MVC

Christen Zarif

outline

- Routing
- Controller
- Action
- View
- Model
- Razor

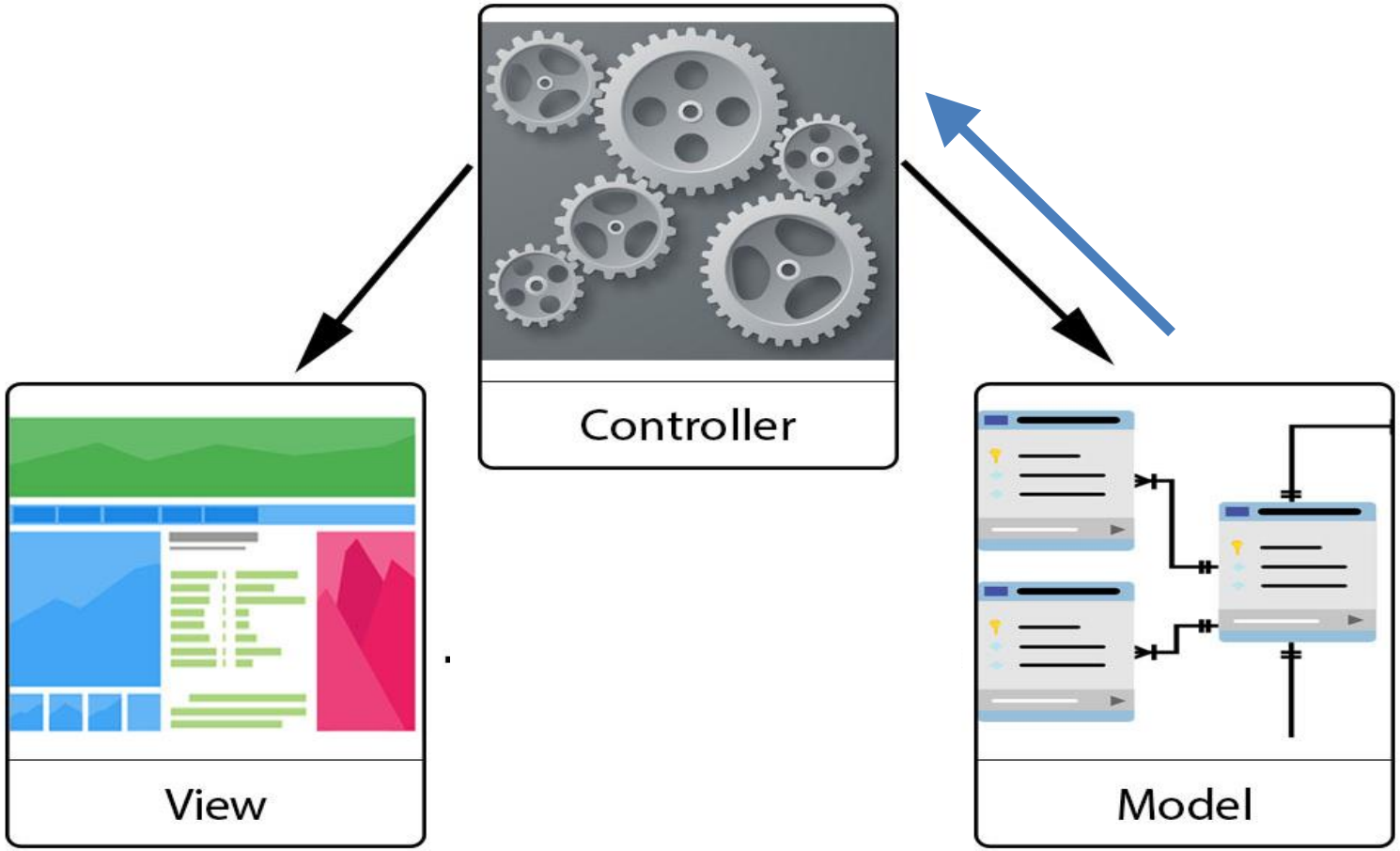
ASP.NET MVC

- ASP.NET MVC is an ***open-source software*** from Microsoft. Its web development framework combines the features of MVC (Model-View-Controller) architecture, the most up-to-date ideas and techniques from ***Agile development*** is it allows for parallel development, and the best parts of the existing ASP.NET platform.

Asp.net MVC Framework Components

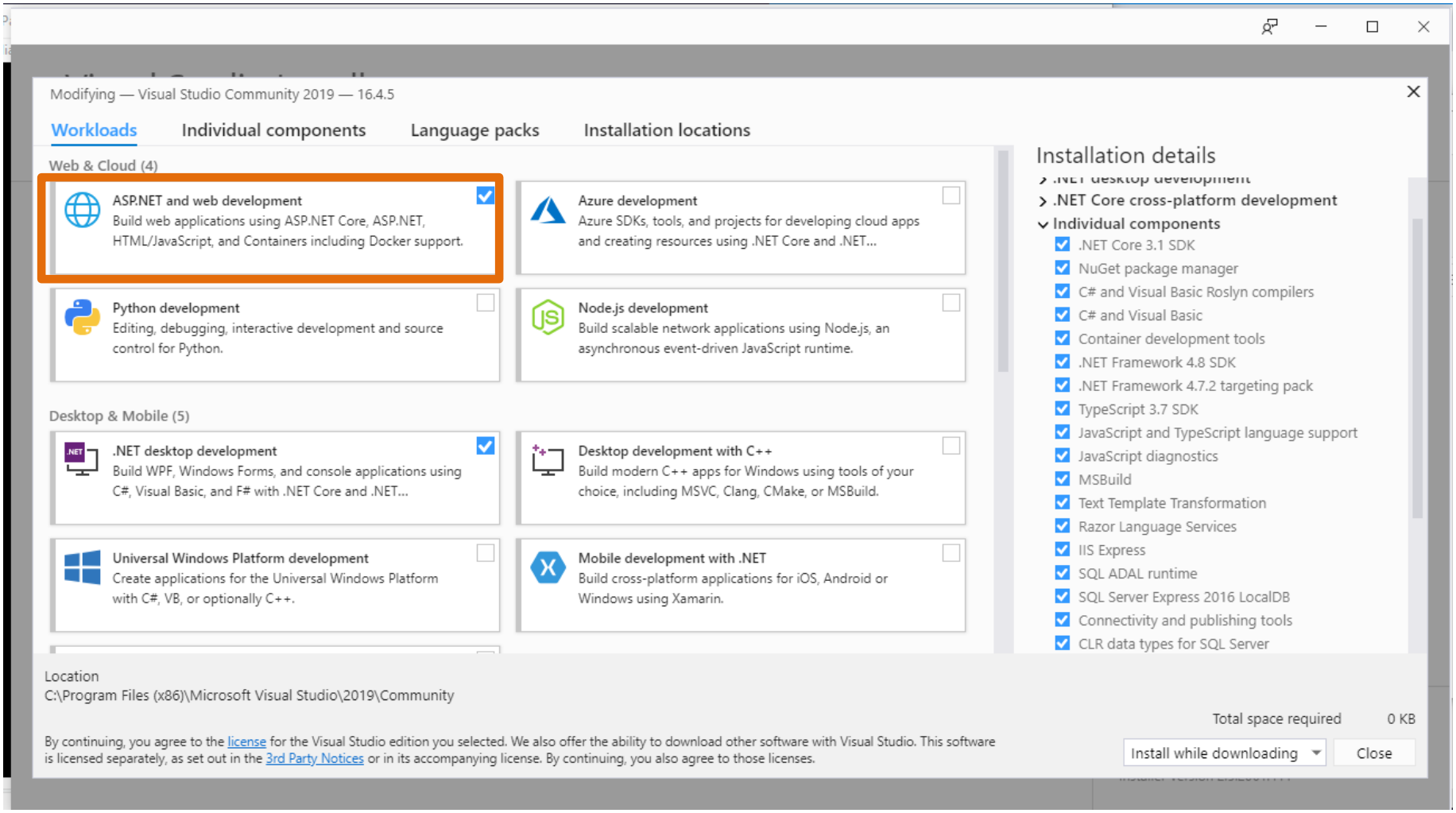
- **Model:**
  - Business/ Domain Logic
  - Model Objects, retrieve and store model state in a persistent Storage (database)
- **Views :**
  - Display applications UI
  - UI created from the model data
- **Controllers:**
  - Handle user input and interaction (Handling Http Request)
  - Work with model
  - Select a view for rendering UI

http://vidly.com/movies



LET’S CREATE FIRST MVC WEB SITE

To create Web application



ASP.NET MVC Folder Structure

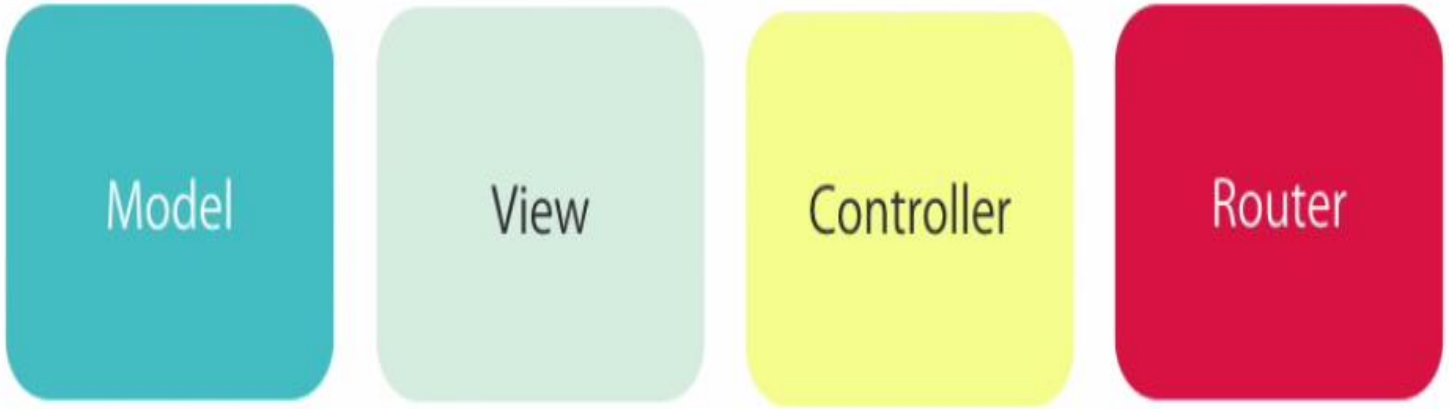
Folder Name	Description
Dependencies	• is the place where the necessary dll.s for the application are stored
Properties	• **launchSettings.json:** file which includes Visual Studio profiles of debug settings. We can also edit settings from the debug tab of project properties.
wwwroot	• Static content is hosted in the **wwwroot** folder. The content such as CSS, Javascript files and Bootstrap, jquery libraries need to be included here. • **Static files** can be stored in any folder under the web root and accessed with a relative path to that root • Now, only those files that are in the web root - wwwroot folder can be served over an http request. All other files are blocked and cannot be served by default
appsettings.json	• is used to store information such as connection strings or application specific settings and these are stored in the JSON format as the file extension suggests.

IIS EXPRESS

ASP.NET MVC Folder Structure(con.)

Folder Name	Description
Program.cs	• is the main entry point for the application. It then goes to the Startup.cs class to finalize the configuration of the application.
Startup.cs	• **includes Configure and ConfigureServices methods** • Startup class is where: Services required by the app are configured. The request handling pipeline is defined.

MVC Architectural Pattern

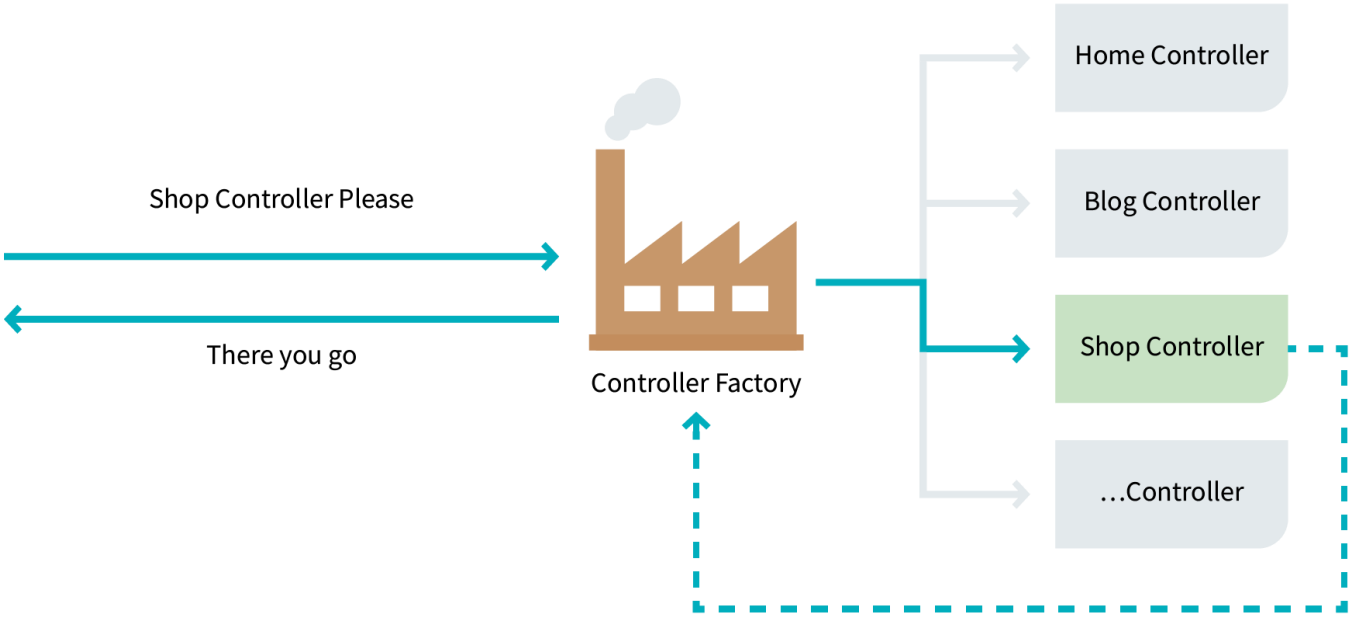


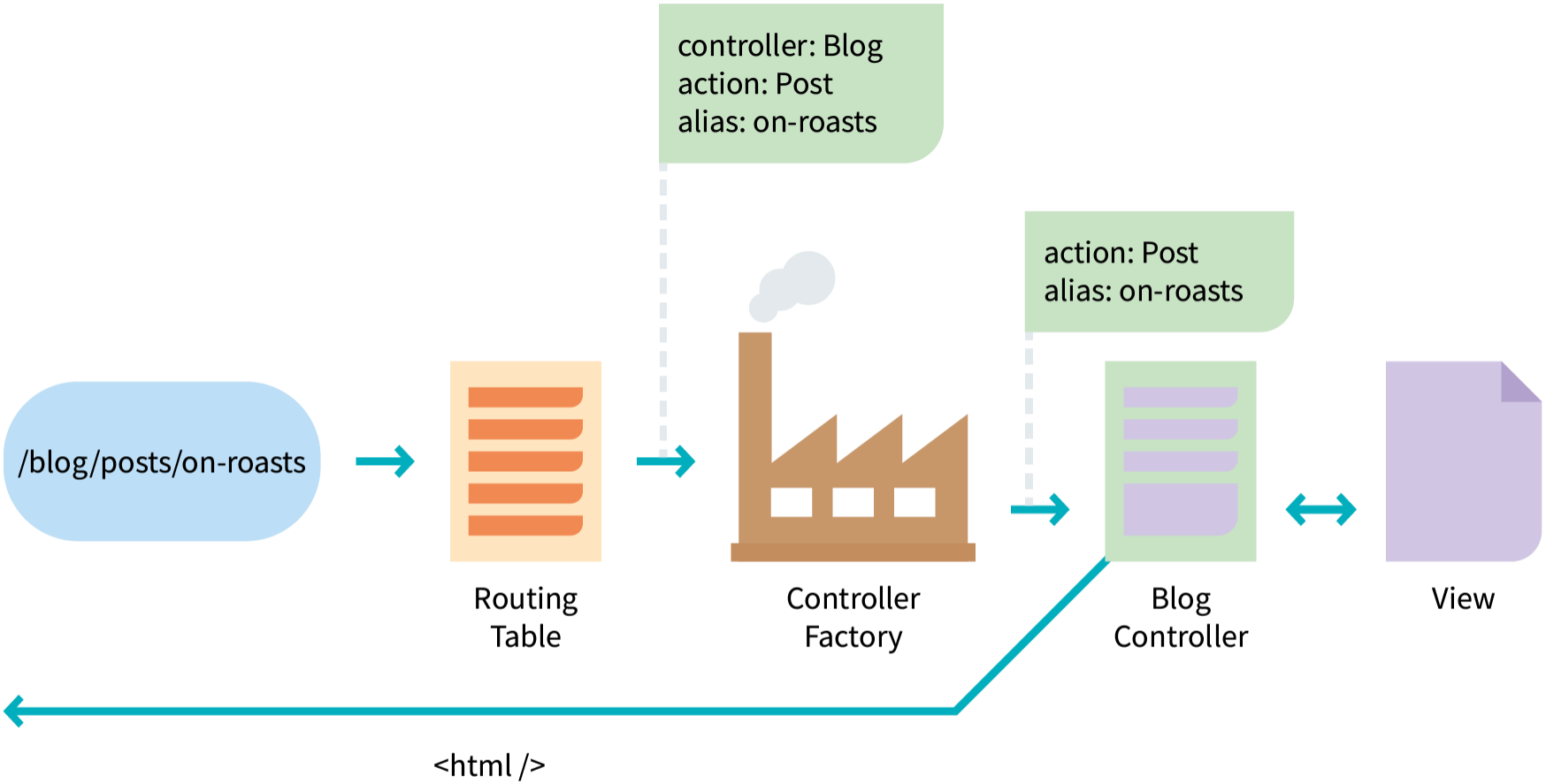
Controller

- The Controller in MVC architecture **handles any incoming URL request**.
- Controller is a class, derived from the base class *Microsoft.AspNetCore.Mvc.Controller*.
- Controller class contains public methods called ***Action*** methods.
- Controller and its action method handles incoming browser requests, **retrieves** necessary model data and **returns** appropriate responses.
- In ASP.NET MVC, every controller class name must end with a word "Controller".
  - For example, controller for home must be ***HomeController*** and controller for student must be ***StudentController***.
- every controller class must be located in **Controller folder** of MVC folder structure.

DefaultControllerFactory

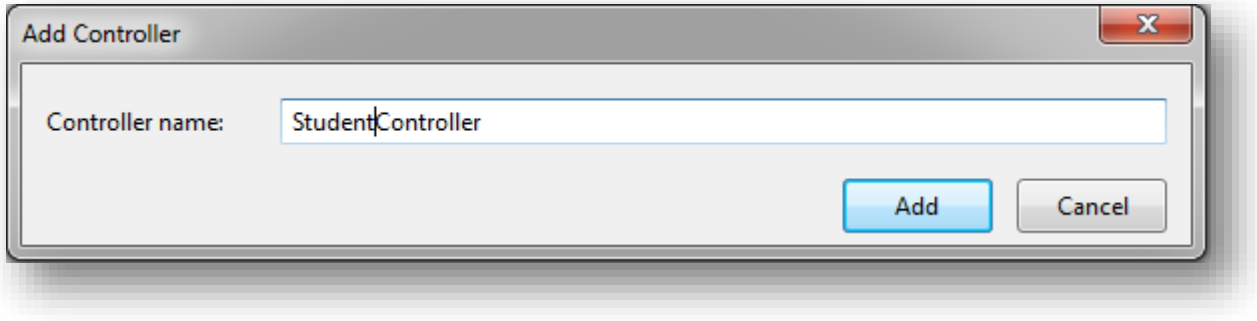
- ASP.NET MVC uses DefaultControllerFactory class for creating controller after receiving request from Route Handler.




controller: Blog
action: Post
alias: on-roasts
action: Post
alias: on-roasts
/blog/posts/on-roasts
Routing Table
Controller Factory
Blog Controller
View
<html />

Scaffolding

- Scaffolding is an automatic code generation framework for ASP.NET web applications.
- Scaffolding reduces the time taken to develop a controller, view etc

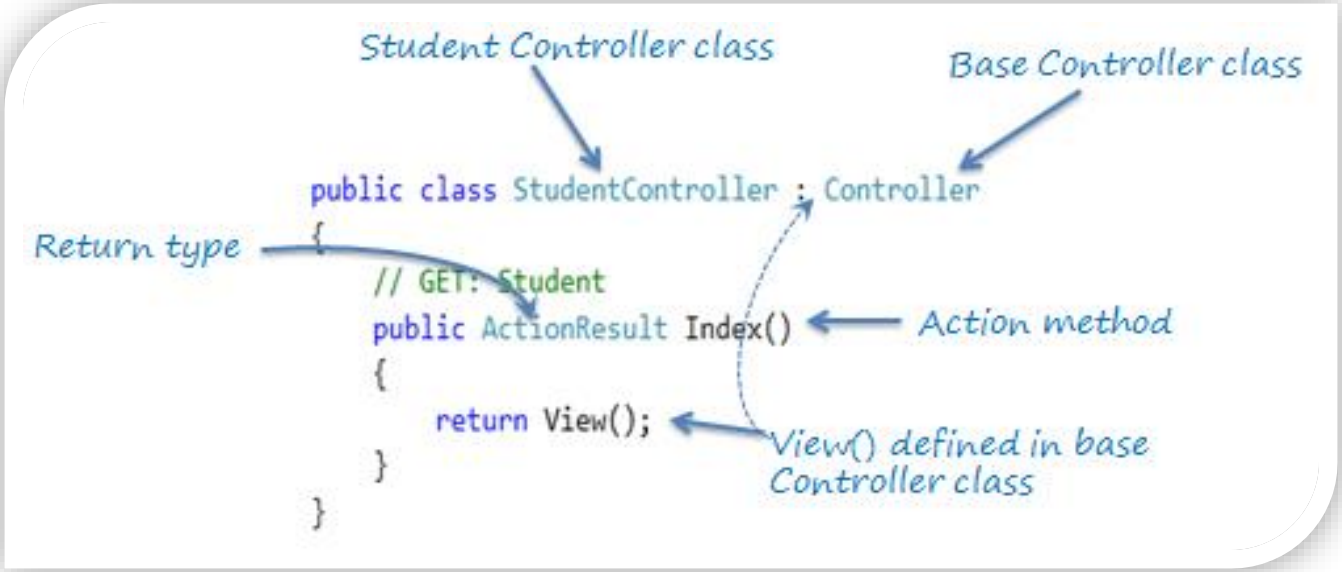


Points to Remember

- A Controller handles incomming URL requests. MVC routing sends request to appropriate controller and action method based on URL and configured Routes.
- All the public methods in the Controlle class are called **Action** methods.
- A Controller class must be derived from Microsoft.AspNetCore.Mvc.***Controller*** class.
- A Controller class name must end with "Controller".
- New controller can be created using different scaffolding templates. You can create custom scaffolding template also.

Action method

- Public methods of a Controller class
- Action method must be public. It **cannot** be private or protected
- Action method **cannot** be overloaded
- Action method **cannot** be a static method.



ActionResult

- The ActionResult class is a base class of all the above result classes, so it can be return type of action methods which returns any type of result listed above
- MVC framework includes various result classes, which can be return from an action methods
- There result classes represent **different types of Responses** such as html, file, string, json, javascript etc.

ActionResult (Con.)

Result Class	Description	Base Controller method
ViewResult	Represents HTML and markup.	View()
EmptyResult	Represents No response.	
ContentResult	Represents string literal.	Content()
FileContentResult,FilePathResult,FileStreamResult	Represents the content of a file	File()
JavaScriptResult	Represent a JavaScript script.	JavaScript()
JsonResult	Represent JSON that can be used in AJAX	Json()
RedirectResult	Represents a redirection to a new URL	Redirect()
RedirectToRouteResult	Represent another action of same or other controller	RedirectToRoute()
PartialViewResult	Returns HTML	PartialView()
HttpUnauthorizedResult	Returns HTTP 403 status	

```
using System.Web.Mvc;
using Vidly.Models;

namespace Vidly.Controllers
{
    public class MoviesController : Controller
    {
        // GET: Movies/Random
        public ActionResult Random()
        {
            var movie = new Movie() { Name = "Shrek!" };

//          return View(movie);
//          return Content("Hello World!");
//          return HttpNotFound();
//          return new EmptyResult();
            return RedirectToAction("Index", "Home", new { page = 1});
        }
    }
}
```

No sugg

Default Action method

- http://IP/Home/Index
- **Index** is a default action method for any controller, as per configured default root
- you can change the default action name as per your requirement in RouteConfig class.

Parameter Binding

Parameter Sources

- In the URL: `/movies/edit/1`
- In the query string: `/movies/edit?id=1`
- In the form data: `id=1`

Example

- **Request Url**:
  - Localhost:3333/home/edit/1
  - Localhost:3333/home/edit?id=1

```
public ActionResult Edit(int id)
{
    return Content("id=" + id);
}
```

Example 2

Action	URL	Source

`public ActionResult Edit(int id)`
`{`
`  return Content("id=" + id);`
`}`
	Localhost:3333/home/edit/1	Route Parameter

	Localhost:3333/home/edit?id=1	Query String

`public ActionResult Edit(int movieId)`
`{`
`  return Content("id=" + movieId);`
`}`		

	Localhost:3333/home/edit?movieId=1	Query String

Common Exception

The parameters dictionary contains a null entry for parameter 'movieId' of non-nullable type 'System.Int32' for method 'System.Web.Mvc.ActionResult Edit(Int32)' in 'Vidly.Controllers.MoviesController'. An optional parameter must be a reference type, a nullable type, or be declared as an optional parameter.
Parameter name: parameters

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

- To solve this exception send nullable Parameter to action method
  - **Public ActionResult Edit(int? movieId)**

Action method Parameters:

- Every action methods can have input parameters as normal methods. It can be **primitive** data type or **complex** type parameters
- Action method can include Nullable type parameters.
- the values for action method parameters are **retrieved from the request's data collection**.
- The **data collection** includes name/values pairs for
  - **Form Data or**
  - **Query String values or**
  - **Route.**
- **Model binding** in ASP.NET MVC automatically maps the URL query string or form data collection to the action method parameters if both names are matching.

Model

- Model represents domain specific data and business logic in MVC architecture
- Adding Model:
  - Right click on Model folder -> Add -> click on Class

View

- View is **a user interface**. View displays data from the model to the user and also enables them to modify the data.

- Every view in the ASP.NET MVC is derived from RazorPage<TModel> class included in Microsoft.AspNetCore.Mvc.Razor namespace

- The Views folder contains :
  - A **separate folder** for each **controller** with the same name as controller
  - **Shared folder** contains views, layouts or partial views which will be shared among multiple views

Razor view engine

- Microsoft introduced the Razor view engine and packaged with MVC 3.
- You can write a **mix** of html tags and server side code using C# or Visual Basic in razor view.
- Razor uses **@** character for server side code instead of traditional **<% %>.**
- Razor views files have .cshtml or vbhtml extension

Razor syntax

Html

Html helper

```
@model IEnumerable<MVC_BasicTutorials.Models.Student>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.StudentName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Age)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.StudentName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Age)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.StudentId }) |
```

Razor

- Razor is one of the **view engine** supported in ASP.NET MVC.
- Razor view with visual basic syntax has .vbhtml file extension and C# syntax has .cshtml file extension
- **Razor syntax has following Characteristics:**
  - **Compact**: Razor syntax is compact which enables you to minimize number of characters and keystrokes required to write a code @.
  - **Easy to Learn**: Razor syntax is easy to learn where you can use your familiar language C# or Visual Basic.
  - **Intellisense**: Razor syntax supports statement completion within Visual Studio.

C# Razor Syntax

- Inline expression:
  - Start with @ symbol to write server side C# or VB code with Html code.
  - For example, write @Variable_Name to display a value of a server side variable.
  - A single line expression does not require a semicolon at the end of the expression.

C# Razor Syntax:

```
<h1>Razor syntax demo</h1>

<h2>@DateTime.Now.ToShortDateString()</h2>
```

Result:

```
Razor syntax demo
08-09-2014
```

C# Razor Syntax (con.)

- Multi-statement Code block:
  - You can write multiple line of server side code enclosed in braces @{ ... }.
  - Each line must ends with semicolon same as C#.

C# Razor Syntax:

```
@{
    var date = DateTime.Now.ToShortDateString();
    var message = "Hello World";
}

<h2>Today's date is: @date </h2>
<h3>@message</h3>
```

Result:

```
Razor syntax demo
Today's date is: 08-09-2014
Hello World!
```

C# Razor Syntax (con.)

- **Display text from code block**:
  - Use **@:** or **<text></text>** to display texts within code block

C# Razor Syntax:

```
@{
    var date = DateTime.Now.ToShortDateString();
    string message = "Hello World!";
    @:Today's date is: @date <br />
    @message
}
```

Razor Syntax:

```
@{
    var date = DateTime.Now.ToShortDateString();
    string message = "Hello World!";
    <text>Today's date is:</text> @date <br />
    @message
}
```

Result:

```
Razor syntax demo
Today's date is: 08-09-2014
Hello World!
```

C# Razor Syntax (con.)

- **if-else condition:**
  - Write if-else condition starting with @ symbol. The if-else code block *must be enclosed in braces { },* even for single statement.

Razor Syntax:

```
@if(DateTime.IsLeapYear(DateTime.Now.Year) )
{
    @DateTime.Now.Year @:is a leap year.
}
else {
    @DateTime.Now.Year @:is not a leap year.
}
```

Result:

```
2014 is not a leap year.
```

C# Razor Syntax (con.)

- **For loop**

Razor Syntax:

```
@for (int i = 0; i < 5; i++) {
    @i.ToString() <br />
}
```

Result:

```
0
1
2
3
4
```

C# Razor Syntax (con.)

- Model:
  - Use **@model** to use model object anywhere in the view.

C# Razor Syntax:

```
@model Student

<h2>Student Detail:</h2>
<ul>
    <li>Student Id: @Model.StudentId</li>
    <li>Student Name: @Model.StudentName</li>
    <li>Age: @Model.Age</li>
</ul>
```

Result:

Student Detail:

- Student Id: 1
- Student Name: John
- Age: 18

C# Razor Syntax (con.)

- **Declare Variables**:
  - Declare a variable in a code block enclosed in brackets and then use those variables inside html with @ symbol.

C# Razor Syntax:

```
@{
    string str = "";

    if(1 > 0)
    {
        str = "Hello World!";
    }
}

<p>@str</p>
```

Result:

```
Hello World!
```

Strong type view

- Pass data using model object
- Model:

```
public class Movie
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

- Controller

```
// GET: Movies/Random
public ActionResult Random()
{
    var movie = new Movie() { Name = "Shrek!" };

    return View(movie);
}
```

- View

```
@model Vidly.Models.Movie
@{
    ViewBag.Title = "Random";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>@Model.Name</h2>
```

@model directive

- The @model directive provides a cleaner and more concise way to reference strongly-typed models from view files
- Razor will derive the view from the Microsoft.AspNetCore.Mvc.Razor.RazorPage<TModel> base class.

Improve your Information

- How to convert HTTP to HTTPS

☺ Thank You ☺