

## Studio 5 & 6

Name and Student Id: **Vinaya Datta Kavuluri - 31131611**

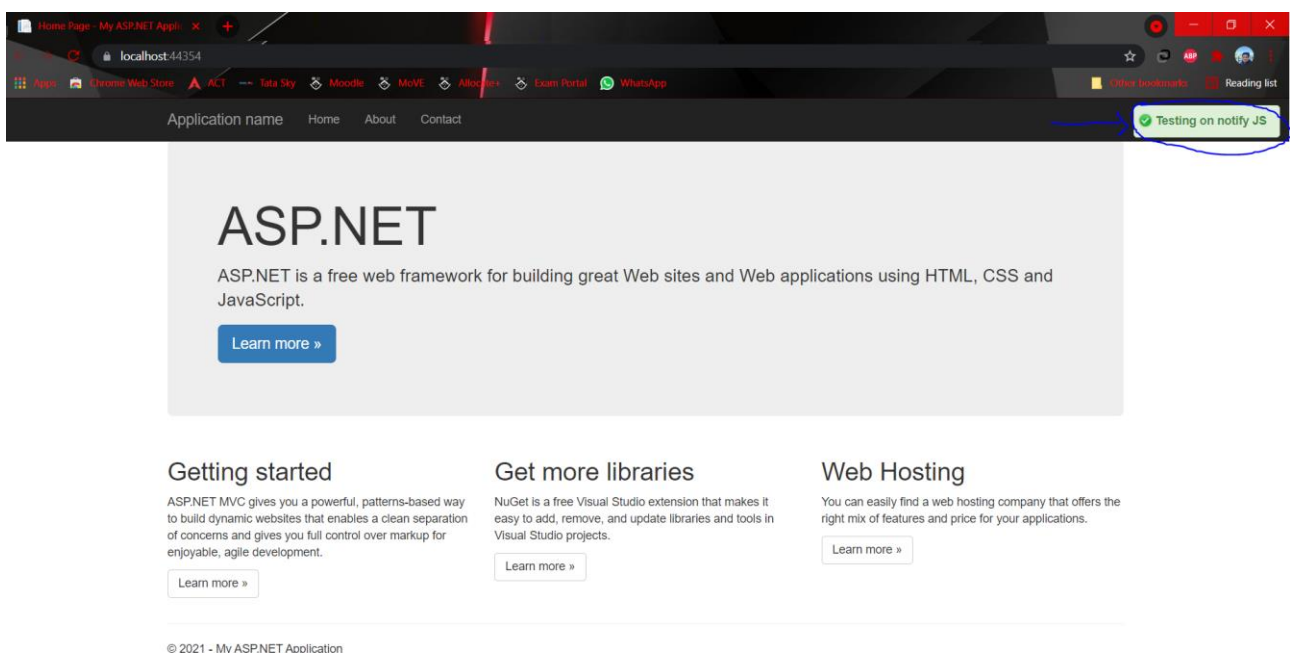
Self-Evaluation {To be highlighted by Student only}:

Need Help	Work in Progress	Pass	Credit	Distinction	High Distinction
-----------	------------------	------	--------	-------------	------------------

### TASK 5.1 (PASS AND CREDIT LEVEL)

JavaScript example – Notify.js

Screenshots of the website:



Link to code repository:

[https://github.com/KVD1302/weekly\\_activities\\_5032/tree/main/Week%205/FIT5032\\_MyJS](https://github.com/KVD1302/weekly_activities_5032/tree/main/Week%205/FIT5032_MyJS)

Difference between defer and async

In HTML5 there are two new boolean attributes for the `<script>` tag which are `async` and `defer`. These two attributes are a must for increasing speed and performance of websites. They allow the elimination of render-blocking JavaScript where the page would have to load and execute scripts before finishing to render the page.

When using `async`, the file gets downloaded asynchronously and then executed as soon as it's downloaded. When using `defer`, the file gets downloaded asynchronously, but only executes when the document parsing is completed. Here, scripts will execute in the same order as they are called. This makes `defer` the attribute of choice when a script depends on another script. For example, if you're using jQuery as well as other scripts that depend on it, you'd use `defer` on them (jQuery included), making sure to call jQuery before the dependent scripts. A good strategy is to use `async` when possible, and then `defer` when `async` isn't an option.

### References:

1. <https://www.digitalocean.com/community/tutorials/html-defer-async>
2. <https://ourcodeworld.com/articles/read/1379/what-is-the-difference-between-the-async-and-defer-attributes-in-javascript>

### TASK 5.2 (DISTINCTION AND HIGH DISTINCTION LEVEL)

Use of Bootstrap datepicker

### Screenshots:

As suggested, 20<sup>th</sup> August was set as default date. So, when ran initially the webpage shows that 20/08/2021 as the default date as seen in the figure below.

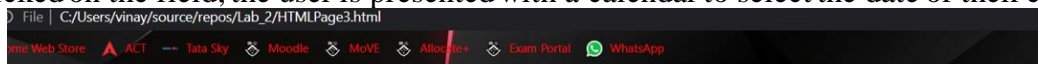


Live Demo: Bootstrap simple datepicker example

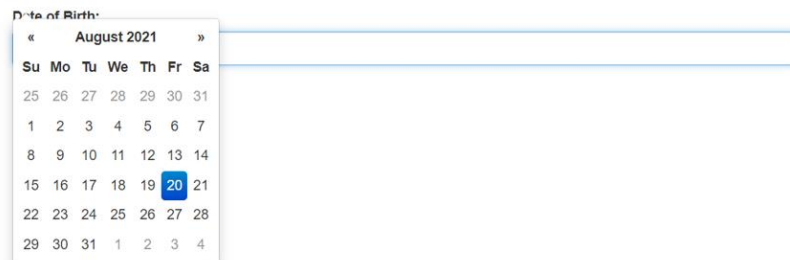
Date of Birth:

20/08/2021

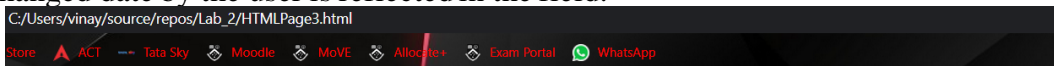
When clicked on the field, the user is presented with a calendar to select the date of their choice.



Live Demo: Bootstrap simple datepicker example



Newly changed date by the user is reflected in the field.



Live Demo: Bootstrap simple datepicker example

Date of Birth:

21/10/2021

Link to code repository:

[https://github.com/KVD1302/weekly\\_activities\\_5032/blob/main/Week%205/Datepicker.html](https://github.com/KVD1302/weekly_activities_5032/blob/main/Week%205/Datepicker.html)

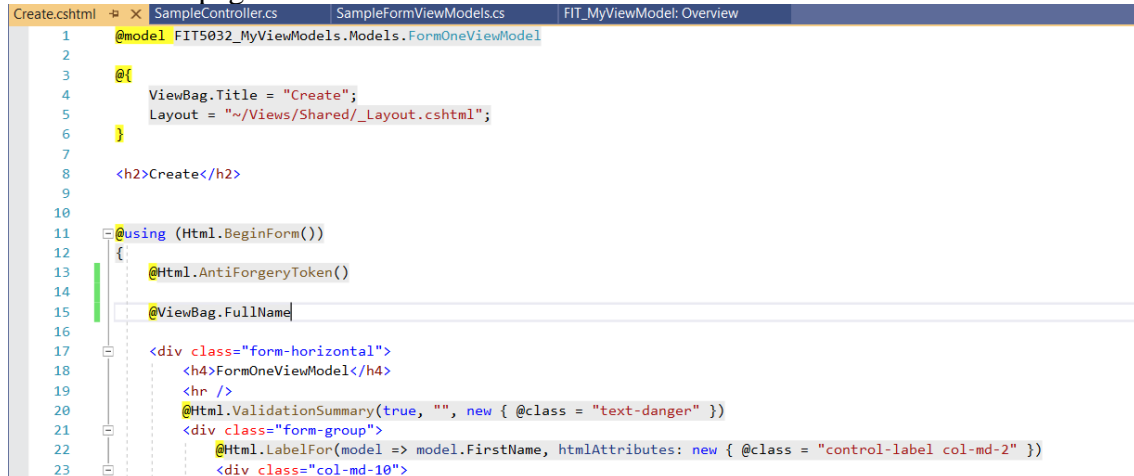
## TASK 6.1 (PASS AND CREDIT LEVEL)

Application of View Model with validations

Screenshots:

Code:

Screenshot of create page with validations

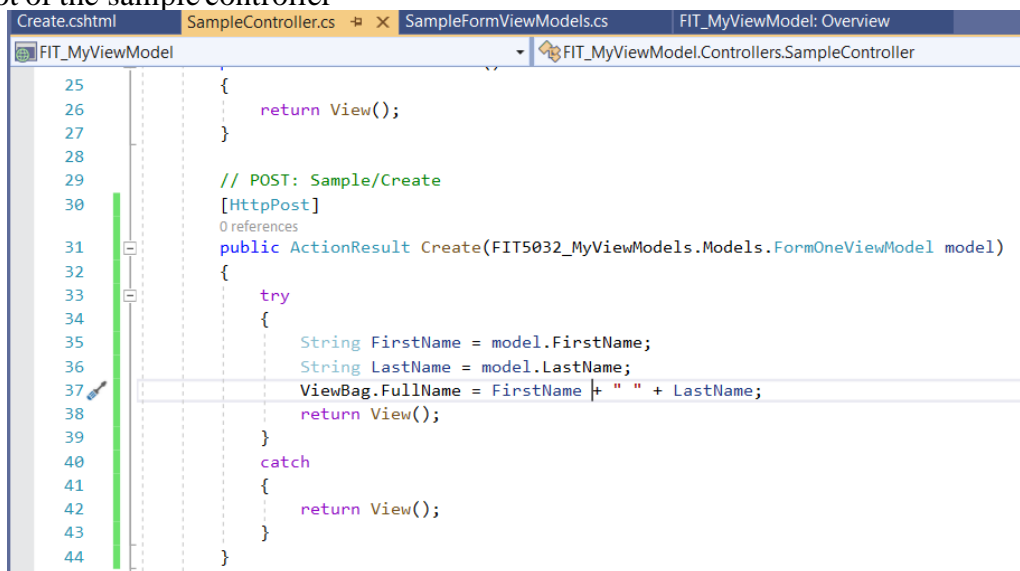


```

1  @model FIT5032_MyViewModels.Models.FormOneViewModel
2
3  @{
4      ViewBag.Title = "Create";
5      Layout = "~/Views/Shared/_Layout.cshtml";
6  }
7
8  <h2>Create</h2>
9
10
11 @using (Html.BeginForm())
12 {
13     @Html.AntiForgeryToken()
14
15     @ViewBag.FullName
16
17     <div class="form-horizontal">
18         <h4>FormOneViewModel</h4>
19         <hr />
20         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
21         <div class="form-group">
22             @Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })
23             <div class="col-md-10">

```

Screenshot of the sample controller

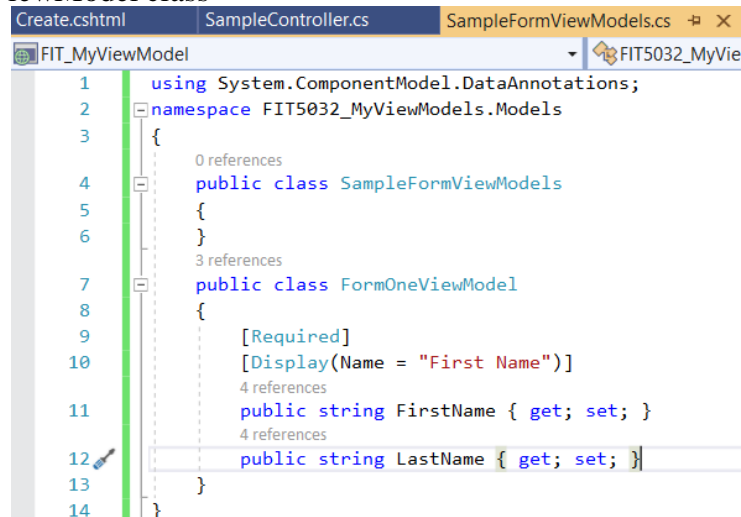


```

25 {
26     return View();
27 }
28
29 // POST: Sample/Create
30 [HttpPost]
31 public ActionResult Create(FIT5032_MyViewModels.Models.FormOneViewModel model)
32 {
33     try
34     {
35         String FirstName = model.FirstName;
36         String LastName = model.LastName;
37         ViewBag.FullName = FirstName + " " + LastName;
38         return View();
39     }
40     catch
41     {
42         return View();
43     }
44 }

```

Screenshot of FormViewModel class



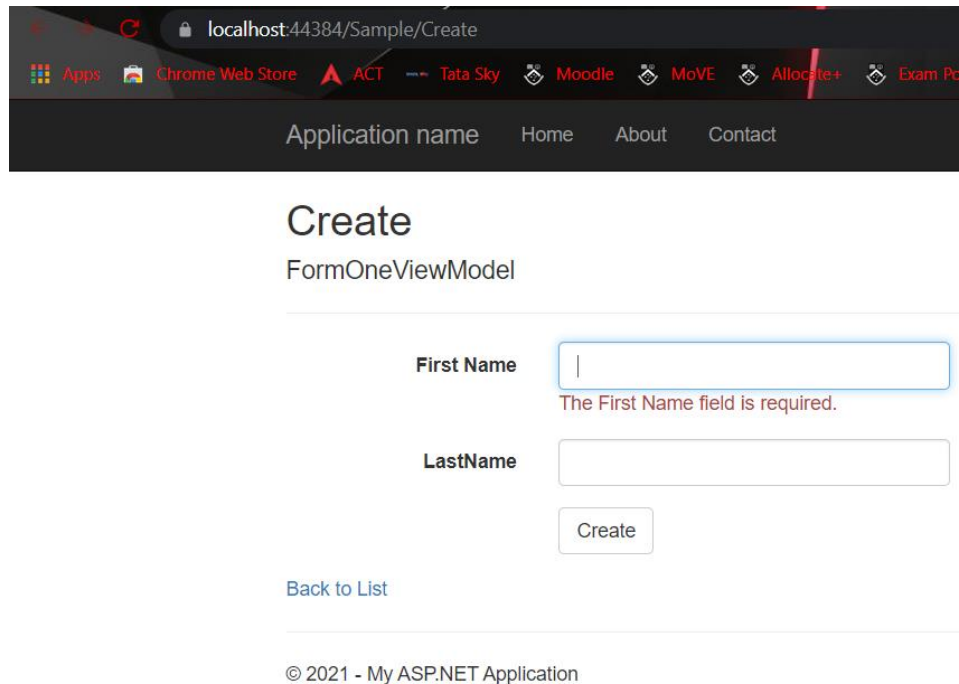
```

1  using System.ComponentModel.DataAnnotations;
2  namespace FIT5032_MyViewModels.Models
3  {
4      public class SampleFormViewModels
5      {
6      }
7      public class FormOneViewModel
8      {
9          [Required]
10         [Display(Name = "First Name")]
11         public string FirstName { get; set; }
12         public string LastName { get; set; }
13     }
14 }

```

**Output:**

This screenshot shows how the application requests the user to atleast enter the 'first name' to create the instance.



localhost:44384/Sample/Create

Apps Chrome Web Store ACT Tata Sky Moodle MoVE Allocate+ Exam Po

Application name Home About Contact

## Create

FormOneViewModel

**First Name**

The First Name field is required.

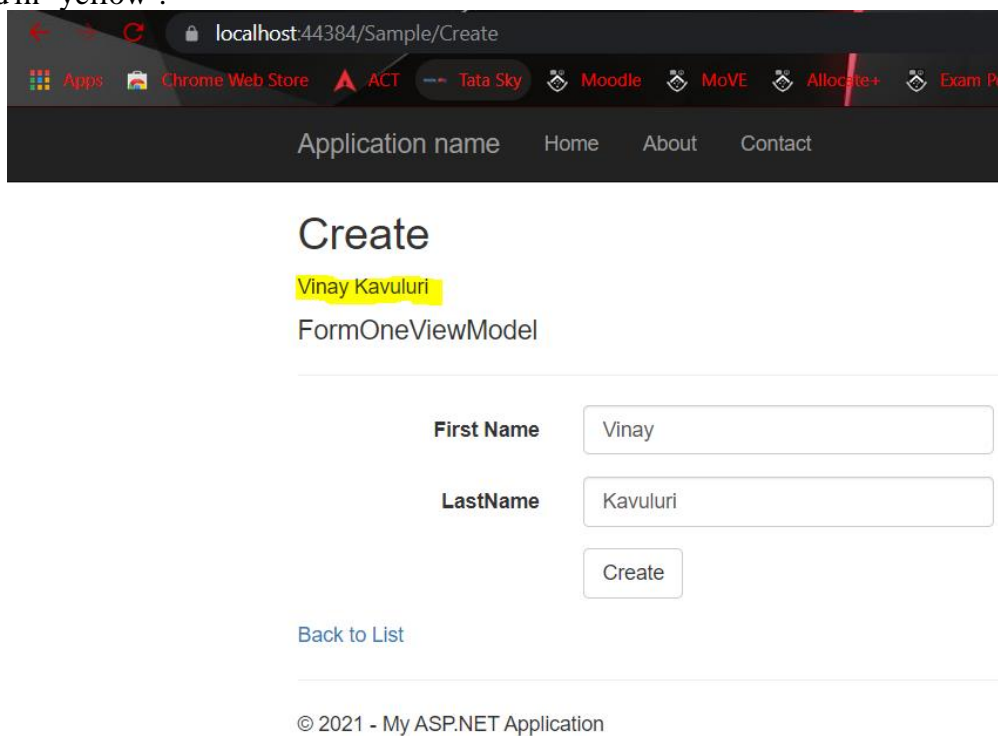
**LastName**

Create

[Back to List](#)

© 2021 - My ASP.NET Application

Now that the user enters a valid input, the application displays the entered name on top as highlighted in 'yellow'.



localhost:44384/Sample/Create

Apps Chrome Web Store ACT Tata Sky Moodle MoVE Allocate+ Exam Po

Application name Home About Contact

## Create

Vinay Kavuluri

FormOneViewModel

**First Name**

**LastName**

Create

[Back to List](#)

© 2021 - My ASP.NET Application

Link to code repository:

[https://github.com/KVD1302/weekly\\_activities\\_5032/tree/main/Week%206/FIT\\_MyViewModel](https://github.com/KVD1302/weekly_activities_5032/tree/main/Week%206/FIT_MyViewModel)

## TASK 6.2 (DISTINCTION AND HIGH DISTINCTION LEVEL)

### Understanding jQuery unobstructive validation

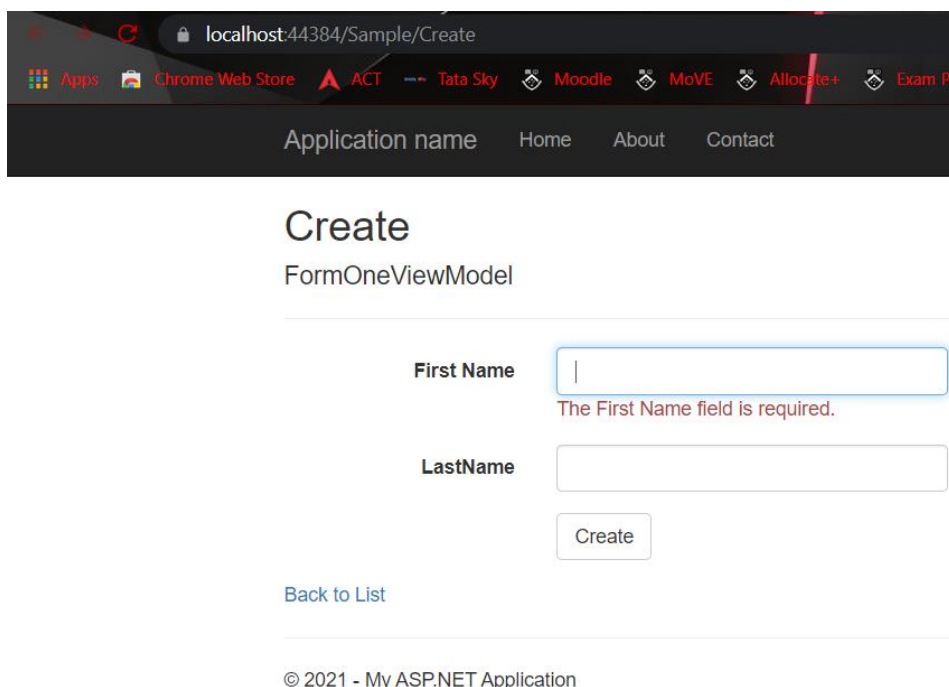
In MVC web applications, Unobtrusive Validation is implemented with the help of the jQuery Validate plugin and the Unobtrusive library. This allows us to add validation to our MVC views without using any additional client-side coding.

If a class is already set up to handle server-side validation using attributes, Unobtrusive Validation allows us to take the already-existing validation attributes and use them client-side as well to make our user experience that much nicer. The Unobtrusive script files are usually included by default with new MVC projects in Visual Studio.

Without Unobtrusive Validation, if we attempt to submit a form with an input error/ unacceptable criteria, the Controller catches the errors in its action. Here, we should render the Validation Message as well as place an error message in CSS class on the input. But with Unobtrusive Validation, this gets a lot simpler. It uses HTML5-compatible "data-" attributes to store all the information it needs to perform this validation. Therefore, we don't need to use any other code besides attributes to enable client-side validation with this library.

The only difference between the server-side validation implementation and its client-side counterpart using Unobtrusive validation is the inclusion of the 'jqueryval' bundle on the latter. By implementing Unobtrusive validation on client side, the browser doesn't make a round-trip to the server on validation failure, and it boosts the user experience of the application.

This is the screenshot from previous exercise which depicts the client-side validation performed by using unobstructive validation.



localhost:44384/Sample/Create

Apps Chrome Web Store ACT Tata Sky Moodle MoVE Allocate+ Exam Po

Application name Home About Contact

## Create

FormOneViewModel

First Name

The First Name field is required.

LastName

Create

[Back to List](#)

© 2021 - My ASP.NET Application

### References:

1. <https://exceptionnotfound.net/asp-net-mvc-demystified-unobtrusive-validation/>
2. <https://www.geeksforgeeks.org/what-is-unobtrusive-validation-in-jquery/>