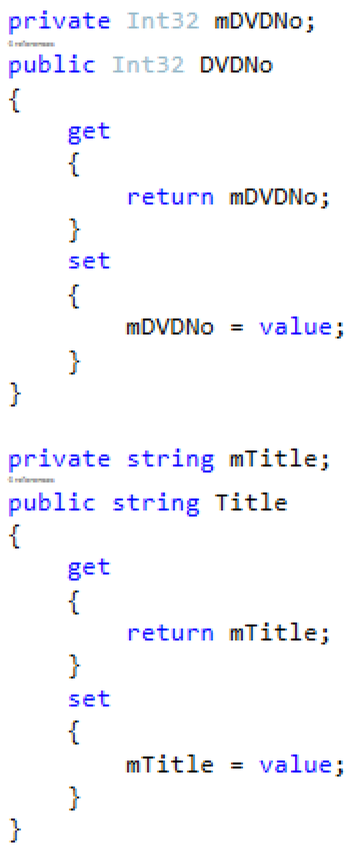
Dynamic Data Hyperlinks and Images

For this lab work we will concentrate on some examples that make use of dynamic content generation using data from the middle layer, hyperlinks and images.

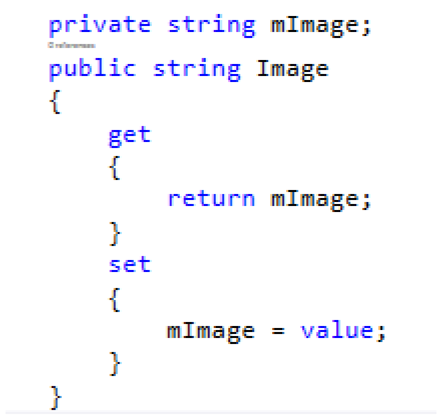
As with the table example last week we don’t want to spend lots of time creating the data layer and middle layer code, so we shall create some middle layer code that simulates the effect we want to see.

Last week we created a new class called clsDVD and added two public properties for DVDNo and Title like so…



```
private Int32 mDVDNo;
public Int32 DVDNo
{
    get
    {
        return mDVDNo;
    }
    set
    {
        mDVDNo = value;
    }
}

private string mTitle;
public string Title
{
    get
    {
        return mTitle;
    }
    set
    {
        mTitle = value;
    }
}
```

To this class we will add a new public property called Image, like so…



```
private string mImage;
0 references
public string Image
{
    get
    {
        return mImage;
    }
    set
    {
        mImage = value;
    }
}
```

We now have three attributes in our class…

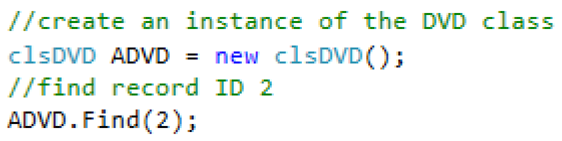
clsDVD
Int32 DVDNo String Title String Image

We will now add a public method that allows us to find a specific DVD which accepts a single parameter specifying the primary key of the DVD we want to find.

For example if this is our data…

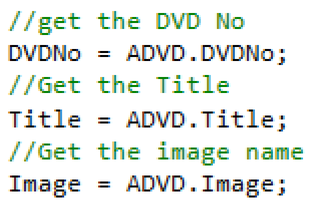
DVDNo	Title	Image
1	Raiders of the Lost Ark	raiders.jpg
2	Star Wars	starwars.jpg
3	The Disaster Artist	disasterartist.jpg

We might use the following C# code…



```
//create an instance of the DVD class
clsDVD ADVD = new clsDVD();
//find record ID 2
ADVD.Find(2);
```

If we use the following code…



```
//get the DVD No
DVDNo = ADVD.DVDNo;
//Get the Title
Title = ADVD.Title;
//Get the image name
Image = ADVD.Image;
```

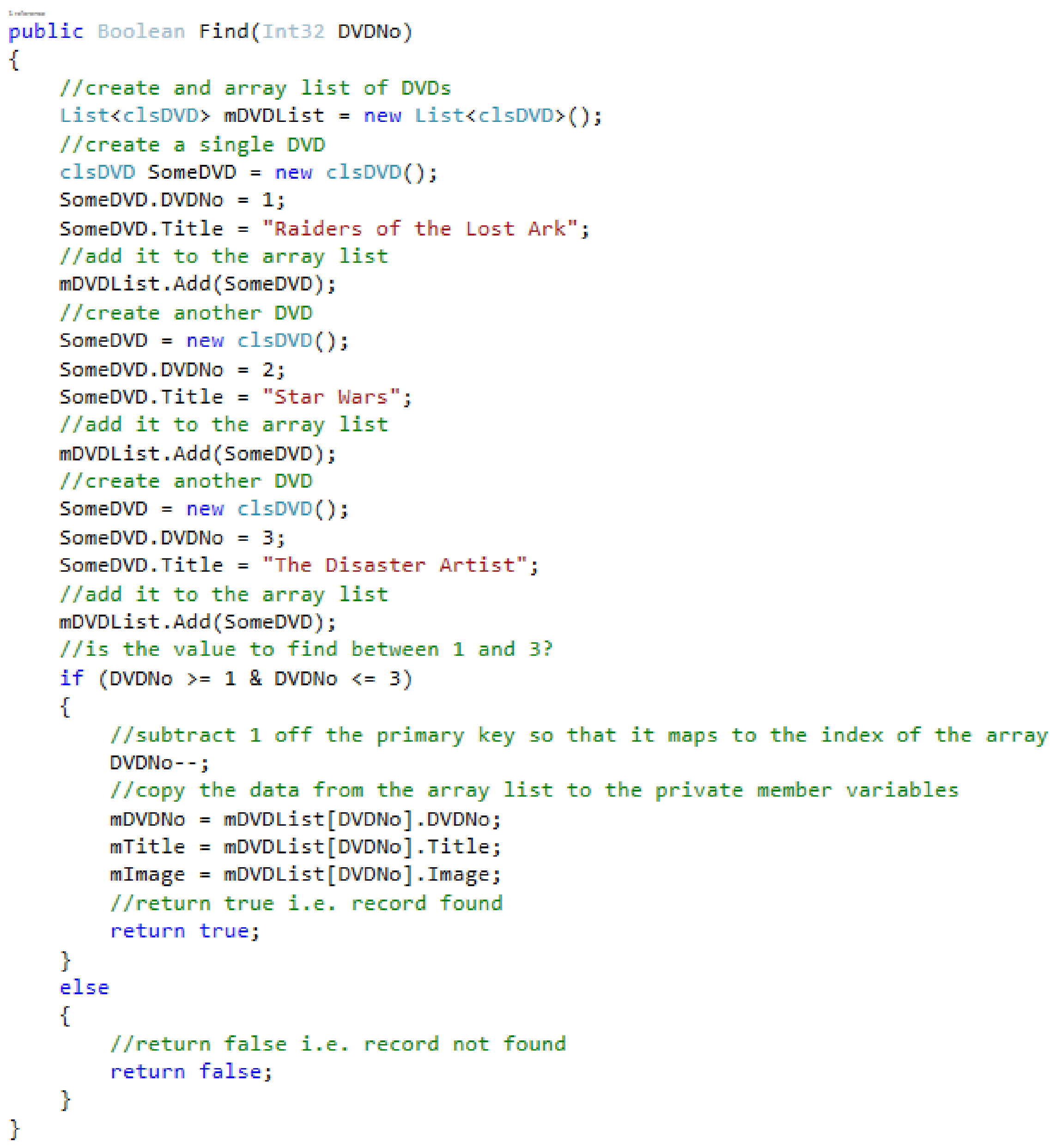
This would produce the following results…

- DVDNo would be assigned 2

- Title would be assigned “Star Wars”
- Image would be assigned “starwars.jpg”

But here is the thing; we want to achieve this functionality without having to write loads of code.

To the class clsDVD add the following public function that simulates the Find functionality…



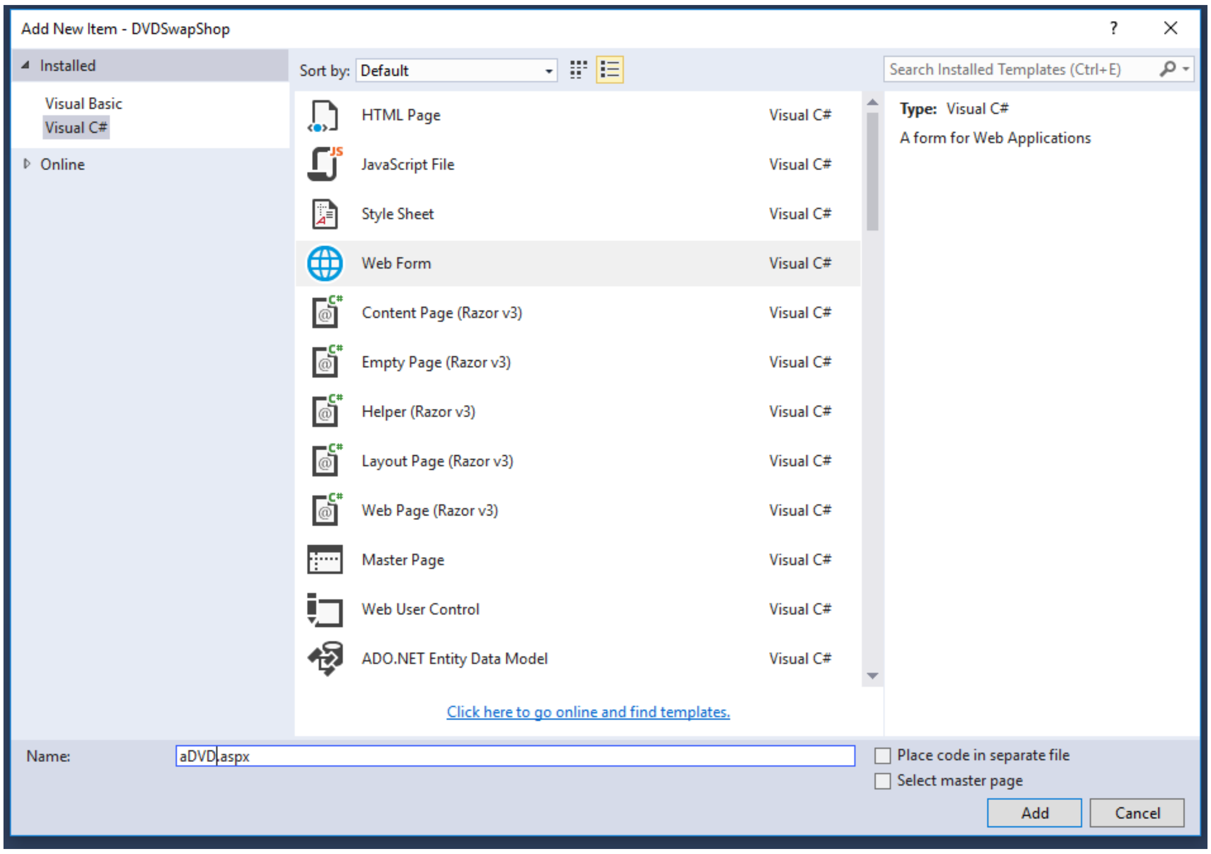
```
1 reference
public Boolean Find(Int32 DVDNo)
{
    //create and array list of DVDs
    List<clsDVD> mDVDList = new List<clsDVD>();
    //create a single DVD
    clsDVD SomeDVD = new clsDVD();
    SomeDVD.DVDNo = 1;
    SomeDVD.Title = "Raiders of the Lost Ark";
    //add it to the array list
    mDVDList.Add(SomeDVD);
    //create another DVD
    SomeDVD = new clsDVD();
    SomeDVD.DVDNo = 2;
    SomeDVD.Title = "Star Wars";
    //add it to the array list
    mDVDList.Add(SomeDVD);
    //create another DVD
    SomeDVD = new clsDVD();
    SomeDVD.DVDNo = 3;
    SomeDVD.Title = "The Disaster Artist";
    //add it to the array list
    mDVDList.Add(SomeDVD);
    //is the value to find between 1 and 3?
    if (DVDNo >= 1 & DVDNo <= 3)
    {
        //subtract 1 off the primary key so that it maps to the index of the array
        DVDNo--;
        //copy the data from the array list to the private member variables
        mDVDNo = mDVDList[DVDNo].DVDNo;
        mTitle = mDVDList[DVDNo].Title;
        mImage = mDVDList[DVDNo].Image;
        //return true i.e. record found
        return true;
    }
    else
    {
        //return false i.e. record not found
        return false;
    }
}
```

This will simulate the functionality we need without having to write all of the code.

Creating the Edit Page

Now that we have a simulated middle layer that allows us to search for dummy data we need to set about creating the presentation layer for the functionality.

Create a new ASPX page called aDVD.aspx with the following settings…


Add New Item - DVDSwapShop

- Installed
  - Visual Basic
  - Visual C#
- Online

Sort by: Default

Search Installed Templates (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#

Type: Visual C#
A form for Web Applications

Click here to go online and find templates.

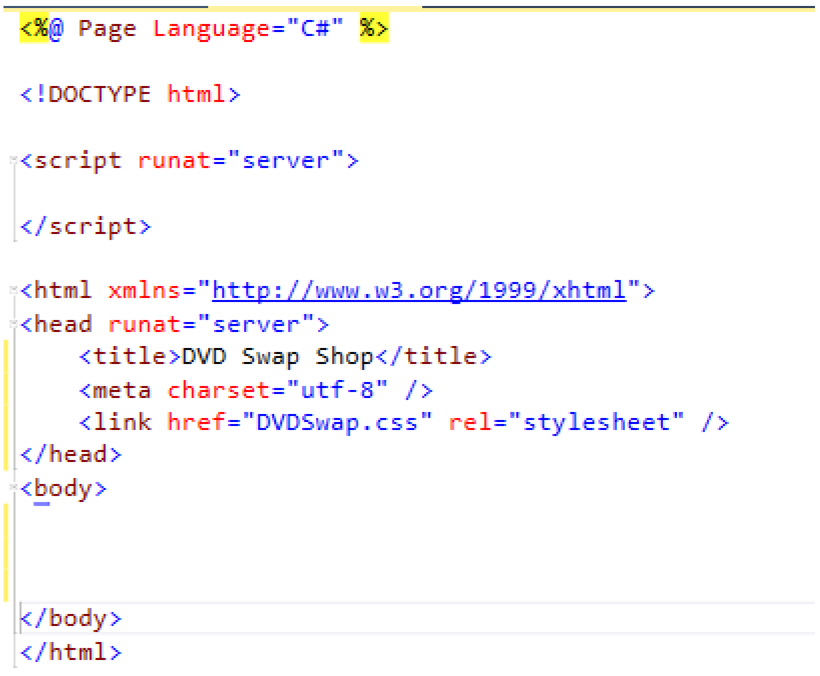
Name: aDVD.aspx

[ ] Place code in separate file
[ ] Select master page

Add Cancel

Make sure that the box “Place code in a separate file” isn’t ticked.

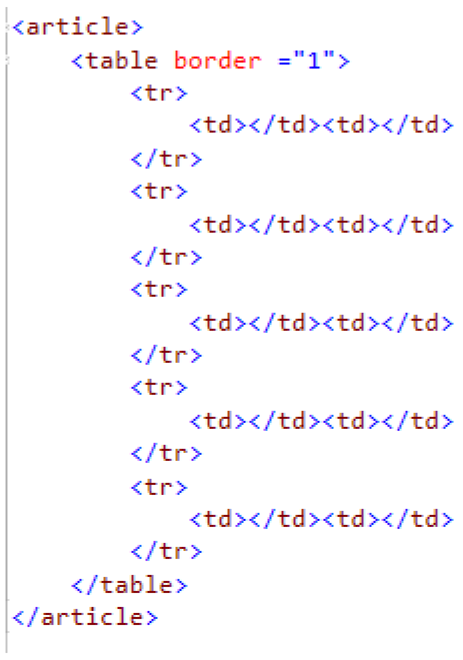
Modify the HTML like so to get rid of the default form and link in the style sheet…



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

<!DOCTYPE html>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>DVD Swap Shop</title>
    <meta charset="utf-8" />
    <link href="DVDSwap.css" rel="stylesheet" />
</head>
<body>



</body>
</html>
```

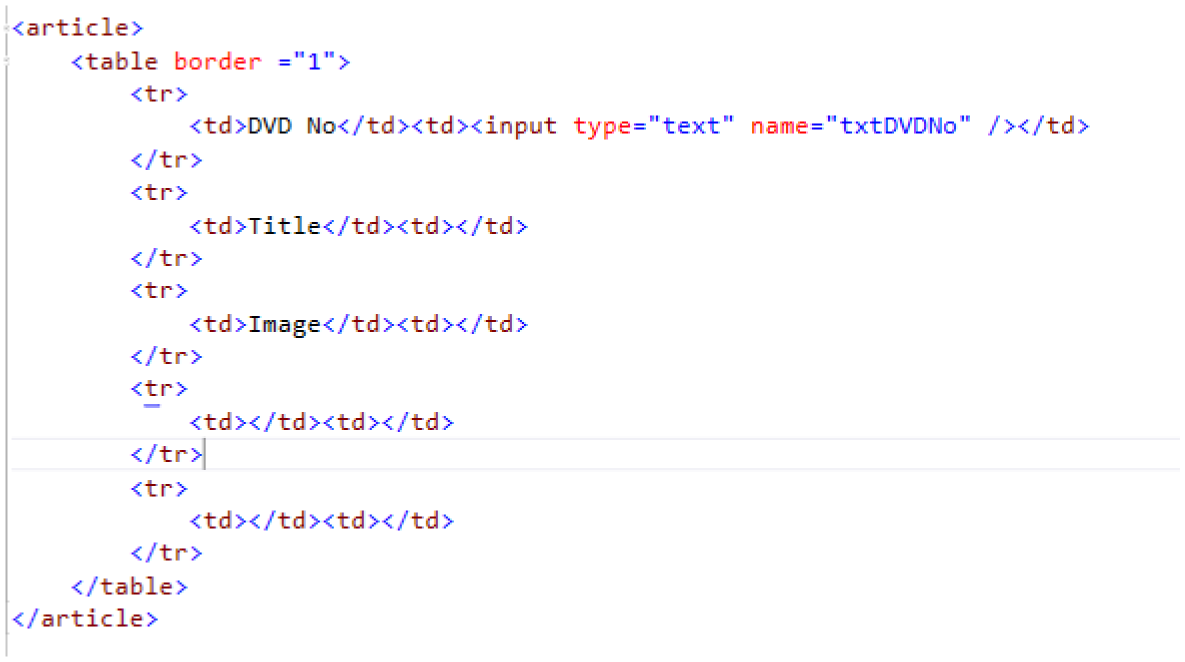
We will now create a page to allow us to display a specific DVD as a data entry form.

Create a new section of mark up for the <article> and within this create a table which is two columns wide and five rows high…



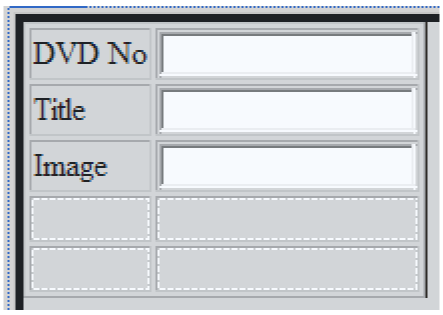
```html
<article>
    <table border ="1">
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

Within the table create suitable labels and text fields such that it produces a simple data entry form like so…



```html
<article>
    <table border ="1">
        <tr>
            <td>DVD No</td><td><input type="text" name="txtDVDNo" /></td>
        </tr>
        <tr>
            <td>Title</td><td></td>
        </tr>
        <tr>
            <td>Image</td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

The finished table should look something like this…



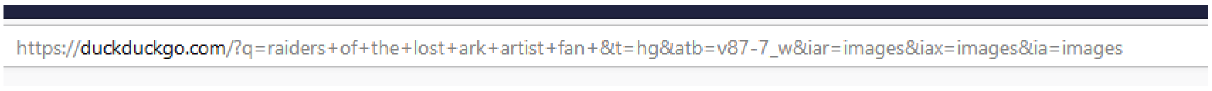
DVD No	
Title	
Image	

Using the Query String

To make this work we need to have a way of communicating to the form which record it needs to display, to do this we will make use of the query string.

The query string is an aspect of the URL that allows us to embed data into the URL.

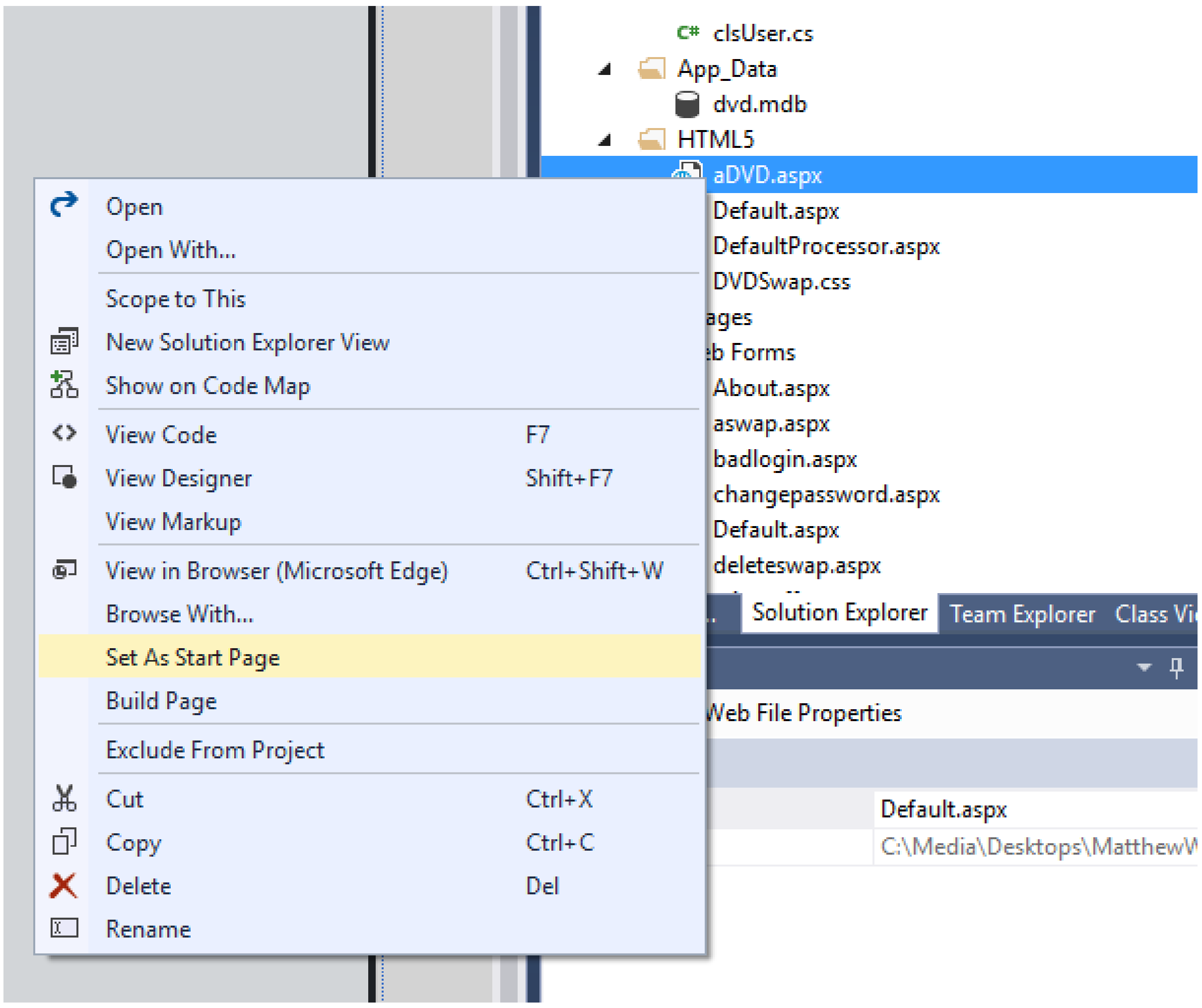
For example if you use a search engine you typically see the search data in the URL like so…



```
https://duckduckgo.com/?q=raiders+of+the+lost+ark+artist+fan+&t=hg&atb=v87-7_w&iar=images&iax=images&ia=images
```

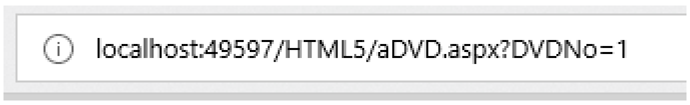
We can use the same approach to sending data to our web page

Right click on the web page and make it the default start up page.



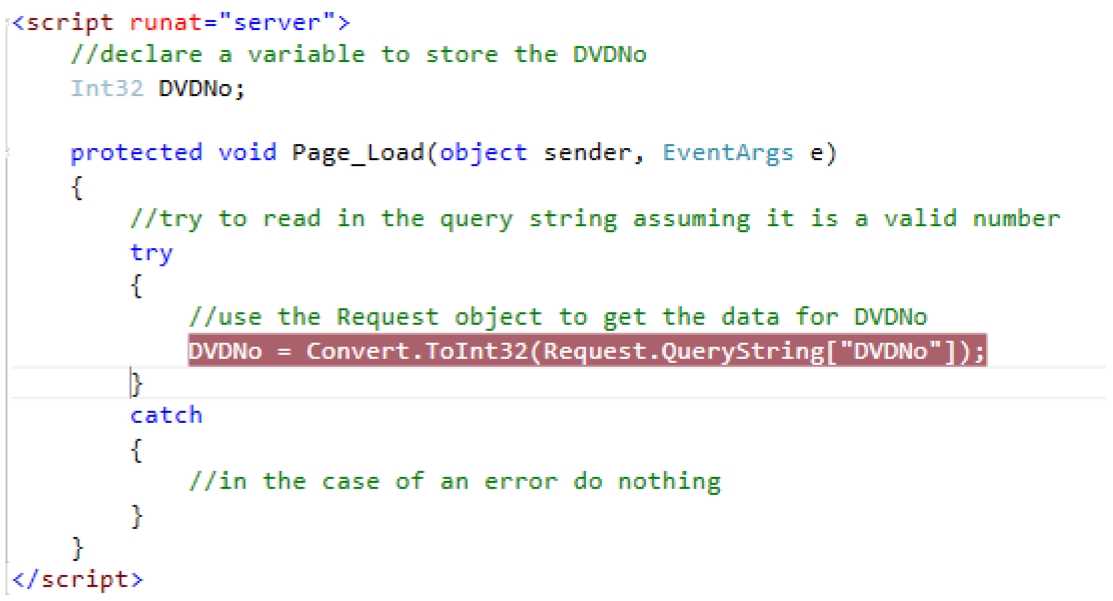
Now when you run the application this will be the first page, not Default.aspx.

Modify the URL like so to add the query string…



```
localhost:49597/HTML5/aDVD.aspx?DVDNo=1
```

Nothing will work yet as we need to add the code to capture the data. To do this we will need to modify the page such that it has some code to handle the load event…

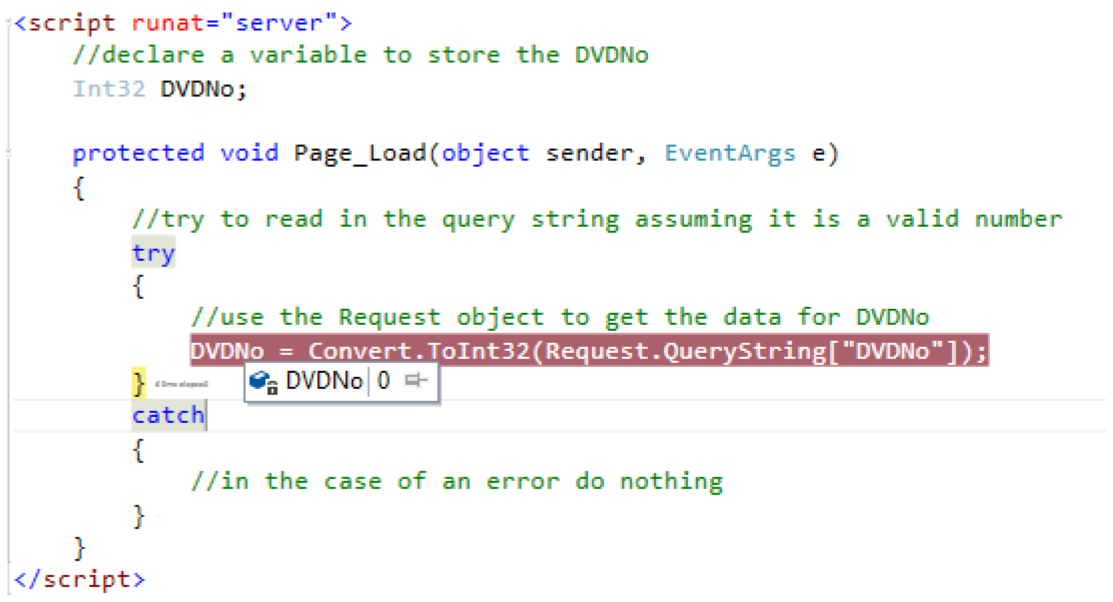


```
<script runat="server">
    //declare a variable to store the DVDNo
    Int32 DVDNo;

    protected void Page_Load(object sender, EventArgs e)
    {
        //try to read in the query string assuming it is a valid number
        try
        {
            //use the Request object to get the data for DVDNo
            DVDNo = Convert.ToInt32(Request.QueryString["DVDNo"]);
        }
        catch
        {
            //in the case of an error do nothing
        }
    }
</script>
```

Some things to notice:

1. The variable has been declared at the top of the script area outside of the function. This will make the data available to the entire page.
2. We have enclosed the code in red within a try and catch statement. This is to stop the program crashing if the query string isn't specified.
3. We have added a breakpoint so that we may see what is going on with the data.

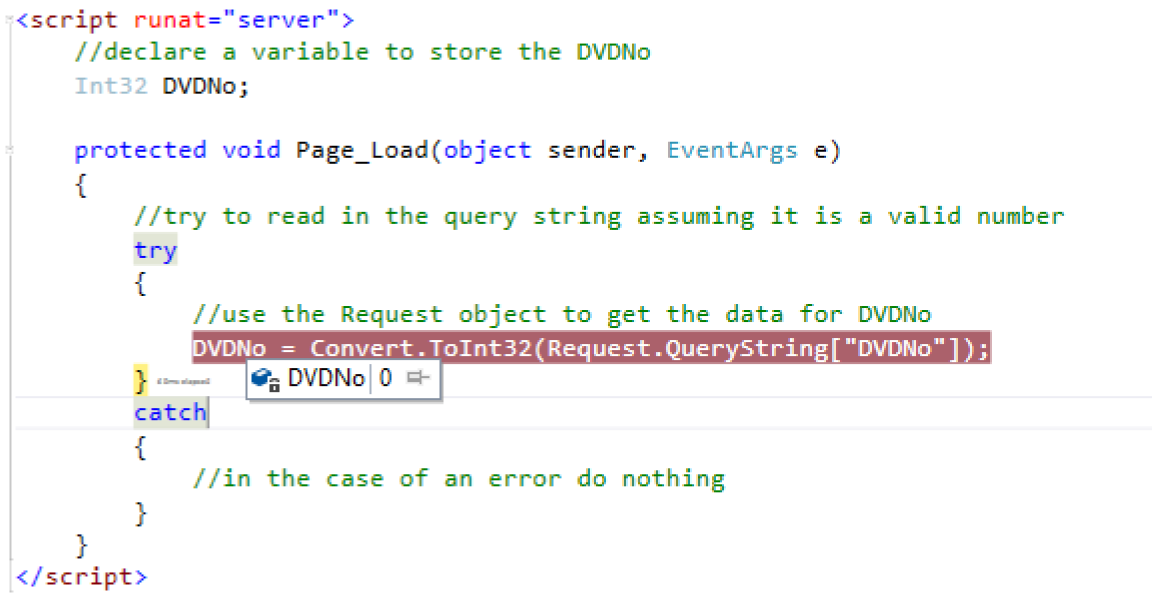
Run the page and the code should stop as soon as the load event is triggered, the variable DVDNo should have the value of zero…



```
<script runat="server">
    //declare a variable to store the DVDNo
    Int32 DVDNo;

    protected void Page_Load(object sender, EventArgs e)
    {
        //try to read in the query string assuming it is a valid number
        try
        {
            //use the Request object to get the data for DVDNo
            DVDNo = Convert.ToInt32(Request.QueryString["DVDNo"]);
        }       DVDNo | 0
        catch
        {
            //in the case of an error do nothing
        }
    }
</script>
```

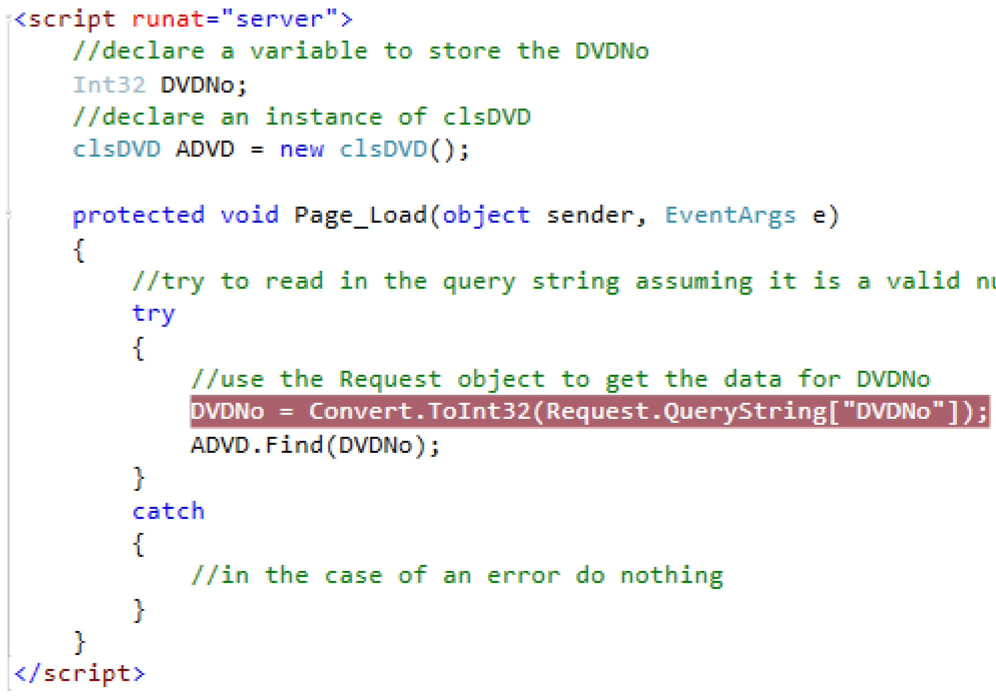
Press F5 to run the program as normal.

Now modify the query string like so and refresh the page. The breakpoint should kick in again as the page load event is triggered but this time the value captured should be 1…



```
<script runat="server">
    //declare a variable to store the DVDNo
    Int32 DVDNo;

    protected void Page_Load(object sender, EventArgs e)
    {
        //try to read in the query string assuming it is a valid number
        try
        {
            //use the Request object to get the data for DVDNo
            DVDNo = Convert.ToInt32(Request.QueryString["DVDNo"]);
        }          DVDNo | 0
        catch
        {
            //in the case of an error do nothing
        }
    }
</script>
```

Modify the code like so…

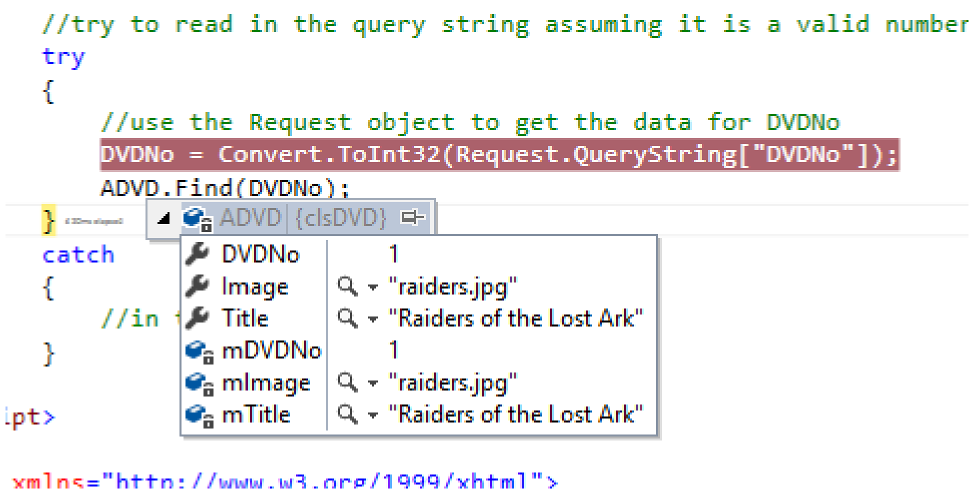


```
<script runat="server">
    //declare a variable to store the DVDNo
    Int32 DVDNo;
    //declare an instance of clsDVD
    clsDVD ADVD = new clsDVD();

    protected void Page_Load(object sender, EventArgs e)
    {
        //try to read in the query string assuming it is a valid number
        try
        {
            //use the Request object to get the data for DVDNo
            DVDNo = Convert.ToInt32(Request.QueryString["DVDNo"]);
            ADVD.Find(DVDNo);
        }
        catch
        {
            //in the case of an error do nothing
        }
    }
</script>
```

In this case we have created an object called ADVD instantiated from clsDVD. This is declared at the top of the script making its available to the whole page.

We then make use of the Find method in the object to get the data we are after.

If you use the debugger you will be able to explore the contents of the object once a valid DVD no has been entered in the query string.



```
    //try to read in the query string assuming it is a valid number
    try
    {
        //use the Request object to get the data for DVDNo
        DVDNo = Convert.ToInt32(Request.QueryString["DVDNo"]);
        ADVD.Find(DVDNo);
    }
    catch
    {
        //in
    }

ipt>

 xmlns="http://www.w3.org/1999/xhtml">
```

ADVD {clsDVD}	
DVDNo	1
Image	"raiders.jpg"
Title	"Raiders of the Lost Ark"
mDVDNo	1
mImage	"raiders.jpg"
mTitle	"Raiders of the Lost Ark"

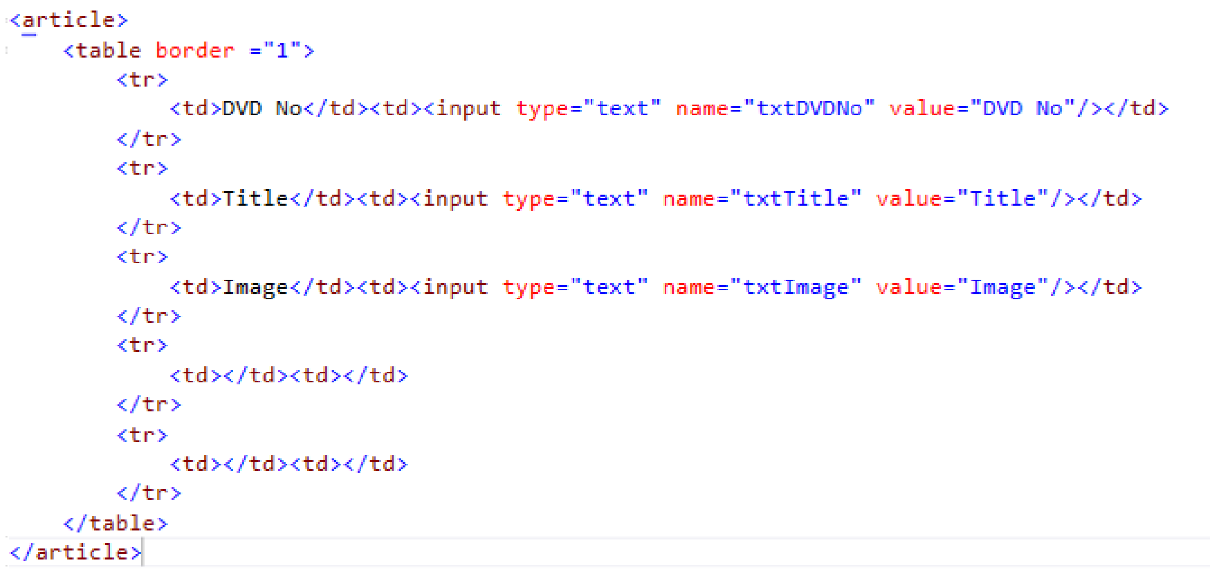
Try changing the data in the query string to make sure that you are able to access all of the records in the dummy data.

Modifying the Page to Display the DVD

To make this work as a dynamic page you will need to embed some code into the page's HTML.

Before we do this we need to add a value attribute to the text boxes so that they display some default data.

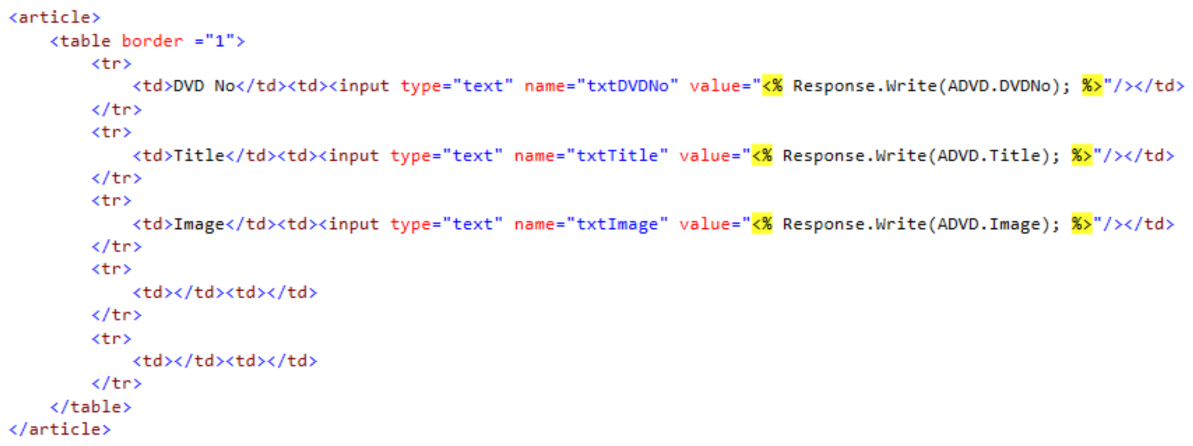
Modify the HTML like so…



```
<article>
    <table border ="1">
        <tr>
            <td>DVD No</td><td><input type="text" name="txtDVDNo" value="DVD No"/></td>
        </tr>
        <tr>
            <td>Title</td><td><input type="text" name="txtTitle" value="Title"/></td>
        </tr>
        <tr>
            <td>Image</td><td><input type="text" name="txtImage" value="Image"/></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

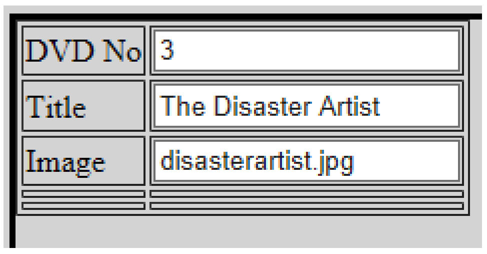
The value attribute allows us to specify default data within the HTML. This isn't quite what we are after though as this data needs to be taken from the middle layer rather than hard coded.

To make this work we need to write the data in the objects properties to the browser using Response.Write like so…



```
<article>
    <table border ="1">
        <tr>
            <td>DVD No</td><td><input type="text" name="txtDVDNo" value="<% Response.Write(ADVD.DVDNo); %>"/></td>
        </tr>
        <tr>
            <td>Title</td><td><input type="text" name="txtTitle" value="<% Response.Write(ADVD.Title); %>"/></td>
        </tr>
        <tr>
            <td>Image</td><td><input type="text" name="txtImage" value="<% Response.Write(ADVD.Image); %>"/></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

Which should give the following effect when you set up the query string correctly…

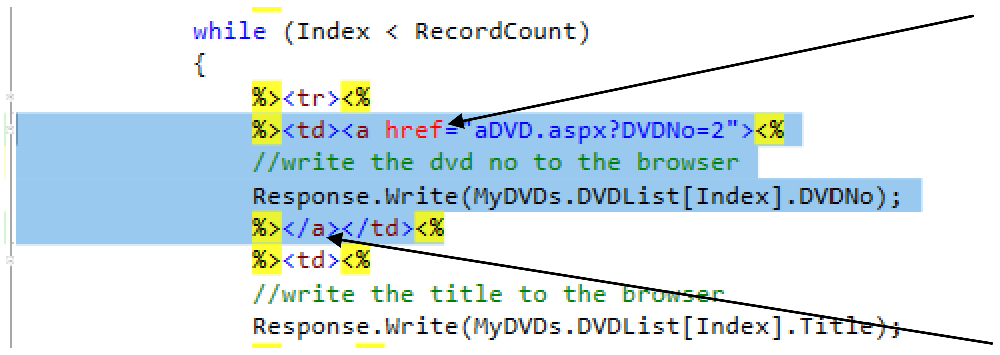


DVD No	3
Title	The Disaster Artist
Image	disasterartist.jpg

But isn’t manually writing data to the query string a bit boring? There has to be a better way.

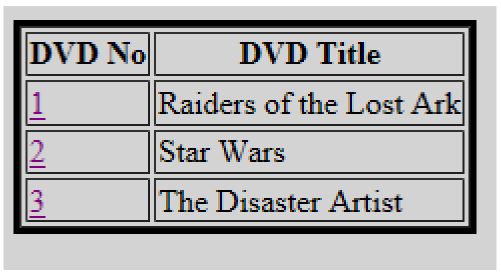
Creating Table Hyperlinks

On the main Default.aspx page modify the table to include some hyperlinks like so (don’t forget to close off the hyperlink correctly)…

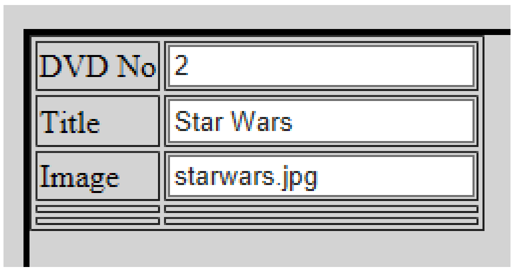


```
            while (Index < RecordCount)
            {
                %><tr><%
                %><td><a href="aDVD.aspx?DVDNo=2"><%
                //write the dvd no to the browser
                Response.Write(MyDVDs.DVDList[Index].DVDNo);
                %></a></td><%
                %><td><%
                //write the title to the browser
                Response.Write(MyDVDs.DVDList[Index].Title);
```

When you run the page you should have a clickable link that currently always takes you to record 2…



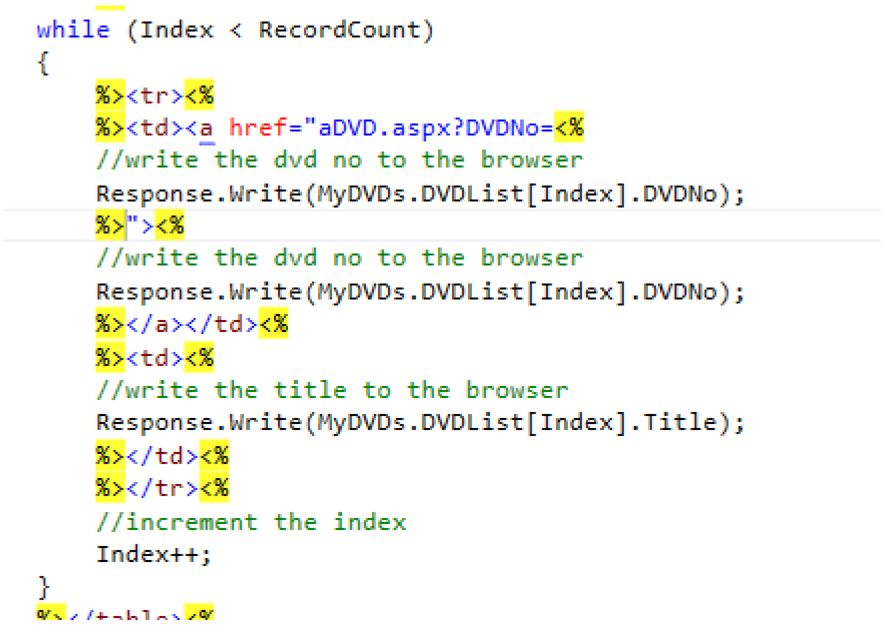
DVD No	DVD Title
1	Raiders of the Lost Ark
2	Star Wars
3	The Disaster Artist



DVD No	2
Title	Star Wars
Image	starwars.jpg

This isn’t quite right; to fix it we need to dynamically code the DVD No into the HTML.

Like so…



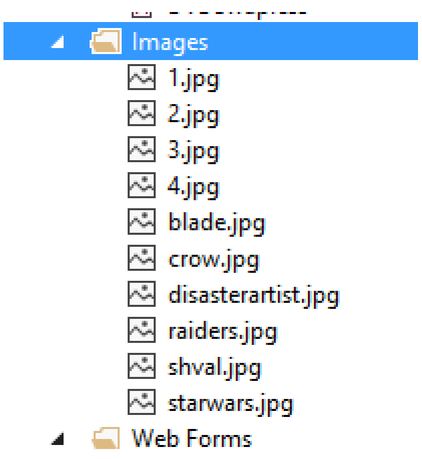
```
while (Index < RecordCount)
{
    %><tr><%
    %><td><a href="aDVD.aspx?DVDNo=<%
    //write the dvd no to the browser
    Response.Write(MyDVDs.DVDList[Index].DVDNo);
    %>"><%
    //write the dvd no to the browser
    Response.Write(MyDVDs.DVDList[Index].DVDNo);
    %></a></td><%
    %><td><%
    //write the title to the browser
    Response.Write(MyDVDs.DVDList[Index].Title);
    %></td><%
    %></tr><%
    //increment the index
    Index++;
}
%></table><%
```

Displaying the Image

The last task is to set up the ADVD page such that it displays the actual image rather than the name of the file.

We can use many of the tricks above to get this effect.

A set of suitable images for this task are available in the Images folder of the Swap Shop…

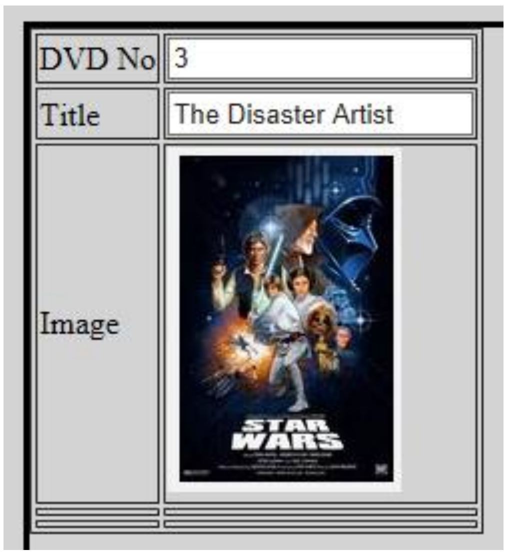


```
◢ Images
    1.jpg
    2.jpg
    3.jpg
    4.jpg
    blade.jpg
    crow.jpg
    disasterartist.jpg
    raiders.jpg
    shval.jpg
    starwars.jpg
◢ Web Forms
```

As is often the case when writing dynamic content it is easier to hard code the HTML first, say like so…

```
<article>
    <table border ="1">
        <tr>
            <td>DVD No</td><td><input type="text" name="txtDVDNo" value="<% Response.Write(ADVD.DVDNo); %>"/></td>
        </tr>
        <tr>
            <td>Title</td><td><input type="text" name="txtTitle" value="<% Response.Write(ADVD.Title); %>"/></td>
        </tr>
        <tr>
            <td>Image</td><td><img src="../Images/starwars.jpg" /></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

Test it to see if it gives the effect you are after…

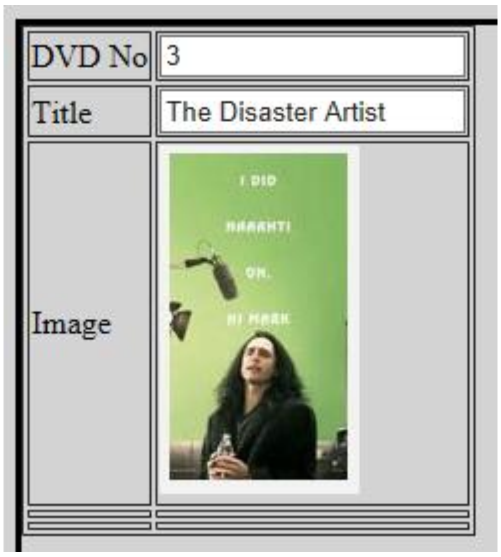


DVD No	3
Title	The Disaster Artist
Image	

Then add the embedded code to create the dynamic content, like so…

```
<article>
    <table border ="1">
        <tr>
            <td>DVD No</td><td><input type="text" name="txtDVDNo" value="<% Response.Write(ADVD.DVDNo); %>"/></td>
        </tr>
        <tr>
            <td>Title</td><td><input type="text" name="txtTitle" value="<% Response.Write(ADVD.Title); %>"/></td>
        </tr>
        <tr>
            <td>Image</td><td><img src="../Images/<% Response.Write(ADVD.Image); %>" /></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
        <tr>
            <td></td><td></td>
        </tr>
    </table>
</article>
```

Giving us…



DVD No	3
Title	The Disaster Artist
Image	

Things to Try Yourself…

Based on what you now know about forms, create an HTML form within this second page we created today. This should send the form data to a suitable form processor. Don't worry about actually writing the data to the data layer, simply use response write to display what would be saved to the data layer.

The second task is to style the new page such that it has a consistent layout that matches the Default.aspx page. Have a go at specifically re-defining the table formatting such that all tables have a distinctive look and feel. (Hint, this is very similar to how we styled the <body> tag for the HTML page earlier in the module.