**CPSC1517 Exercise 01**

**Purpose**

To practice how to use Razor to create an ASP.NET Layout Page and how to create ASP.NET Content Pages that uses a layout page.

**Description**

The web pages you will be creating in exercises 1-4 will consist of a layout, a navigation menu to switch between the different pages, and a basic form.

In this exercise you will create:

* an ASP.NET Layout Page with a common layout for all pages and a common navigation menu for all pages.
* a basic 2 column form which will contain a textbox and a checkbox list.
* The necessary code to process the form on a post back.

**Resources**

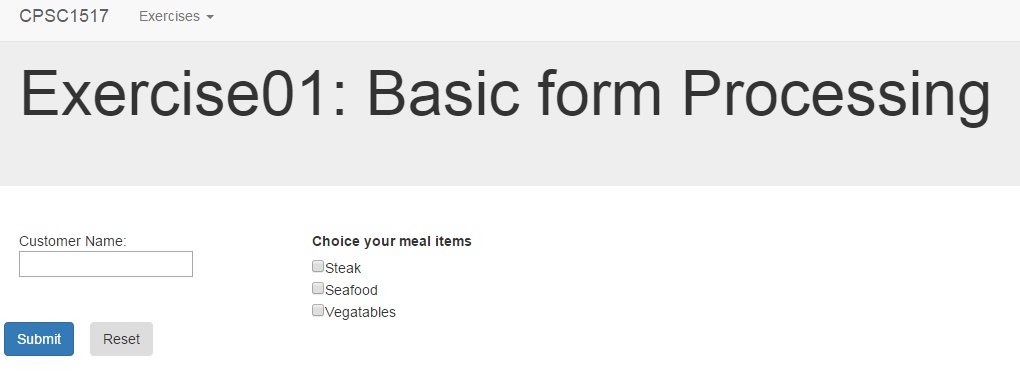
[Creating a Consistent Layout working Bootstrap](http://getbootstrap.com/css/#overview)

**Setup**

1. Fork the Razor-1 repo from my github (RobbinLawCPSC1517/), to your github account location after you logon to your account.
2. Clone your version of the Razor-1 repo to your local machine.
3. Get all the packages via NuGet with the Restore and/or Update features, but do not update jQuery or Bootstrap.

**Create and Process a Basic Form**

You will create a simple form and process the post back using C# via Razor. You will use some formatting features of Bootstrap on the form. The form (shown below) will contain a text box and a checkbox selection list in a 2-column form using bootstrap for formatting. When you press Submit, your form will display the entered and selected data/choices.



Create the form

1. Create a form in Exercise01.cshtml with an id of **EX01,** method of post.
2. Within this <form> tag create 2 more <div class-“row”> tags.
3. Within the first **<div class-“row”>** create 2 more **<div>**. The first **<div>** will span 3 medium bootstrap columns and the second 9 bootstrap columns. ([Creating a Consistent Layout working Bootstrap](http://getbootstrap.com/css/#overview))
4. In the first column add a text for your label (do not use the **<Label>** tag): Customer Name and a **<input>**. Add a value=’’@.....” to the **<input>** so that input will be retained on posts. You will need to add a variable to your form code processing to handle the value parameter in the **<input>** tag.
5. In the second column use a **<Label>** tag to add the title Choose your meal items. Add **at least 3** meal items as a set of checkboxes.
6. Add two **<input>** tags of **type**: submit; **class**: btn; and **value** of submit and reset respectively. Make the submit button the primary button. **DO NOT** use type: reset.
7. Save and view your page.

Code the form processing logic.

Code the necessary logic to process this form. Your logic will need to:

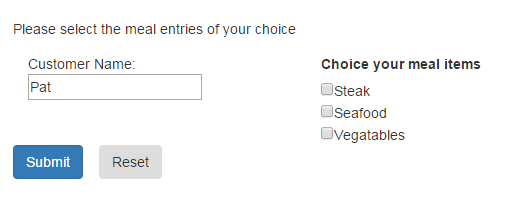
1. Retrieve the values from your various form tags.
2. If the reset button was pressed, set the value for your customer name to nothing.
3. If the submit button was pressed:
   1. Check that a customer name was entered. If not display a message.
   2. Check that a meal selection was made, if not display a message.
   3. If A and B are valid, display a message containing the customer name and a bulleted containing the selected meal items.

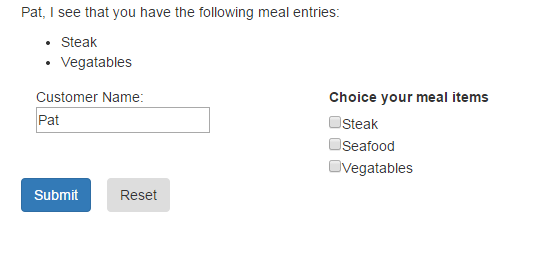
**Submission.**

Once you have completed the exercise, you **must** demonstrate your work to you instructor **in class.** Your instructor **may**instruct you to submit the completed work to a secondary location such as your class moodle site, or github repository. **This exercise is either evaluated as complete or incomplete**.

**Sample processing images.**

You may customize the processing messages to suit your own content.





**CPSC1517 Exercise 02**

**Purpose**

To practice how to use Razor to Content Pages that will display data queried from a database. You will practice basic input argument validation and page to page data transfer.

**Description**

The web pages you will create in exercises 1-4 will have a consist layout, a navigation menu to switch between the different pages and a basic form. You will add this exercise to your existing exercise project.

In this exercise you will:

* maintain your navigation menu
* create web pages that will use various filter searches; query a database with the argument and display the results
* pass data from one page to another
* use techniques to display multiple record collections

**Resources**

Moodle course notes and class examples.

[Creating a Consistent Layout working Bootstrap](http://getbootstrap.com/css/#overview)

**Setup**

1. Download and restore the Chinook database (.bak).
2. Create three new content web pages called Exercise02\_A, Exercise02\_B and Exercise02\_C in your ExercisePages folder.
3. Update your Layout Page to have a menu item referencing the new files.
4. Add a <connectionString> entry to your web.config file using configSource.
5. Create a config file (referenced by step 4) and add a connection string to point to the Chinook database.

**Exercise Page Processing**

You will create three content web pages and process the post back using C# via Razor. You will use some formatting features of Bootstrap on the form.

## Exercise02\_A content page

This page will use a single form. The page will prompt the user to enter a search argument: artist name (or partial name). A page Fetch Artist button must validate a name was entered. If no name is entered, issue an appropriate message to the user. If a name was entered, query the database for artists that match the search argument. Depending on the results of the query; issue an appropriate message if no data was found or forward the artist name (or partial name) to page B.

Use the following to create the partial string parameter

var @partialstring = “%” + yourtextboxvalue + “%”;

Use the following SQL query and the partial string parameter

SELECT ArtistId  
FROM Artists  
WHERE Name like @0

<table>

<thead>

<tr>

<th>AlbumID</th>

<th>Title</th>

<th>ReleaseYear</th>

<th>ReleaseLabel</th>

</tr>

</thead>

<tbody>

@foreach (var row in results)

{

<tr>

<td>@row.AlbumID</td>

<td>@row.Title</td>

<td>@row.ReleaseYear</td>

<td>@row.ReleaseLabel</td>

</tr>

}

</tbody>

</table>

## Exercise02\_B content page

This page will use a single form. The page will receive a search argument: artist name (or partial name) from the previous page. The page must validate a name was received. If no name is received, return to Exercise02\_A. If a name was received, query the database for artists that match the search argument. Issue an appropriate message if no data was found. Build a list of artists in the <select> tag (dropdownlist) with the artistid as the <option> tag value and the artist name as the displayed text. Include a prompt line in your <select> tag. Include a Fetch Album button.

When the Fetch Album button is pressed verify that there are artist albums from which to select. If there are albums also verify that something other than the prompt line has been selected. Obtain the value selected in the dropdownlist. Pass the option value (artistid) to Exercise02\_C content page.

Use the following to create the partial string parameter

var @partialstring = “%” + yourreceivedvalue + “%”;

Use the following SQL query and the partial string parameter

## SELECT ArtistId, Name FROM Artists WHERE Name like @0 ORDER BY Name

## Exercise02\_C content page

## This page should receive 1 piece of data from a previous web page (Exercise02\_B): artistid. If no data was received, return the user to Exercise02\_A. If data was received, query the database for the artist that matches the artistid. Display the name of the artist as a header. Next, query the database to obtain all the albums associated with the artist. Depending on the results of the query; issue an appropriate message if no data was found or display the returned data in a WebGrid. There is no form on this page.

Use the following SQL queries

SELECT Name   
FROM Artists  
WHERE ArtistId = @0 (Where @0 is your artistid value)

SELECT AlbumId, Title, ReleaseYear, ReleaseLabel   
FROM Albums  
WHERE ArtistId = @0 (Where @0 is your artistid value)  
ORDER BY Title

**Display formatting.**

## This would be the resulting display on the webgrid page.

## 

Display formatting requirements may be adjusted by your instructor.

**Submission.**

Once you have completed the exercise, you **must** demonstrate your work to you instructor **in class.** Your instructor **may**instruct you to submit the completed work to a secondary location such as your class moodle site, or github repository. **This exercise is either evaluated as complete or incomplete**.

**CPSC1517 Exercise 03**

**Purpose**

To practice how to use Razor to Content Pages that will insert data into a database. You will practice basic input argument validation.

**Description**

The web pages you will created in exercises 1-4 will have a consist layout, a navigation menu to switch between the different pages and a basic form. You will add this exercise to your existing exercise project.

In this exercise you will:

* maintain your navigation menu
* create a web page that will input a new record for a database table
* do validation against inputted data to ensure a valid data and output appropriate messages

**Resources**

Moodle course notes and class examples.

[Creating a Consistent Layout working Bootstrap](http://getbootstrap.com/css/#overview)

**Setup**

1. Download and restore the Chinook database (.bak).
2. Create a new content web page called Exercise03 in your ExercisePages folder.
3. Update your Layout Page to have a menu item referencing Exercise03.

.

**Create and Process a Basic Form**

You will create a content web page and process the post back using C# via Razor. You will use some formatting features of Bootstrap on the form.

## Exercise03 content page

This page will prompt the user to enter input values to add a new album record to the Chinook database. A sample form has been supplied showing a possible field layout. The fields are to be validated according to the supplied Validation table. The foreign key record field will use the <select> control to allow the user to select from a list. The <select> list must start with a prompt entry. Data is to be maintained in the input fields and <select> list during validation. Clear and reset these fields when the user cancels.

Use the following SQL commands:

SELECT ArtistId, Name,   
FROM Artists  
ORDER BY Name

INSERT INTO Albums (Title, ArtistId, ReleaseYear, ReleaseLabel)  
VALUES(@0, @1, @2, @3)

**Validation Requirements.**

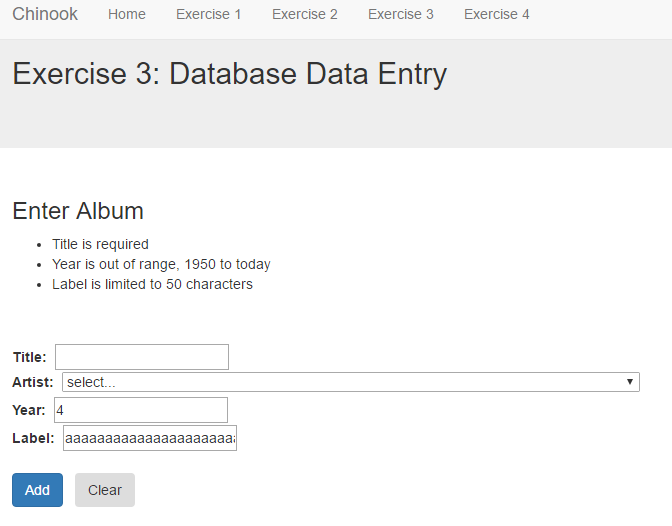
|  |  |  |
| --- | --- | --- |
| **Database Field** | **Datatype** | **Validation Requirements** |
| Title | String | Required, max length 160 |
| ArtistId | Int (Foreign Key) | Required |
| ReleaseYear | Int | Required, between 1950 and this year inclusive (DateTime.Today.Year) |
| ReleaseLabel | String | Max Length 50 |

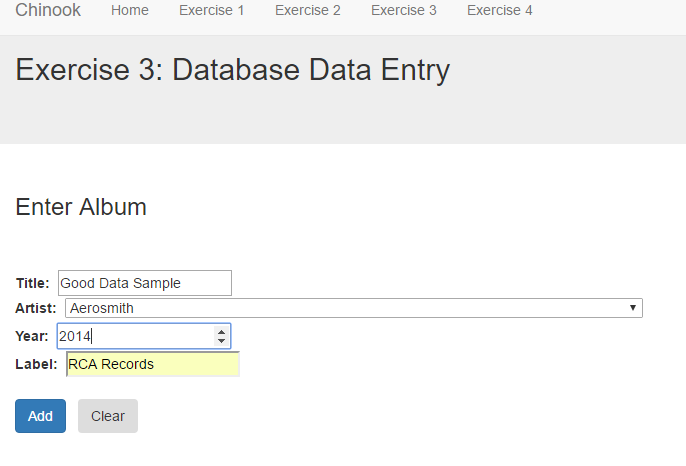
**Display formatting.**

Display formatting requirements will be outlined by your instructor.

**Submission.**

Once you have completed the exercise, you **must** demonstrate your work to you instructor **in class.** Your instructor **may**instruct you to submit the completed work to a secondary location such as your class moodle site, or github repository. **This exercise is either evaluated as complete or incomplete**.





**CPSC1517 Exercise 04**

**Purpose**

To practice how to use Razor on Content Pages that will update or delete data in a database. You will practice basic input argument validation.

**Description**

The web pages you will create in exercises 1-4 will have a consist layout, a navigation menu to switch between the different pages and a basic form. You will add this exercise to your existing exercise project.

In this exercise you will:

* maintain your navigation menu
* create the necessary content web page(s) that will update/delete a record in a database table
* do validation against inputted data to ensure valid data and output appropriate messages

**Resources**

Moodle course notes and class examples.

[Creating a Consistent Layout working Bootstrap](http://getbootstrap.com/css/#overview)

**Setup**

1. Download and restore the Chinook database (.bak).
2. Create new content web page(s) called Exercise04\_x in your ExercisePages folder where x is A, B etc. The number of pages you create will depend on how you tackle the exercise.
3. Update your Layout Page appropriately.

.

**Create and Process a Basic Form**

You will create the necessary content web page(s) and process the post back using C# via Razor. You may use formatting features of Bootstrap on the form. Navigation (except for menu entries) is up to you.

## Exercise04\_x content page

You are to add the necessary web page(s) to create a solution for updating and deleting an Albums record in the database.

The approach to this solution is up to you, whether you use a single page or multiple pages.

You may use exercise pages that you have already created though they may need to be altered.

You are required to have a filtered search lookup to find the particular album record to change.

A dropdownlist of albums is not acceptable as there are hundreds of albums in existence.

You must get confirmation from the user before deleting an album.

Your solution must use user friendly error handling.

Your solution must validate updated data.

Appropriate messages to the user are expected.

Use the following SQL commands:

SELECT AlbumId, Title, ArtistId, ReleaseYear, ReleaseLabel   
FROM Albums  
WHERE AlbumId = @0

UPDATE Albums   
SET Title =@0,   
 ArtistId = @1,   
 ReleaseYear = @2,  
 ReleaseLabel = @3  
WHERE AlbumId = @4

DELETE Albums   
WHERE AlbumId = @0

Exercise 3 has the SELECT commands to obtain the AstistId for use in a <select> control.

Exercise 2 has query SELECT commands to obtain a list of Albums. Exercise 2 can be used to find a desired album to update/delete. Modifications to Exercise 2 would be needed. If you do modify Exercise 2 to be used in solving Exercise 4, your solution for Exercise 2 **still must** work correctly.

**Validation Requirements.**

|  |  |  |
| --- | --- | --- |
| **Database Field** | **Datatype** | **Validation Requirements** |
| Title | String | Required, max length 160 characters |
| ArtistId | Int (Foreign Key) | Required |
| ReleaseYear | Int | Required. Year must be between 1950 and this year  (DateTime.Today.Year) |
| ReleaseLabel | String | Max length 50 characters |

**Display formatting.**

Display formatting is up to you.

**Submission.**

Once you have completed the exercise, you **must** demonstrate your work to you instructor **in class.** Your instructor **may**instruct you to submit the completed work to a secondary location such as your class moodle site, or github repository. **This exercise is either evaluated as complete or incomplete**.

Request.Form["ProductSearch"];

// use name value

//The Request.Form command is used to collect values in a form with method="post".

//Request.Forfm( element )[(index)|.Count]

//The value of Request.Form(element) is an array of all the values of element that occur in the request body

Response.Redirect("ProductWebGridDisplay.schtml?pid=" + productsearch);

//Response.Redirect($"ProductWebGridDisplay.schtml?pid={productsearch}");

//Response.Redirect(string) or Response.Redirect(string, Boolean)

//Redirects a request to a new URL and specifies the new URL.

//Redirect(String, Boolean):Redirects a client to a new URL. Specifies the new URL and whether execution of the current page should terminate.

<label id="forProductSearch">Product Name</label><br /><!--It also works w/o label-->

<p><strong><span style="color:firebrick">Enter a product name (or partial name).</span></strong></p>

//span seems like div, class="", style="color: "

var queryStatement = "SELECT ProductID, ProductName, CategoryID, "

+ "UnitPrice, UnitsInStock, Discontinued "

+ "FROM Products "

+ "WHERE ProductName like @0ORDER BY ProductName"; //lack of a blank space

/ vs \

Layout = "~/\_Layout.cshtml";