# Lesson 3 – Responsive Forms & Validation

# Setup

1. In the **Solution Explorer**, right-click on the **Project 'Lesson03'** > **Set as Startup Project**.
2. Right-click **Lesson03** and select **Build**



1. Run the project.

Graphical user interface, text, application

Description automatically generated

1. From the **Models** folder inthe lesson materials, **copy all files** into the **Models** folder in VS2022 Lesson03. **Do not drag** **in the whole folder**, or you will create L03/Models/Models, which will not work.
2. From the **Views** folder in the lesson materials, **copy all the sub-folders and contents** into the **Views** folder in VS2022 Lesson 03. Do not drag in the whole folder.
3. From the **Controllers** folder in the lesson materials, copy **all the files** into the **Controllers** folder in VS2022 Lesson 03. Do not drag in the whole folder.
4. Drag the **images folder** from the lesson materials **onto the** **wwwroot** folder in VS2022 Lesson 03.
5. Drag the **utils folder** from the lesson materials **onto** **Lesson03** in VS2022 Lesson 03.
6. Drag the **libman.json file** from the lesson materials **onto** **Lesson03** in VS2022.. Right-click **libman.json > Restore Client-Side Libraries**. This step creates **wwwroot/lib** folder structure containing the necessary bootstrap libraries.
7. Modify the **Program.cs** file by adding the following line to the very top of the file, then save the file.

global using RP.SOI.DotNet.Utils;

1. Right-click the project **Lesson03** and select **Build**. Ensure there are no errors.



1. Run the project with debugging.

Graphical user interface, text, application

Description automatically generated

CONTINUE IF YOUR PROJECT IS ERROR-FREE.

# Extension Methods

1. Run your project, add the controller (**demo**) and action (**StringExtMethods**) to your browser's URL.

Graphical user interface, text, application

Description automatically generated

1. Type in a few words of text and click the **[UpperLower]** button and observe the output.

Graphical user interface, text

Description automatically generated

1. The **[Stretch]** button is not working because the **Stretch** extension method is undefined. **Stop** your project. **You cannot edit code on a running project**. Perform the following:
2. Show the **Task List** window (**CTRL \ T**), and filter by project Lesson03. Click the Description column to order the tasks.

Graphical user interface, application

Description automatically generated

1. Select the double-click TODO Task 1. Referring to the slides on Extension Methods in the student PDF, add the missing code to the Stretch extension method.
2. Run your project again and ensure the [Stretch] button works as intended.

# Views and Models

1. Run your project, add the controller (**demo**) and action (**ShowTriangle**) to your Browser's URL to see the output. Refresh the browser. Does the information change with each refresh?

Graphical user interface, text, application

Description automatically generated

1. Open **DemoController.cs** and examine the method **ShowTriangle()**. Answer the following questions.

|  |  |  |
| --- | --- | --- |
| a. | What makes the sides have different values each time? | Random rnd = new(); |
| b. | What is the range of values for each side? | 1 to 4 |
| c. | What is the variable that holds the model? | shape |
| d. | What is the class of the model? | Triangle.cs |
| e. | What is the name of the View that the action is calling? | TriView.cshtml |

1. Open **Views/Demo/TriView.cshtml** and examine code. Answer the following questions

|  |  |  |
| --- | --- | --- |
| a. | Write down the line that specifies the class of the model. | @model Lesson03.Models.Triangle |
| b. | What is the difference between @Model and @model? | @model is the model directive which is the class of the model  @Model is used to access a particular object |
| c. | What are the three possible outcomes of a triangle? | Normal, equilateral, or isosceles. |
| d. | How does the view determine the outcome? | If 3 sides generated are all equal, it is an equilateral, if any of the 2 sides have the same number generated, it will be an isosceles triangle. Else it will just be a normal triangle |
| e. | How does the view calculate the perimeter of the triangle? | @(Model.Side1 + Model.Side2 + Model.Side3)  Razor code |

# Add Bootstrap to create a responsive form

1. Run your project, add the controller (parent) and action (volunteer) to your Browser's URL to see the output. The form will be unresponsive but otherwise functional.
2. Lesson03 contains a completed Organic Fruits applications and serves as a reference for both responsive and non-responsive forms:

* Non bootstrap version: Controller: **organic**, Action: **subscription**.
* Bootstrap version (Labels Left): Controller: **organic**, Action: **subscriptionA**.
* Bootstrap version (Floating Labels): Controller: **organic**, Action: **subscriptionB**.

Referencing the **Organic Fruits** project, modify the non-responsive **VolunteerBS.cshtml** view to make it **responsive**, with **floating labels**. To view the bootstrap form, add an **id** of "**BS**" to the url as shown below. The "BS" is the id, which is concatonated with the word Volunteer in the controller so that the correct view is called.

Graphical user interface, application

Description automatically generated

* 1. **Remove** the **CSS styles** and the **breaks (<br />)**. Add bootstrap styles to the form. Refer to **Organic/SubscriptionB** and view as required.
  2. Column size classes:
* Title Radio Group: **col-4**
* Name Text Box: **col-6**
* Postal Code: **col-3**
* Mobile: **col-3**
* Activity: **col-5**
* Days: **col-4**

When complete, your form will be responsive and resemble the screenshot below.

Graphical user interface, application

Description automatically generated

# Validating inputs

1. The **Volunteer.cshtml** does not provide any validation. To perform validation, we need to add code to the **Parent** controller, **Submit** action. The validation messages are as follows:

* Please enter all fields.
* Mobile phone must be exactly eight digits.
* Check at least one day.

You may add other validation you deem suitable. ***Hint***: changing type="text" to type="number" for both the postal code and the mobile will ensure the user can only enter numbers.

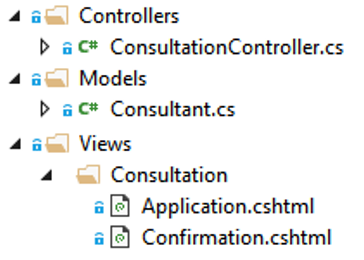
1. Open Task List window and double click Task 2. Complete Task 2 to validate user enters or selects all fields.
2. Complete Task 3 to provide validation on the mobile phone.
3. Complete Task 4 to ensure the selection of at least one day.
4. Run your project again, add the controller and action to your Browser's URL to see the output. Check the error messages display correctly. When all validations pass, you will see the following page.

A picture containing text

Description automatically generated

# Solving the problem

1. The project structure of your solution should be as follows:



# Create the Model

1. Right-click on the folder **Models**, **Add**, **Class…** Enter **Consultant.cs** and click [Add]. Add in the following code:

|  |
| --- |
| namespace Lesson03.Models;  public class Consultant  {  public string StaffName { get; set; } = null!;  public string Email { get; set; } = null!;  public string AcadPos { get; set; } = null!;  public DateTime StartDate { get; set; }  public DateTime EndDate { get; set; }  } |

**Note:** C# introduced null safety. For this reason, declare StaffName, Email and AcadPos with:

**property\_name = null!;**

This is known as the **null forgiving operator**. There will be more on null safety in subsequent weeks. But, for the moment, know that the null forgiving operator tells the compiler that something that could be null is safe to access. More detail in the stack overflow link below.

[https://StackOverflow/C#/damit](https://stackoverflow.com/questions/54724304/what-does-null-statement-mean)

# Create the Application view

1. Views/Consultation/ApplicationBS.cshtml is currently non-responsive. First, addd a reference to the bootstrap CSS library.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <link href="~/lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />  <title>ApplicationBS</title>  </head>  <body> |

1. **Remove** the **CSS styles** and the **breaks (<br />)**. Add bootstrap styles to the form. Refer to **Organic/SubscriptionBS** and **Parent/VolunteerBS** views as required.

# Create the Controller

1. Right-click on the folder **Controllers** > **Add** > **Controller…**

Select **MVC Controller – Empty** and click [Add].

Enter **ConsultationController.cs** and click [Add].

Add the following bolded code to the controller, outside the namespace declaration.

|  |
| --- |
| using Lesson03.Models;  using Microsoft.AspNetCore.Mvc;  namespace Lesson03.Controllers;  public class ConsultationController : Controller  {... |

1. Replace the Index action with the following code and fill in the blanks to make the code functional.

public IActionResult Apply(string id)

{

return View("Application" + id);

}

public IActionResult Confirm()

{

IFormCollection form = HttpContext.Request.Form;

string staffName = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString().Trim();

string email = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString().Trim();

string acadPos = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString();

string startDate = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString().Trim();

string endDate = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString().Trim();

string agree = form["\_\_\_\_\_\_\_\_\_\_\_\_"].ToString();

if (ValidUtl.CheckIfEmpty(\_\_\_\_\_\_\_\_\_\_\_\_))

{

ViewData["Message"] = "Please enter or select all fields";

return View("Application");

}

if (!startDate.\_\_\_\_\_\_\_\_\_\_\_\_("yyyy-MM-dd") ||

!endDate.\_\_\_\_\_\_\_\_\_\_\_\_("yyyy-MM-dd"))

{

ViewData["Message"] = "Invalid Date Format";

return View("Application");

}

if (startDate.\_\_\_\_\_\_\_\_\_\_\_\_("yyyy-MM-dd") > endDate.\_\_\_\_\_\_\_\_\_\_\_\_("yyyy-MM-dd"))

{

ViewData["Message"] = "End date should be later than Start date";

return View("Application");

}

// Create Consultant object

Consultant cst = new();

cst.StartDate = \_\_\_\_\_\_\_\_\_\_\_\_.ToDate("yyyy-MM-dd");

cst.EndDate = \_\_\_\_\_\_\_\_\_\_\_\_.ToDate("yyyy-MM-dd");

cst.StaffName = \_\_\_\_\_\_\_\_\_\_\_\_;

cst.Email = \_\_\_\_\_\_\_\_\_\_\_\_;

cst.AcadPos = \_\_\_\_\_\_\_\_\_\_\_\_;

// Calculate Renumeration

int payment = CalcRenumeration(\_\_\_\_\_\_\_\_\_\_\_\_);

// Pass renumeration to View using ViewData

ViewData["Amount"] = \_\_\_\_\_\_\_\_\_\_\_\_;

// Pass Consultant object to View as Model

return View("Confirmation", \_\_\_\_\_\_\_\_\_\_\_\_);

}

private const int WKDAY\_PROF = 120;

private const int WKDAY\_LECT = 100;

private const int WKDAY\_ADJT = 90;

private const int WKEND\_EXTRA = 80;

private int CalcRenumeration(Consultant cst)

{

int days = (cst.EndDate - cst.StartDate).Days + 1;

int sat\_sun = CalcWeekendDays(cst.StartDate, cst.EndDate);

int amount = sat\_sun \* WKEND\_EXTRA;

if (cst.AcadPos.Equals("Professor"))

{

amount += WKDAY\_PROF \* days ;

}

else if (cst.AcadPos.Equals("Lecturer"))

{

amount += WKDAY\_LECT \* days ;

}

else if (cst.AcadPos.Equals("Adjunct"))

{

amount += WKDAY\_ADJT \* days ;

}

return amount;

}

private int CalcWeekendDays(DateTime start, DateTime end)

{

int wk = 0;

int days = (end - start).Days + 1;

for (int i = 0; i < days; i++)

{

DateTime day = start.AddDays(i);

if (day.DayOfWeek == DayOfWeek.Saturday ||

day.DayOfWeek == DayOfWeek.Sunday)

wk++;

}

return wk;

}

1. Remove the comments at the top of **Confirmation.cshtml** view to reference the Consultant model.

@model Lesson03.Models.Consultant

1. To view the bootstrap form, add an **id** of "**BS**" to the url. The "BS" is the id, which is concatonated with the word **Application** in the controller so that the correct view is called.

Graphical user interface, text, application

Description automatically generated

1. Provide your own calculation for the following scenarios and check against your program output.

|  |  |  |
| --- | --- | --- |
| 18-Nov to 25-Nov 2022 | **My Calculation** | **Program Output** |
| **Professor** | 1120 | 1120 |
| **Lecturer** | 960 | 960 |
| **Adjunct** | 720 | 720 |

1. Test the validation and see whether the appropriate messages will be output. Paste a screen shot of an Adjunct Lecturer in the space below

|  |
| --- |
|  |

* *Worksheet —*