codes...

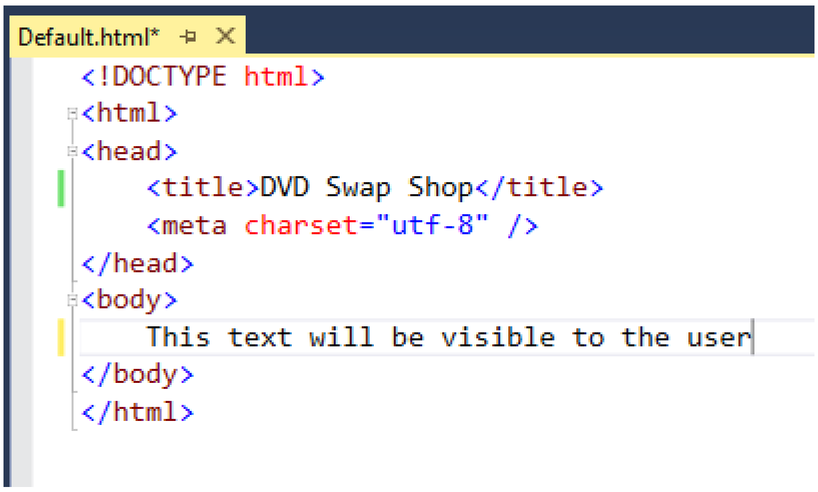
46, 41, 44, 45

We need to know which character set the data is encoded in before we can convert the numeric codes back to letters.

UTF-8 tells us that the text is encoded using UCS Transformation Format — 8-bit (UCS stands for Universal Character Set) which means that we know how to convert the codes to letters at both ends. UTF-8 has become the dominant character set for the World Wide Web.

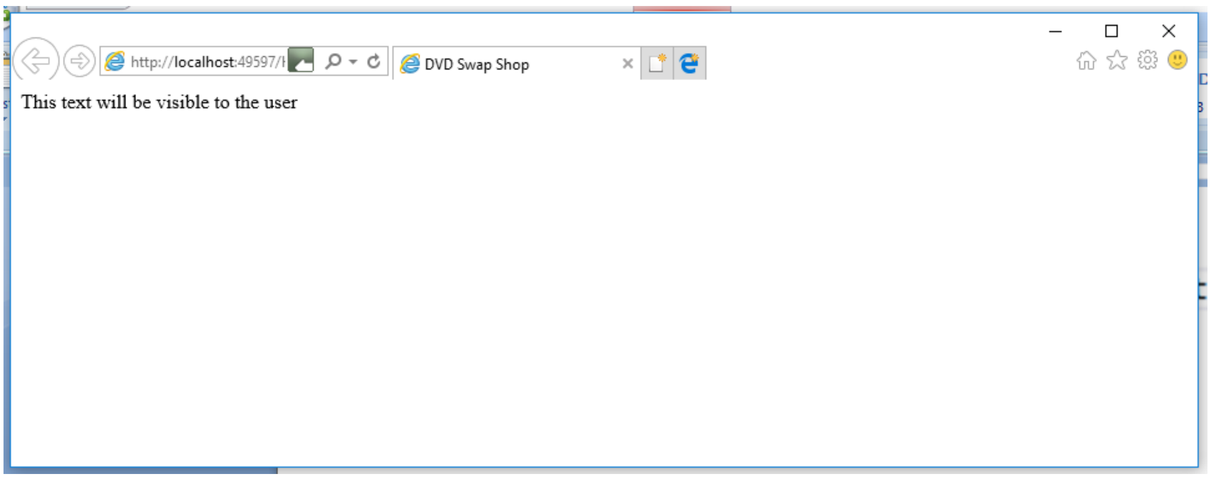
The last tag to look at is the `<body></body>` tag. This tag marks up where the page text that the user gets to see begins and ends.

Modify the web page like so then run the page in the browser.



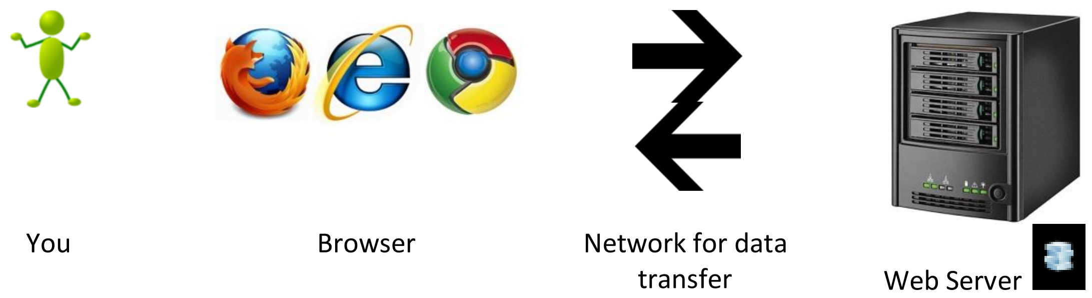
```
<!DOCTYPE html>
<html>
<head>
    <title>DVD Swap Shop</title>
    <meta charset="utf-8" />
</head>
<body>
    This text will be visible to the user
</body>
</html>
```

You should see something like this...



This text will be visible to the user

We have just completed our first run through the client server model…

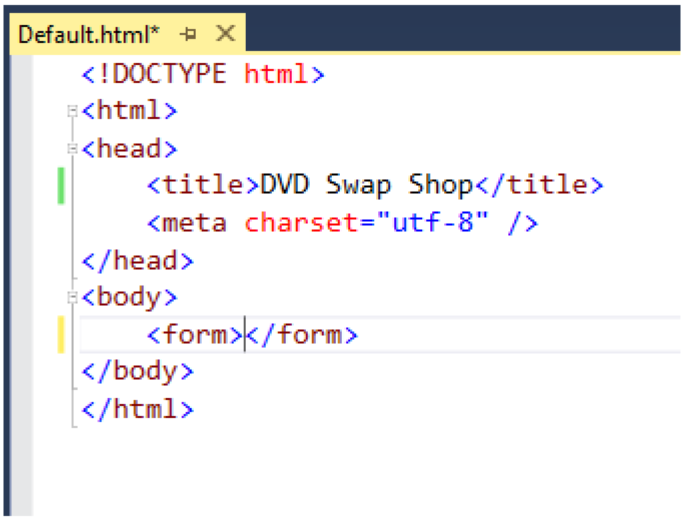


The user (you) has just opened a page in the browser. The browser has just sent a request to the (local) server. The server has found the default document for the site Default.html and sent it over the "network" back to the browser so that you can see it.

Creating a Form

As with everything else in HTML 5 to create a form within the page we will need to use a tag to define it.

The following HTML 5 will create the form nicely (let's also get rid of the text we just created)…

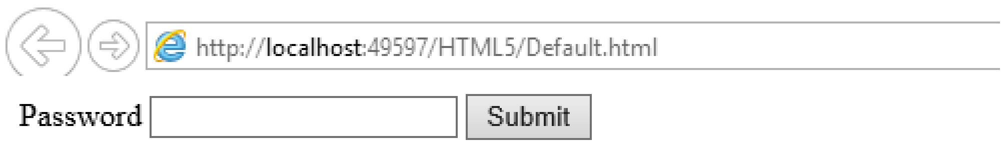


```
<!DOCTYPE html>
<html>
<head>
    <title>DVD Swap Shop</title>
    <meta charset="utf-8" />
</head>
<body>
    <form></form>
</body>
</html>
```

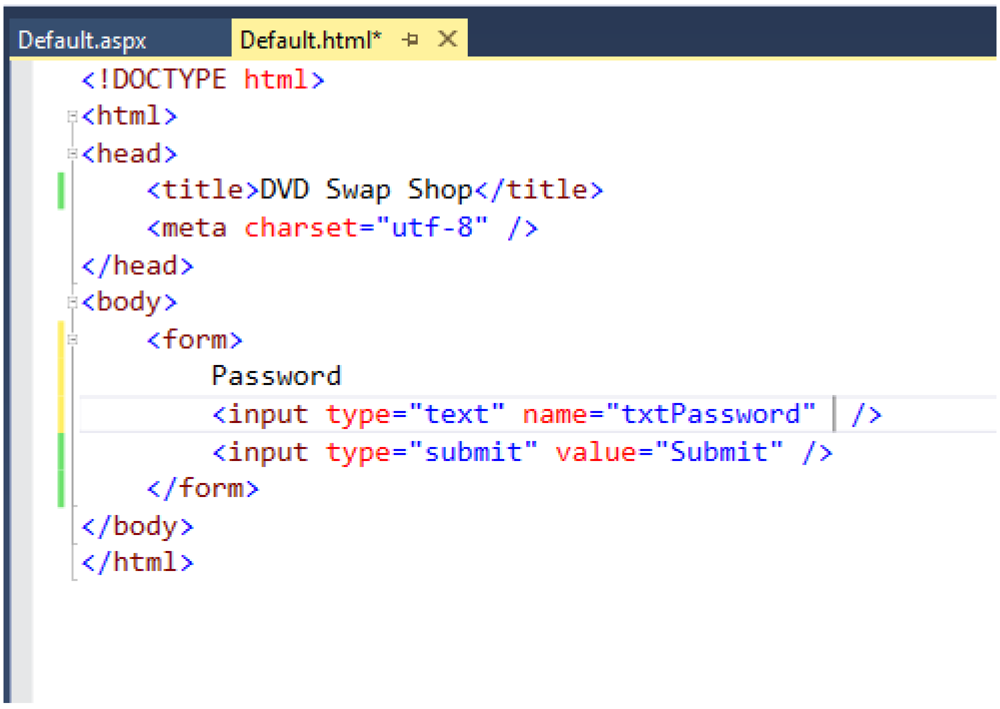
That's very nice but so far it does nothing useful.

The next step is to add some controls to the form so that the user can do something.

We will create two controls, a text box to type some stuff and a button to submit the page to the server.



To do this we need to modify the form code like so…



```
<!DOCTYPE html>
<html>
<head>
    <title>DVD Swap Shop</title>
    <meta charset="utf-8" />
</head>
<body>
    <form>
        Password
        <input type="text" name="txtPassword" />
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
```

View the page in the browser and it should look like the page above.

Now we get to the exciting bit. Run the page and type some text into the text box and press submit. What happens?

Absolutely nothing!

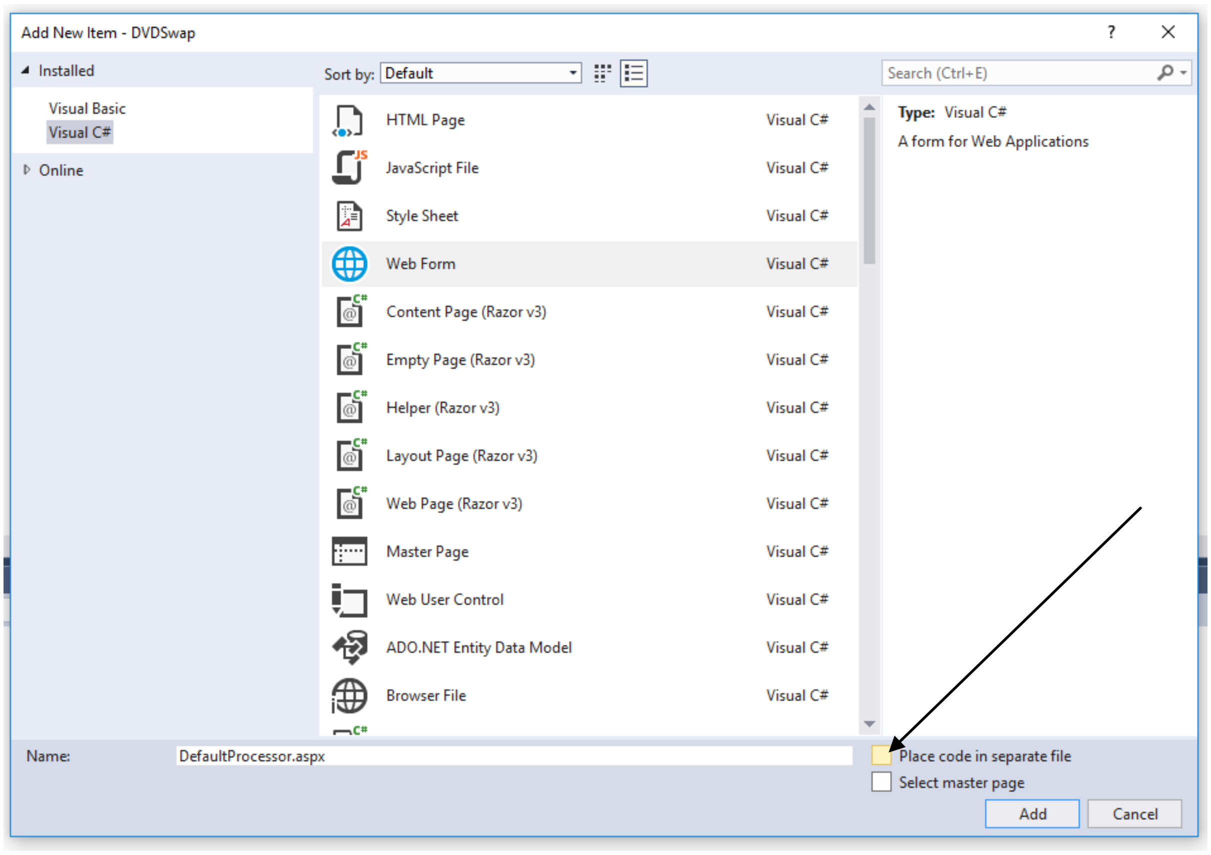
So what is the problem?

Adding the Form Processor

For an HTML form to work, we need to add an extra file called a form processor. The form processor accepts the data from the browser at the server and examines the data in the form.

We place code in the form processor that does this job and could be written in a number of different languages such as PHP or Java. In this case we will create the form processor in C# since we are using Microsoft technology.

In Visual Studio create a new ASPX page with the following settings…


Add New Item - DVDSwap

Installed: Visual Basic, Visual C#

Online

Sort by: Default

HTML Page	Visual C#
JavaScript File	Visual C#
Style Sheet	Visual C#
Web Form	Visual C#
Content Page (Razor v3)	Visual C#
Empty Page (Razor v3)	Visual C#
Helper (Razor v3)	Visual C#
Layout Page (Razor v3)	Visual C#
Web Page (Razor v3)	Visual C#
Master Page	Visual C#
Web User Control	Visual C#
ADO.NET Entity Data Model	Visual C#
Browser File	Visual C#

Type: Visual C#

A form for Web Applications

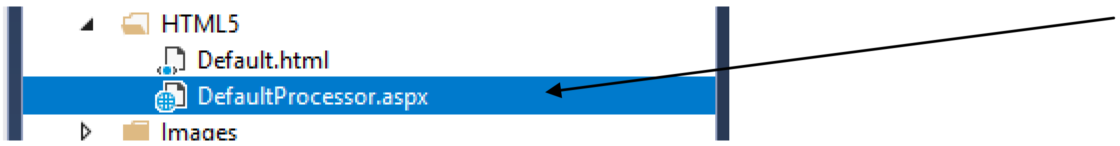
Name: DefaultProcessor.aspx

Place code in separate file

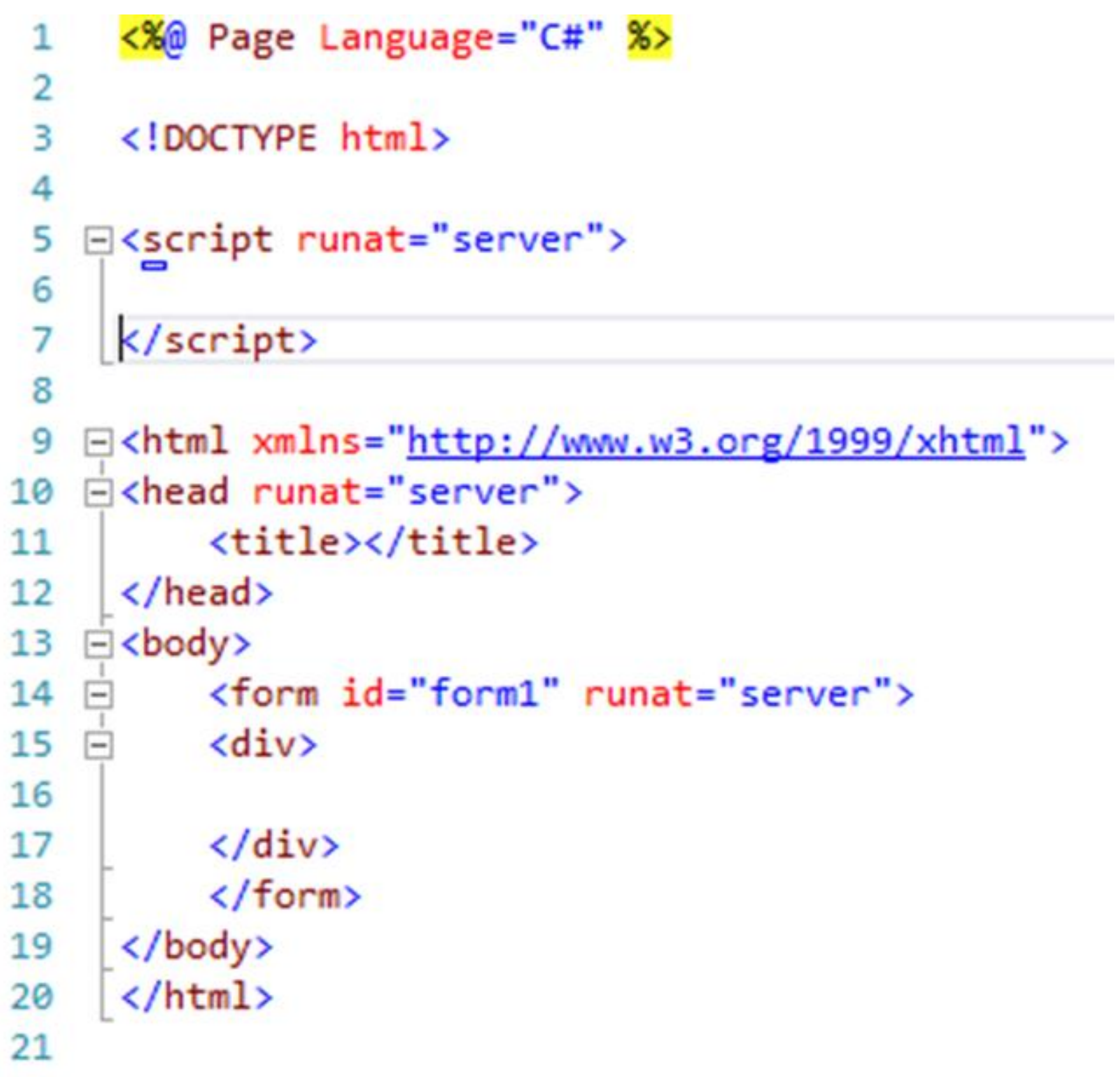
Select master page

One thing we will do differently is to un-tick the box that says “Place code in separate file”.

Rather than creating two files for the ASPX file we will have just one like so…



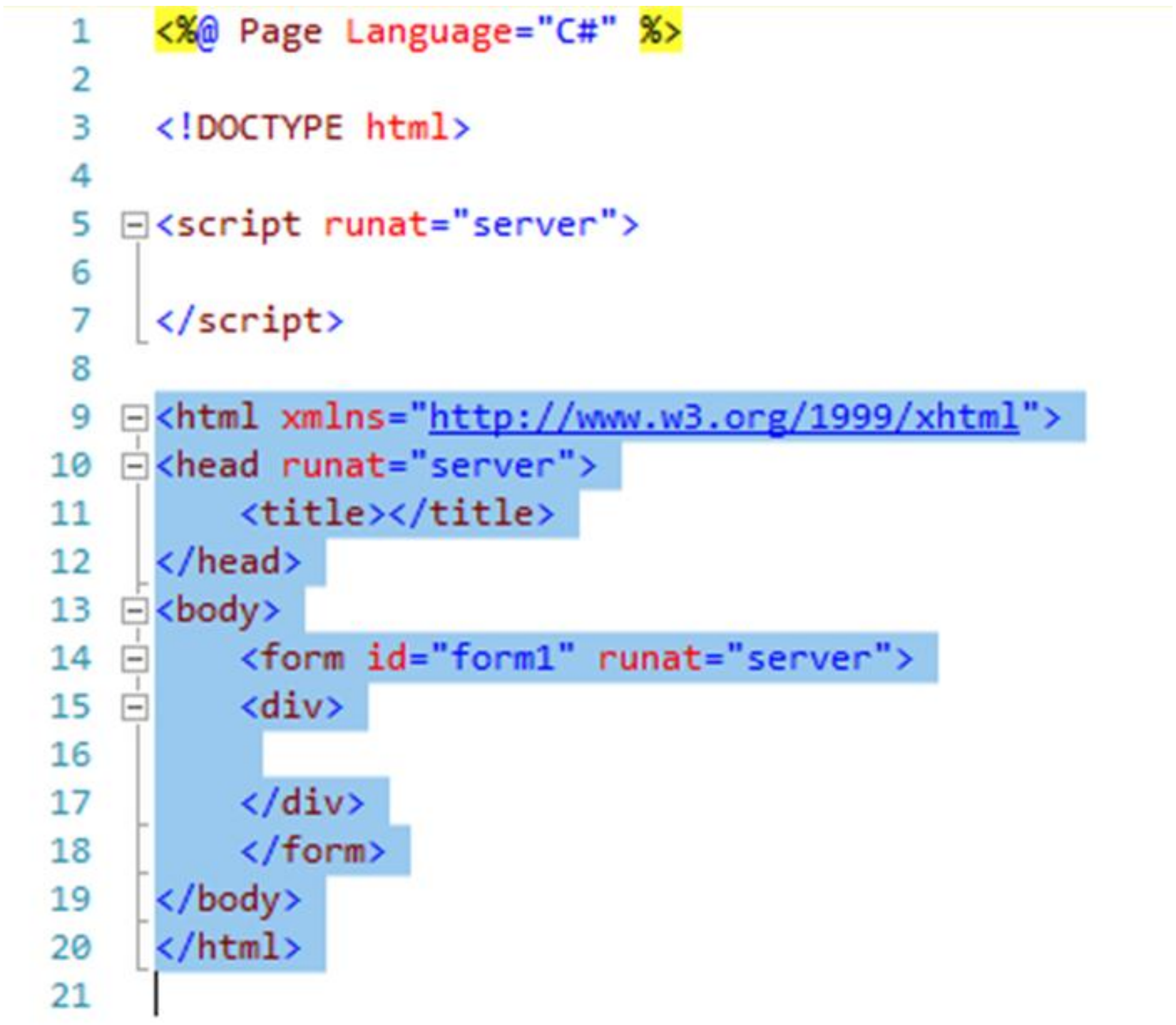
The code for the new processor currently looks like this…



```
<%@ Page Language="C#" %>

<!DOCTYPE html>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>

```

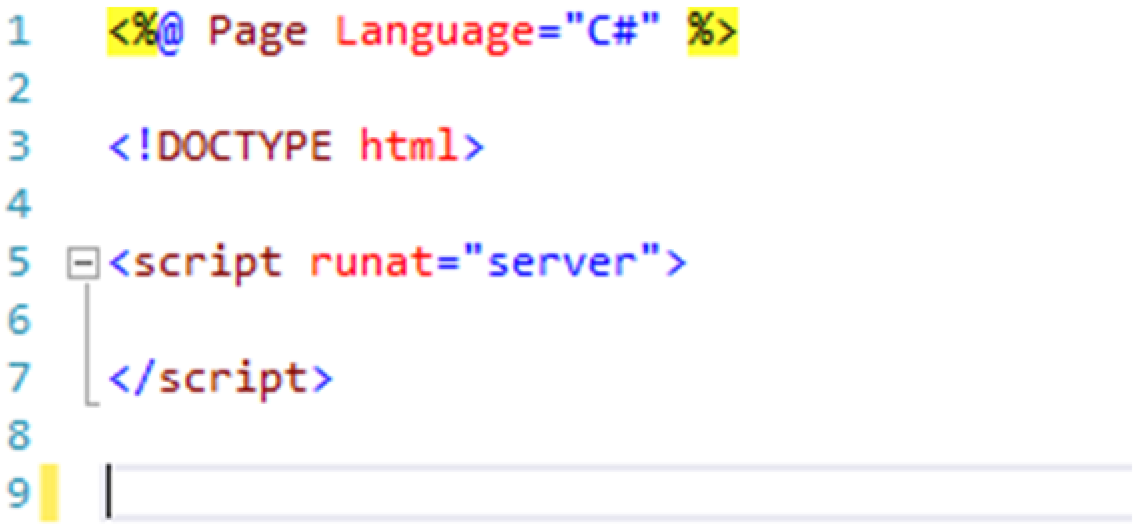
Since this is still a web form there is a load of HTML code that we don't need.

The section of code defining the web form can be deleted…



```
<%@ Page Language="C#" %>

<!DOCTYPE html>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>

```

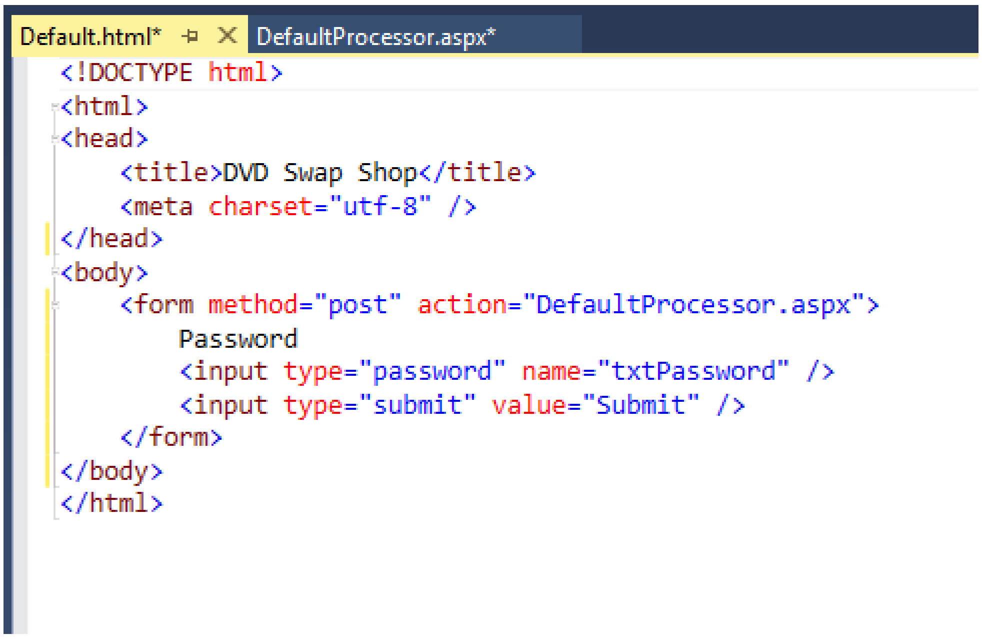
Leaving just the script component of the form processor…



```
<%@ Page Language="C#" %>

<!DOCTYPE html>

<script runat="server">

</script>

```

Now that we have the form processor set up we can make the link between this file (on the server) and the HTML page (from our browser)

Go back to the HTML page and modify the form tag like so…



```
<!DOCTYPE html>
<html>
<head>
    <title>DVD Swap Shop</title>
    <meta charset="utf-8" />
</head>
<body>
    <form method="post" action="DefaultProcessor.aspx">
        Password
        <input type="password" name="txtPassword" />
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
```

We shall look at `method="post"` once we have everything working. The section `action="DefaultProcessor.aspx"` tells the form to deliver the data to the form processor DefaultProcessor.aspx.

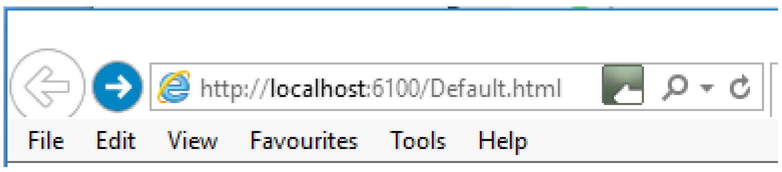
The sequence is:

1. The browser requests the default page from the server
2. The server delivers the page Default.html to the browser
3. The user types some data into the text box and presses submit
4. The page is sent back to the server
5. The server receives the page and passes the form data to the processor DefaultProcessor.aspx

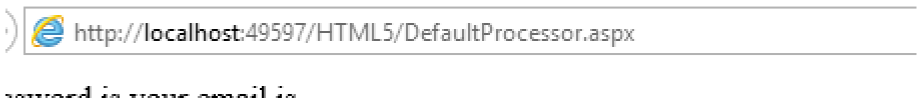
As before, view the page in the browser and see what happens.

As above, nothing happens – but not quite.

Notice that the first page that opens is the default page Default.html



Once you press submit the URL should have re-directed to the form processor…



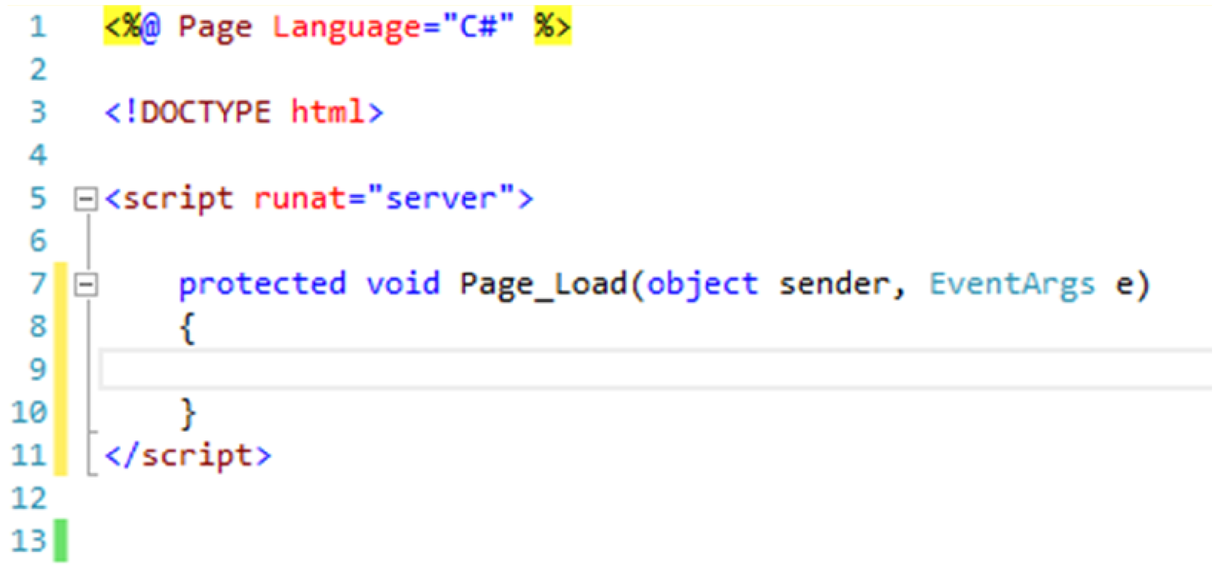
But apart from the page re-direction nothing visible has happened. Quite a lot has actually happened but we are not yet in a position to make use of the data in the HTML form.

To fix the situation we need to add some C# code to the form processor that examines the data in the HTML form and does something with it.

Adding Code to the Form Processor

When we press the submit button at the browser the data in the form is delivered to the server for processing. The server looks at the HTML for the form and sees that the form processor is called Default.aspx. The server opens this file which triggers an event called the load event. The load event is triggered each time the processor is opened and this is requires a function so that we can link some code to the event.

Modify the script section of the form processor to add an event-handler function for the load event like so…

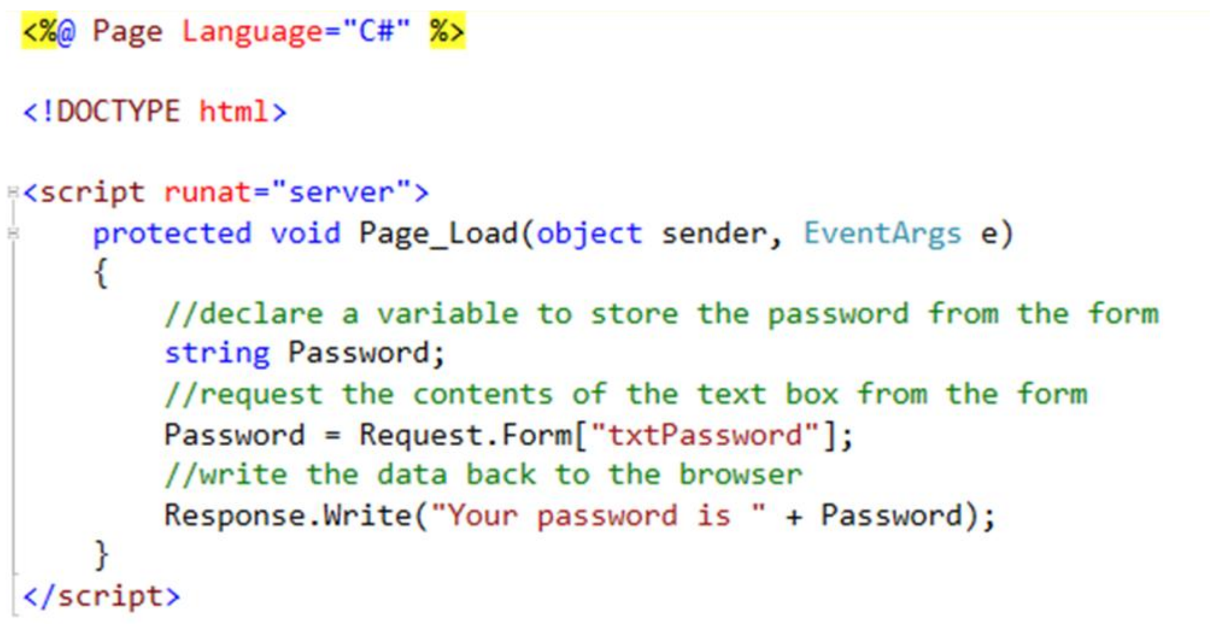


```
<%@ Page Language="C#" %>

<!DOCTYPE html>

<script runat="server">

    protected void Page_Load(object sender, EventArgs e)
    {

    }
</script>

```

This function will run every time the server loads the form processor.

Now we need some code to get the data out of the form to see what we have got.

Add the following code to the event handler function for the load event…



```
<%@ Page Language="C#" %>

<!DOCTYPE html>

<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        //declare a variable to store the password from the form
        string Password;
        //request the contents of the text box from the form
        Password = Request.Form["txtPassword"];
        //write the data back to the browser
        Response.Write("Your password is " + Password);
    }
</script>
```

So what is going on here?

First we are declaring a string to allow us to store the data entered by the user in the RAM.

Next we are performing a request to the form.

```
        //request the contents of the text box from the form
        Password = Request.Form["txtPassword"];
```

Request.Form looks at the incoming data and looks for an HTML control called txtPassword. Assuming it finds that control the assignment operator copies its data to the variable Password.

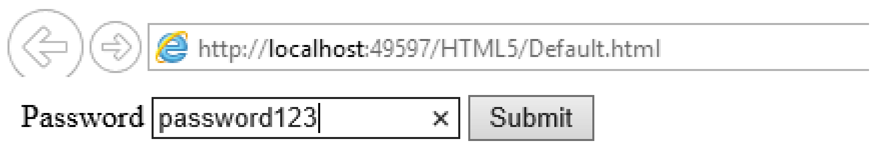
Response.Write allows us to quickly send data back to the browser so that we can see what has happened. In the case…

```
        //write the data back to the browser
        Response.Write("Your password is " + Password);
```

Should display whatever data was originally submitted in the HTML form.

Try it and see what happens.

You should first see your web page containing the HTML form.



http://localhost:49597/HTML5/Default.html

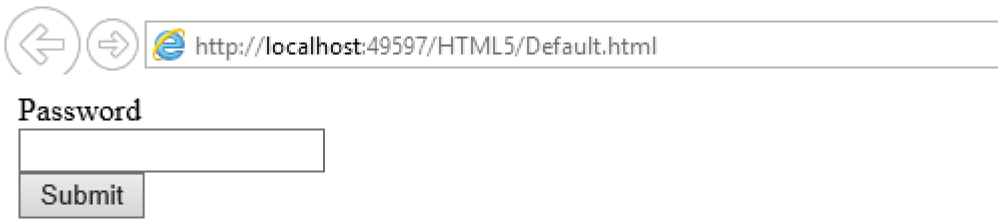
Password password123 Submit

Type some data and press submit. You should see the message displayed by the form processor.

To tidy up the form add some breaks to start different sections on a new line.

```
<form method="post" action="DefaultProcessor.aspx">
    Password
    <br />
    <input type="text" name="txtPassword" />
    <br />
    <input type="submit" value="Submit" />
</form>
```

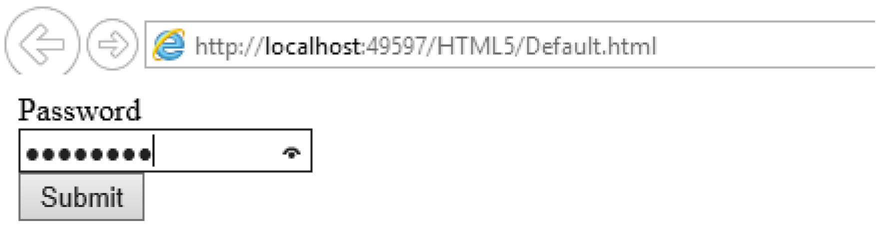
Still not the most exciting interface in the world but heading in the right direction…



Notice also what happens if we now change the input type of the password box from “text” to “password”…

```
<form method="post" action="DefaultProcessor.aspx">
    Password
    <br />
    <input type="password" name="txtPassword" />
    <br />
    <input type="submit" value="Submit" />
</form>
```

This allows us to create the following more appropriate behaviour for password input.



Finishing off

Now that you have a password field set up for the form modify the design such that it includes the option for the user to enter their email address. Also modify the form processor such that it displays both the user name and password entered.