**ViewImports in ASP.NET Core MVC**

**The ASP.NET Core MVC and Razor comes with a lot of new advanced features for working with the Razor views. ViewImports is one of the new features, we are going to discuss the following pointers.**

1. **What is \_ViewImports.cshtml?**
2. **Understanding ViewImports with an example.**
3. **Creating ViewImports.cshtml file in ASP.NET Core MVC Application.**
4. **Understanding the hierarchical Order of ViewImports file in ASP.NET Core MVC.**

**What is \_ViewImports.cshtml in ASP.NET Core MVC Application?**

**In ASP.NET Core MVC Application, the \_ViewImports.cshtml file provides a mechanism to include the directives globally for Razor Pages so that we don’t have to add them individually in each and every page. As of this time, the \_ViewImports.cshtml file supports the following directives:**

1. **@addTagHelper**
2. **@tagHelperPrefix**
3. **@removeTagHelper**
4. **@namespace**
5. **@inject**
6. **@model**
7. **@using**

* **The @addTagHelper, @tagHelperPrefix, and @removeTagHelper directives are basically used to manage of Tag Helpers.**
* **The @namespace directive is basically used to specify the namespace that the ViewImports belongs to.**
* **With the help of @inject directive, it supports Dependency injection.**
* **We already use the @model directive in our previous applications when we are working with models. The @model directive is basically used to specify the Model for your view.**
* **The @using directive basically used to include the common namespaces globally so that you don’t have to include the namespaces in each and every view page.**

**Note: We are using only @using directive in ViewImports.cshtml file. Rest directives are going to be discussing in upcoming classes.**

**Let us understand ViewImports with an example**

**Create a model called Publishers within the Models folder of your application. Once you create the Publishers.cs class file then copy and paste the following code in it.**

**namespace CodeBitsPublisher.Models**

**{**

**public class Publisher**

**{**

**public int PublisherId { get; set; }**

**public string PublisherName { get; set; }**

**public string Phone { get; set; }**

**public string Address { get; set; }**

**public string City { get; set; }**

**}**

**}**

**As you can see, here we created the Publisher model with five properties.**

**Modifying the Home Controller:**

**Modify the Home Controller as shown below.**

**using CodeBitsPublisher.Models;**

**using Microsoft.AspNetCore.Mvc;**

**using System.Collections.Generic;**

**namespace CodeBitsPublisher.Controllers**

**{**

**public class HomeController : Controller**

**{**

**public ViewResult Index()**

**{**

**List<Publisher> listPublishers = new List<Publisher>()**

**{**

**new Publisher() { PublisherId = 111, PublisherName = "Unilag Press", Phone = "0904-678-6786", Address = "Shomolu, Bariga", City = "Shomolu" },**

**new Publisher() { PublisherId = 112, PublisherName = "Ota Press", Phone = "0802-678-4566", Address = "Iju Atan Road", City = "Ota" },**

**new Publisher() { PublisherId = 113, , PublisherName = "Etim Press Ltd", Phone = "0705-234-7843", Address = "Sabo Bus Stop, Yaba", City = "Lagos" },**

**new Publisher() { PublisherId = 114, PublisherName = "AY Publisher", Phone = "0804-678-5467", Address = "Itire Surulere", City = "Surulere" },**

**new Publisher() { PublisherId = 115, PublisherName = "Oyo Press", Phone = "0904-678-6786", Address = "Apata Road", City = "Apata" }**

**};**

**return View(listPublishers);**

**}**

**public ViewResult Details(int Id)**

**{**

**var PublisherDetails = new Publisher() { PublisherId = Id, , PublisherName = "Ota Press", Phone = "0802-678-4566", Address = "Iju Atan Road", City = "Ota" };**

**return View(PublisherDetails);**

**}**

**}**

**}**

**As you can see, here we have two action methods. One action method is used to display all the Publisher data while the other action method takes the Publisher id as a parameter and return that Publisher information.**

**Modifying the Index and Details view:**

**Index.cshtml:**

**@model List<CodeBitsPublisher.Models.Publisher>**

**@{**

**Layout = null;**

**}**

**<html>**

**<head>**

**<title>Index</title>**

**</head>**

**<body>**

**<table>**

**<thead>**

**<tr>**

**<th>Publisher ID</th>**

**<th>Publisher Name</th>**

**<th>Phone</th>**

**<th>Address</th>**

**<th>City</th>**

**</tr>**

**</thead>**

**<tbody>**

**@foreach (var Publisher in Model)**

**{**

**<tr>**

**<td>**

**@Publisher.PublisherId**

**</td>**

**<td>**

**@Publisher.PublisherName**

**</td>**

**<td>**

**@Publisher.Phone**

**</td>**

**<td>**

**@Publisher.Address**

**</td>**

**<td>**

**@Publisher.City**

**</td>**

**</tr>**

**}**

**</tbody>**

**</table>**

**</body>**

**</html>**

**Details.cshtml:**

**@model CodeBitsPublisher.Models.Publisher**

**@{**

**Layout = null;**

**}**

**<html>**

**<head>**

**<title>Publisher Detaills</title>**

**</head>**

**<body>**

**<div>**

**Publisher Id : @Model.PublisherId**

**</div>**

**<div>**

**Publisher Name : @Model.PublisherName**

**</div>**

**<div>**

**Phone : @Model.Phone**

**</div>**

**<div>**

**Address : @Model. Address**

**</div>**

**<div>**

**City : @Model.City**

**</div>**

**</body>**

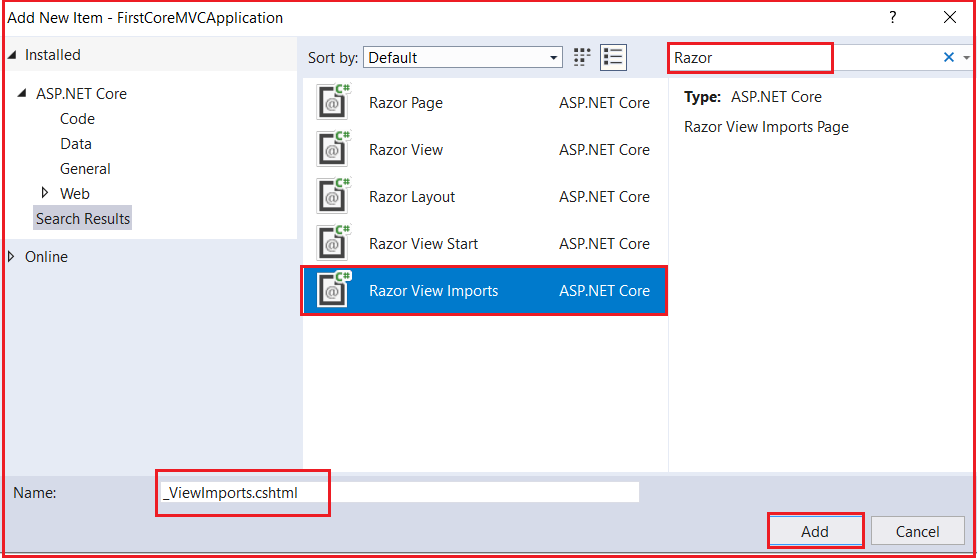
**</html>**

**In the above Index and Details view, we are using the @model directive to specify the model for the view. If you notice, then you can see in both the views we have specified the fully qualified name for the model such as “CodeBitsPublisher.Models.Publisher”.**

**Now let us see how to move the namespace to the ViewImports file so that we can only specify the model name.**

**Creating ViewImports.cshtml file in ASP.NET Core MVC Application**

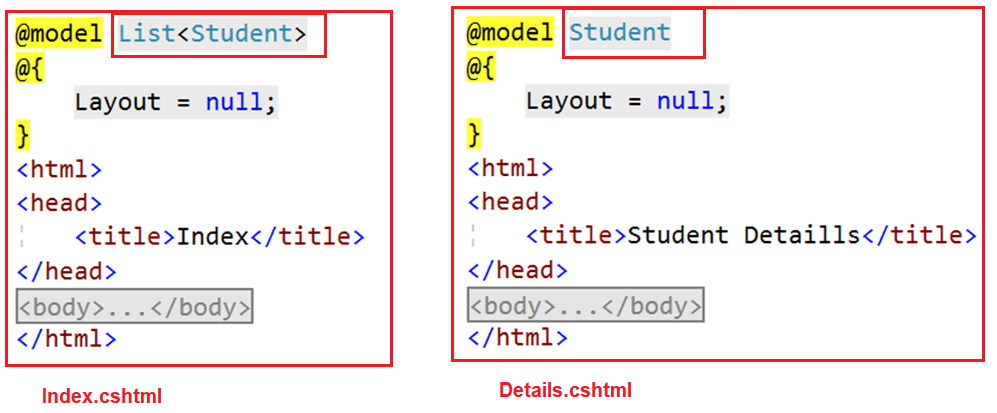
**In general, \_ViewImports.cshtml files are created within the Views or within the subfolder of the Views folder. To create the “\_ViewImports.cshtml” file right-click on the Views folder and then select the “Add – New Item” option from the context menu, which will open the “New Item” window. From the “New Item” window search for “Razor” and then select the “Razor View Import” and click on the “Add” button as shown in the below image which should create the “\_ViewImport.cshtml” within the “Views” folder.**

****

**Once the \_ViewImports.cshtml file is created, then copy and paste the following code in it.**

**@using CodeBitsPublisher.Models;**

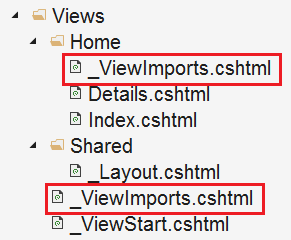
**As we placed the above namespace in the ViewImports file, now all the types that are present in the above namespace are available to each and every view in the “Home” folder. So now we don’t need to type the fully qualified name of the Type. So, modify the Index and Details view as shown below.**

****

**As you can see in the above image, we are removing the namespace and only specified the model name. Run the application and it should work as expected.**

**\_ViewImports file is hierarchical Order in ASP.NET Core MVC:**

**Just like the \_ViewStart file, the \_ViewImports file is also hierarchical. It is also possible to pace the \_ViewImports in the subfolder of the Views folder as shown in the below image. Here we have one \_ViewImports file in the Views folder and another \_ViewImports file within the Home folder.**

****

**The settings that are specified in the \_ViewImports file present in the Home subfolder will overwrite the settings specified in the \_ViewImports file in the Views folder.**

**Note: If you specified a setting in the view itself, then that setting overrides the matching settings specified in the parent \_ViewImports files in the folder hierarchy.**