ASP.NET MVC INTRODUCTION

Tran Khai Thien

MVC Architecture

- MVC stands for **Model**, **View** and **Controller**. MVC separates application into three components - Model, View and Controller.
- **Model**: Model represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database.

  Model is a data and business logic.

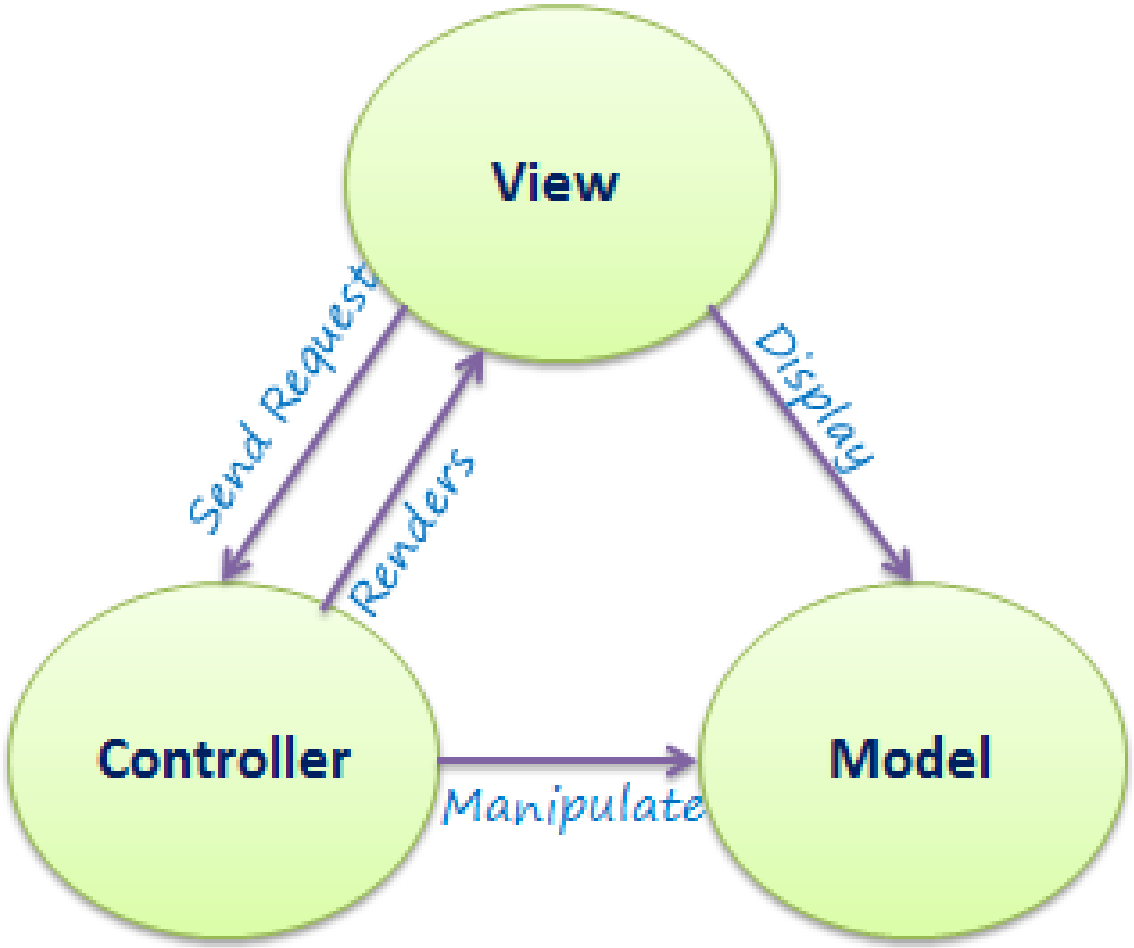
- **View**: View is a user interface. View display data using model to the user and also enables them to modify the data.

  View is a User Interface.

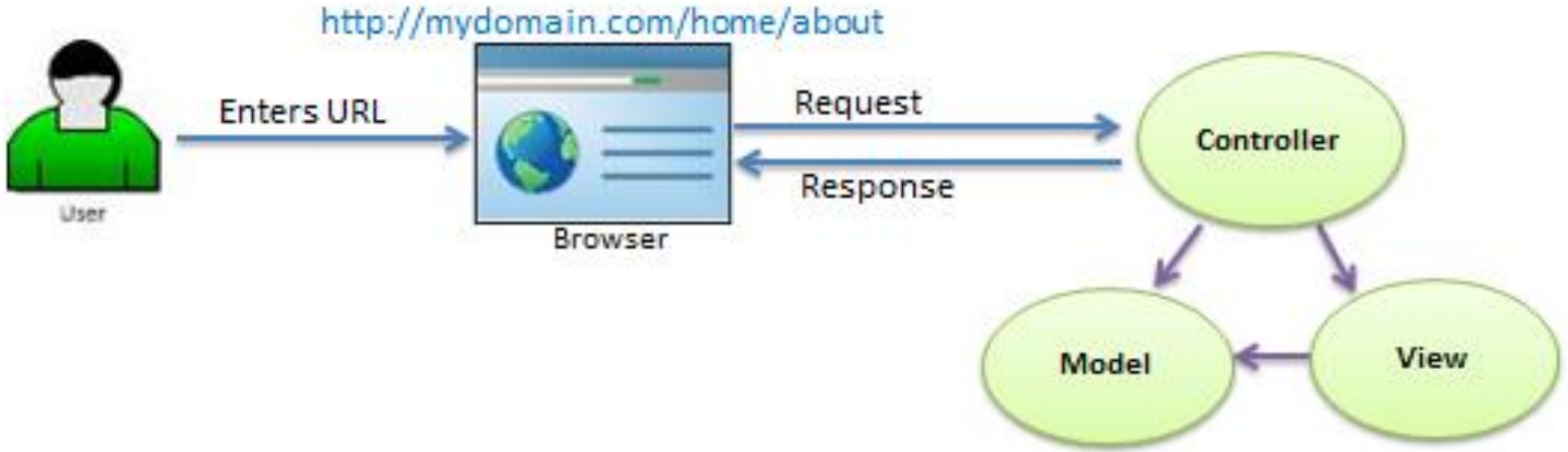
- **Controller**: Controller handles the user request. Typically, user interacts with View, which intern raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.

  Controller is a request handler.

MVC Architecture



MVC Architecture



Request/Response in MVC Architecture

MVC Architecture

Points to Remember

- MVC stands for Model, View and Controller.
- Model is responsible for maintaining application data and business logic.
- View is a user interface of the application, which displays the data.
- Controller handles user's requests and renders appropriate View with Model data.
- 28-Aug-2014: MVC 5.2 - VS 2013 .NET 4.5

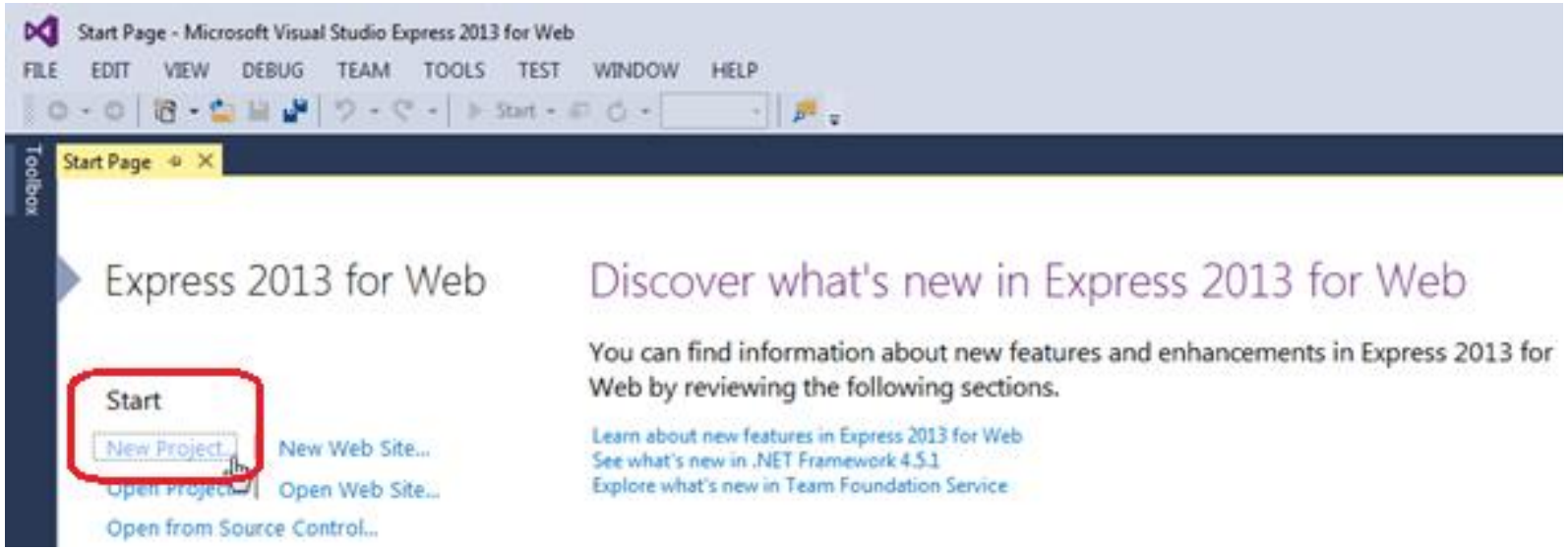
Create First ASP.NET MVC Application

Development Environment setup:

- You can develop ASP.NET MVC application with appropriate version of Visual Studio and .NET framework, as you have seen in the previous section of version history.
- Here, we will use MVC v5.2, Visual Studio 2013 for Web Express edition and .NET framework 4.5 to create MVC application.

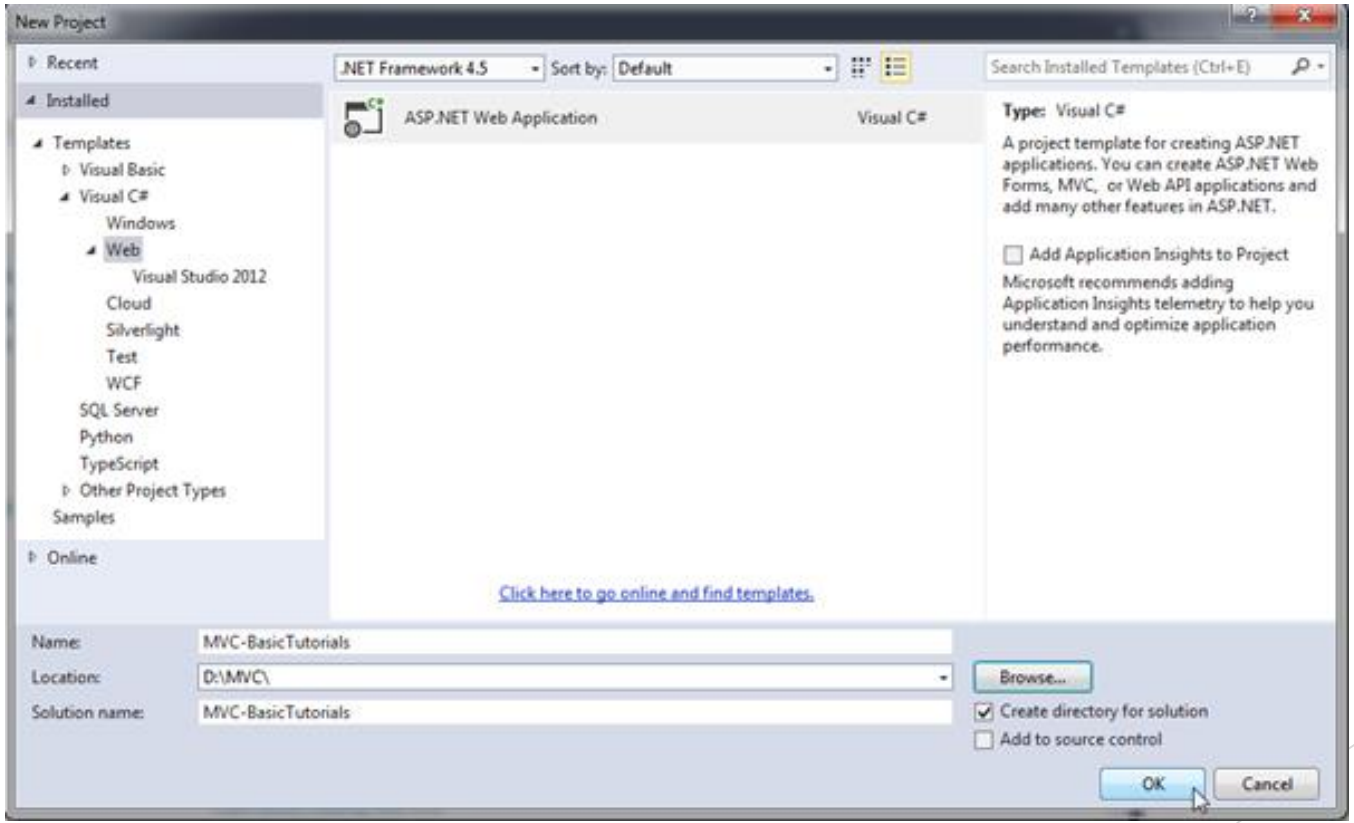
Create First ASP.NET MVC Application (1)

First of all, open a Visual Studio 2013 for Web and click on a **New Project** link in the **Start** page. Alternatively, you can also select **File menu**-> **New Project.**



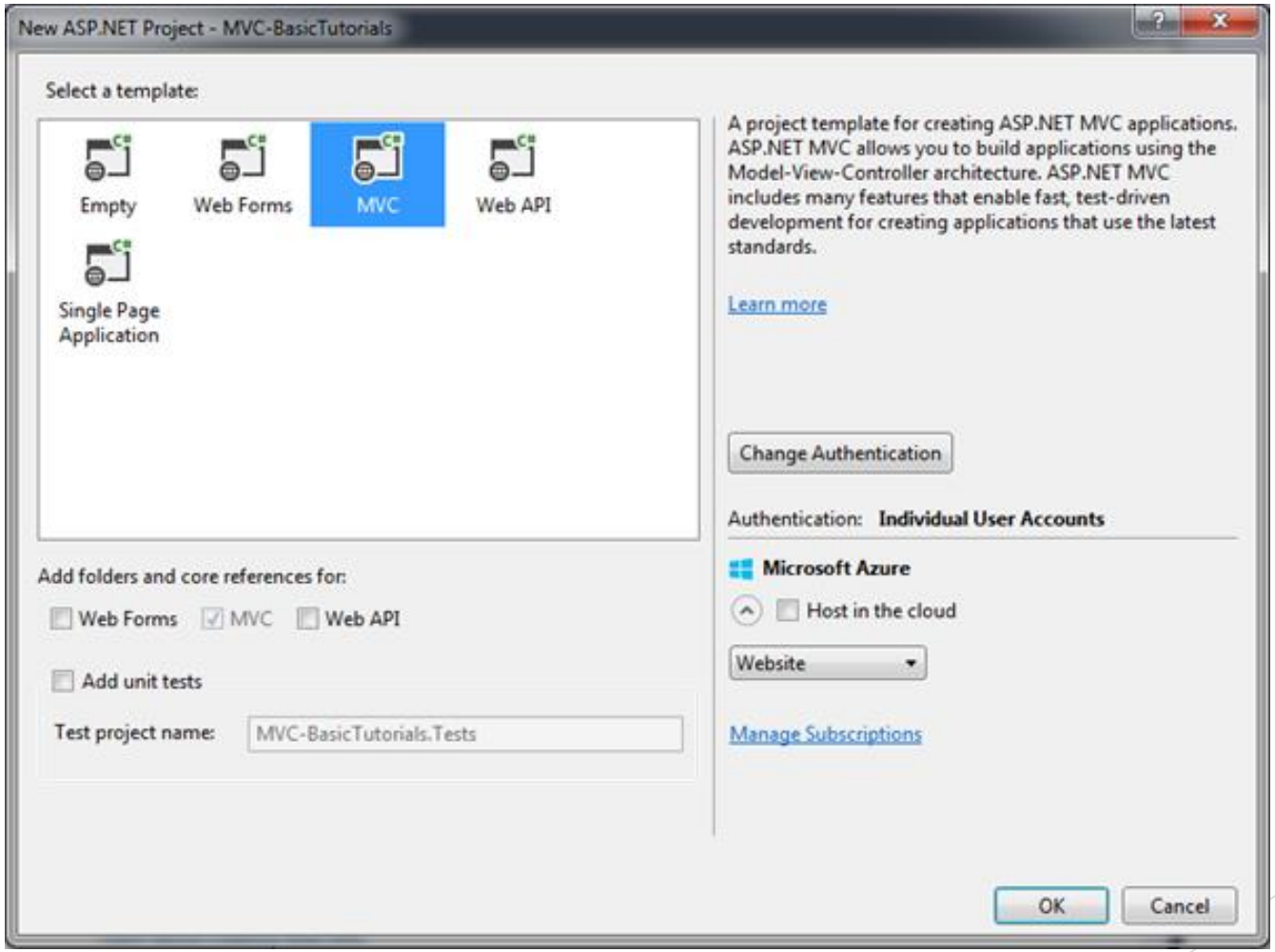
Create First ASP.NET MVC Application (2)

From the **New Project** dialog as shown below, expand Visual C# node and select **Web** in the left pane, and then select **ASP.NET Web Application** in the middle pane. Name your project MVC-BasicTutorials. (You can give any appropriate name for your application). Also, you can change the location of the MVC application by clicking on **Browse..** button. Finally, click **OK.**



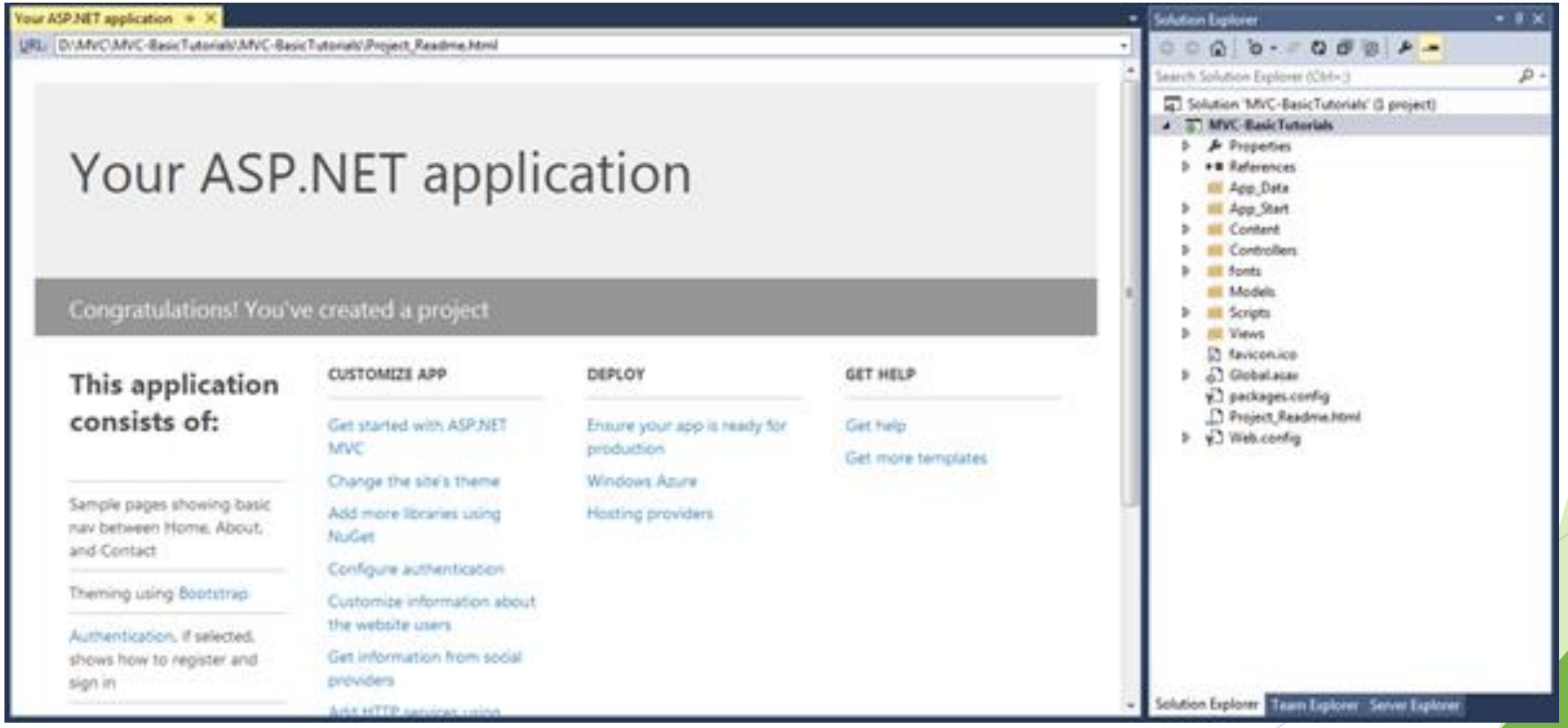
Create First ASP.NET MVC Application (3)

- From the **New ASP.NET Project** dialog, select MVC as shown below.



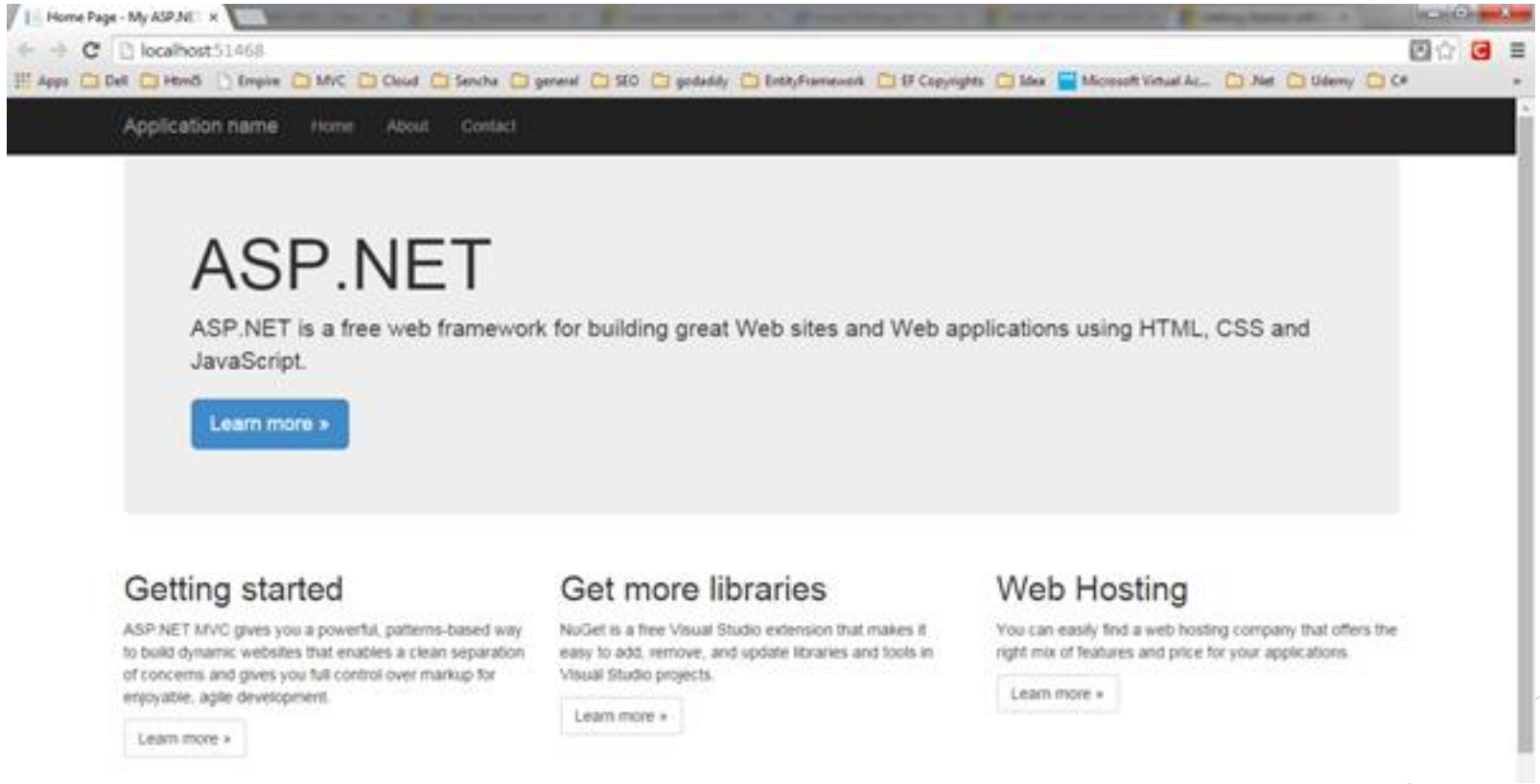
Create First ASP.NET MVC Application (4)

- Wait for some time till Visual Studio creates a simple MVC project using default template as shown below.



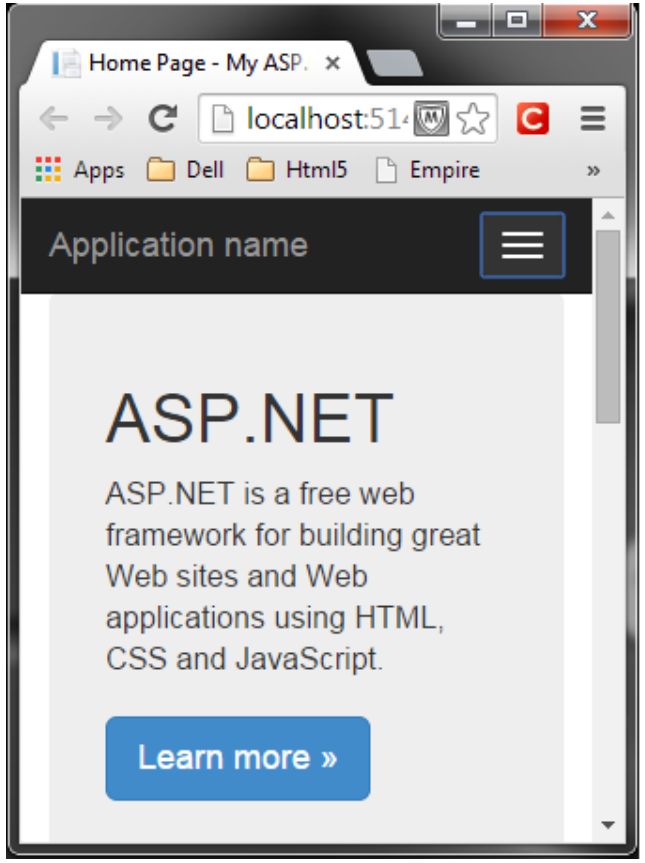
Create First ASP.NET MVC Application (5)

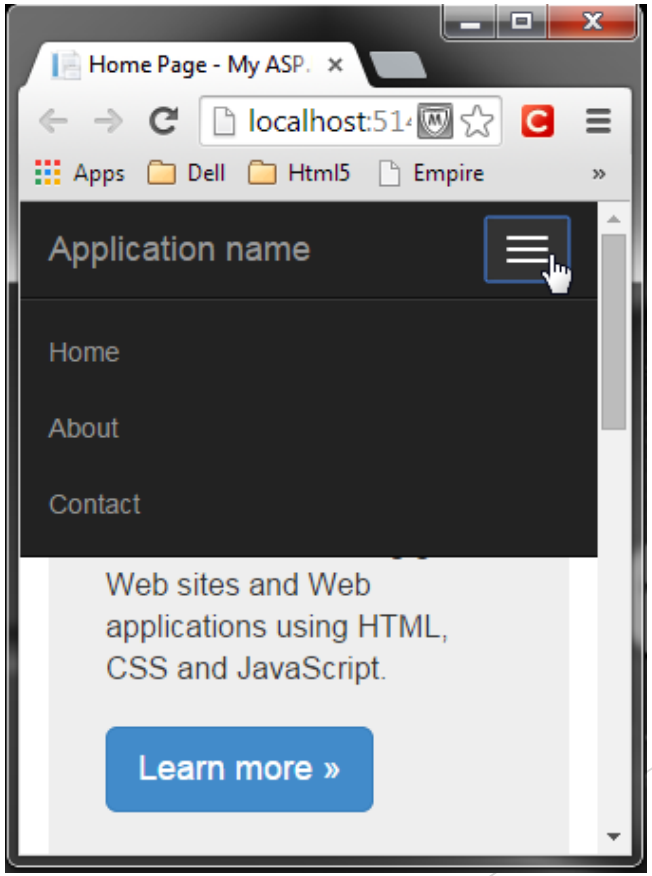
- Now, press F5 to run the project in debug mode or Ctrl + F5 to run the project without debugging. It will open home page in the browser as shown below.



Create First ASP.NET MVC Application (6)

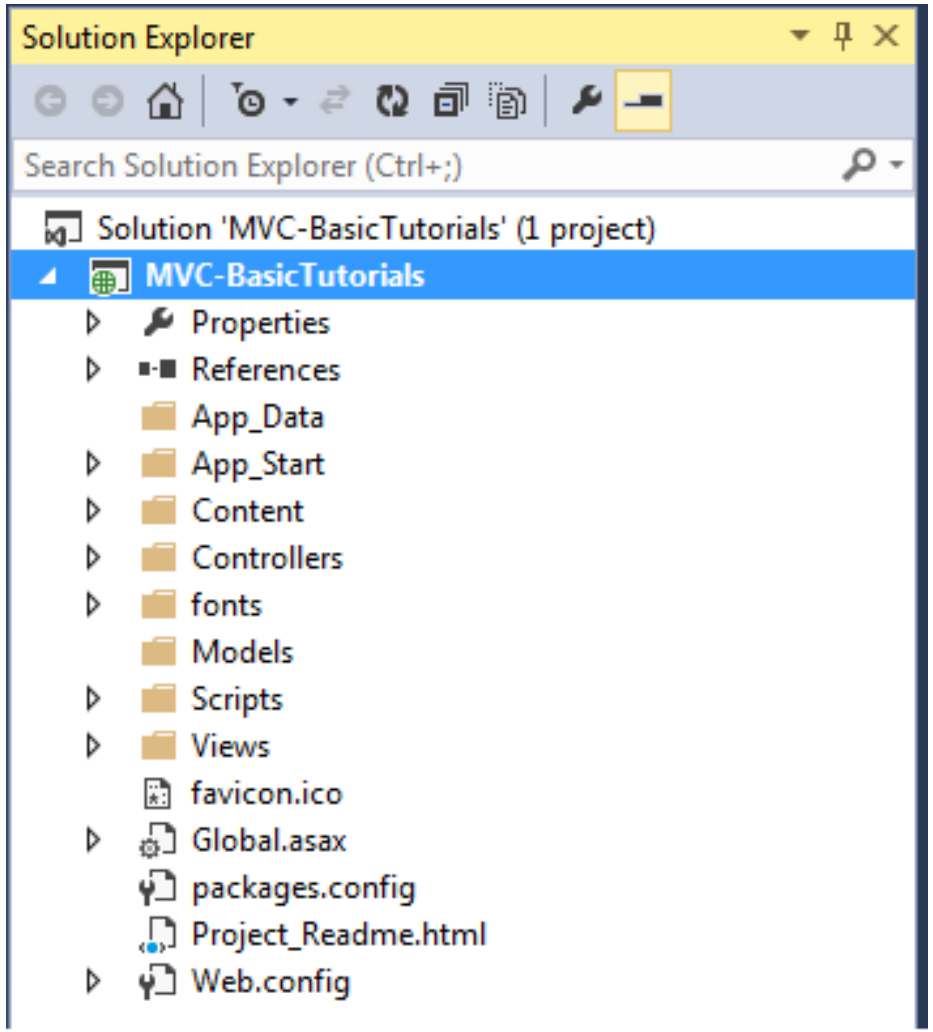
MVC 5 project includes JavaScript and CSS files of bootstrap 3.0 by default. So you can create responsive web pages. This responsive UI will change its look and feel based on the screen size of the different devices. For example, top menu bar will be changed in the mobile devices as shown below.





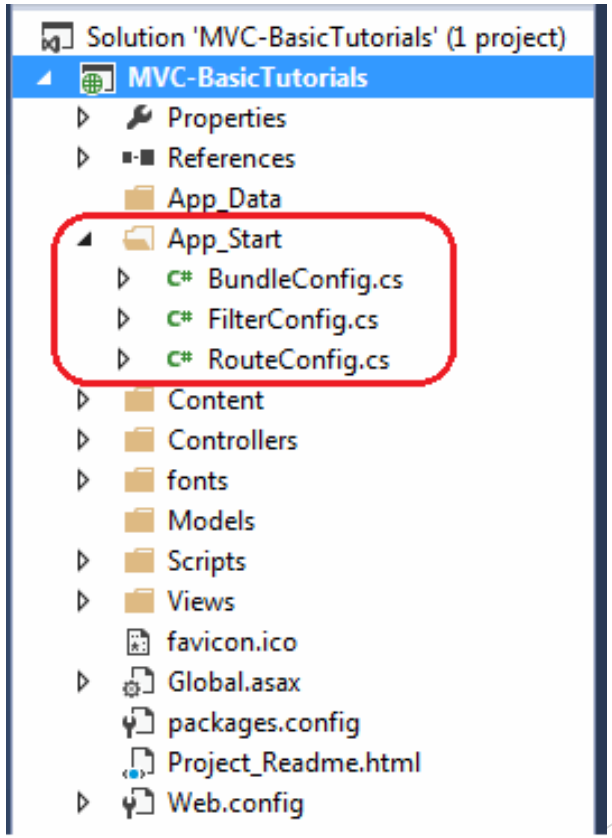
ASP.NET MVC Folder Structure

Visual Studio creates the following folder structure for MVC application by default.



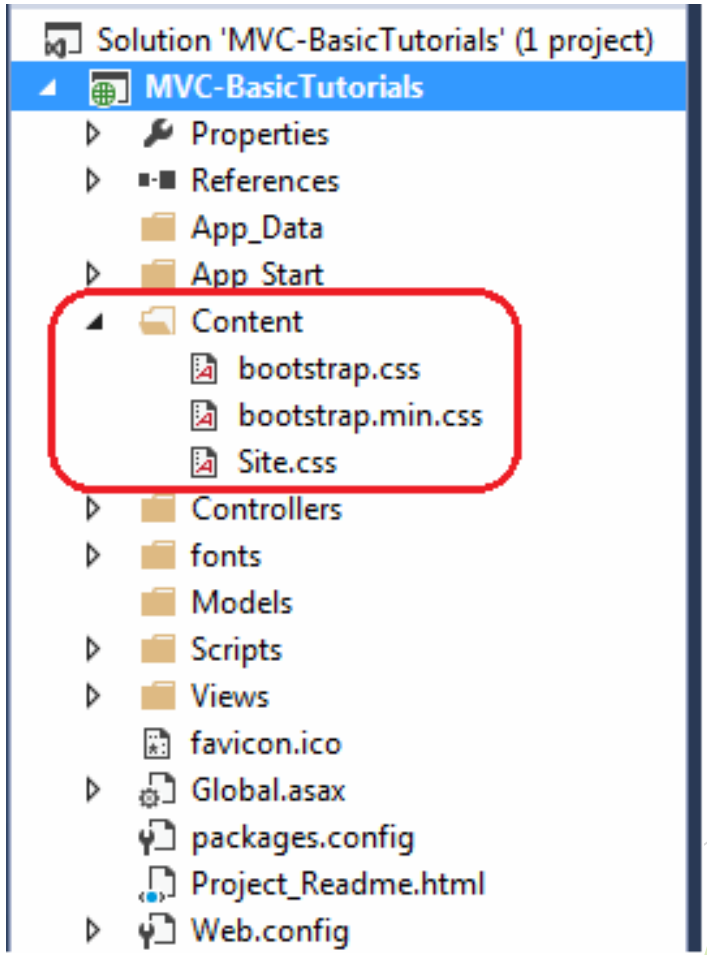
ASP.NET MVC Folder Structure

- App_Data: App_Data folder can contain application data files like LocalDB, .mdf files, xml files and other data related files.
- App_Start: App_Start folder can contain class files which will be executed when the application starts. Typically, these would be config files like AuthConfig.cs, BundleConfig.cs, FilterConfig.cs, RouteConfig.cs etc. MVC 5 includes BundleConfig.cs, FilterConfig.cs and RouteConfig.cs by default.



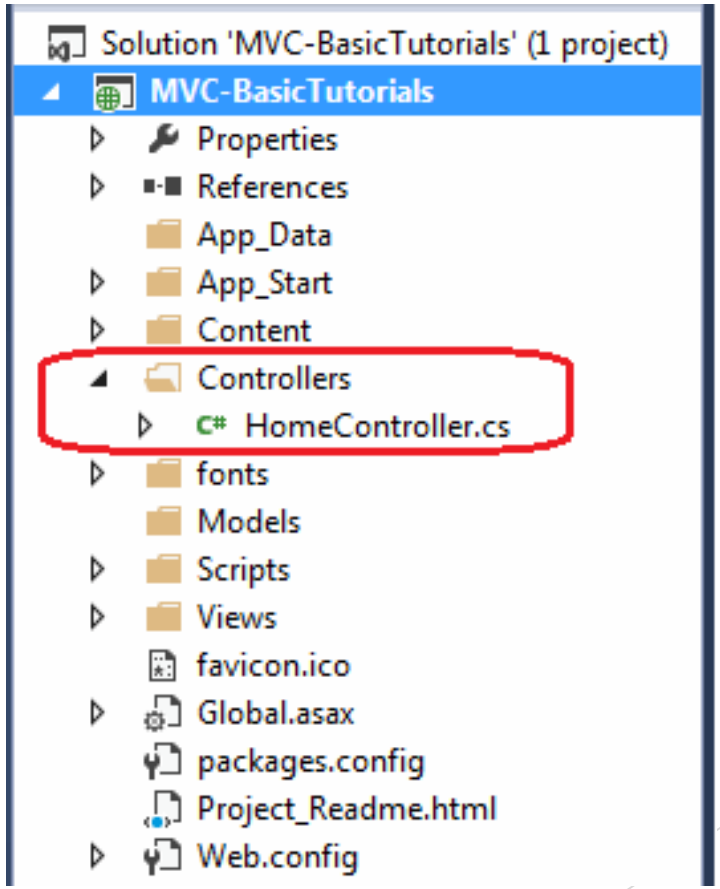
ASP.NET MVC Folder Structure

- Content: Content folder contains static files like css files, images and icons files. MVC 5 application includes bootstrap.css, bootstrap.min.css and Site.css by default.



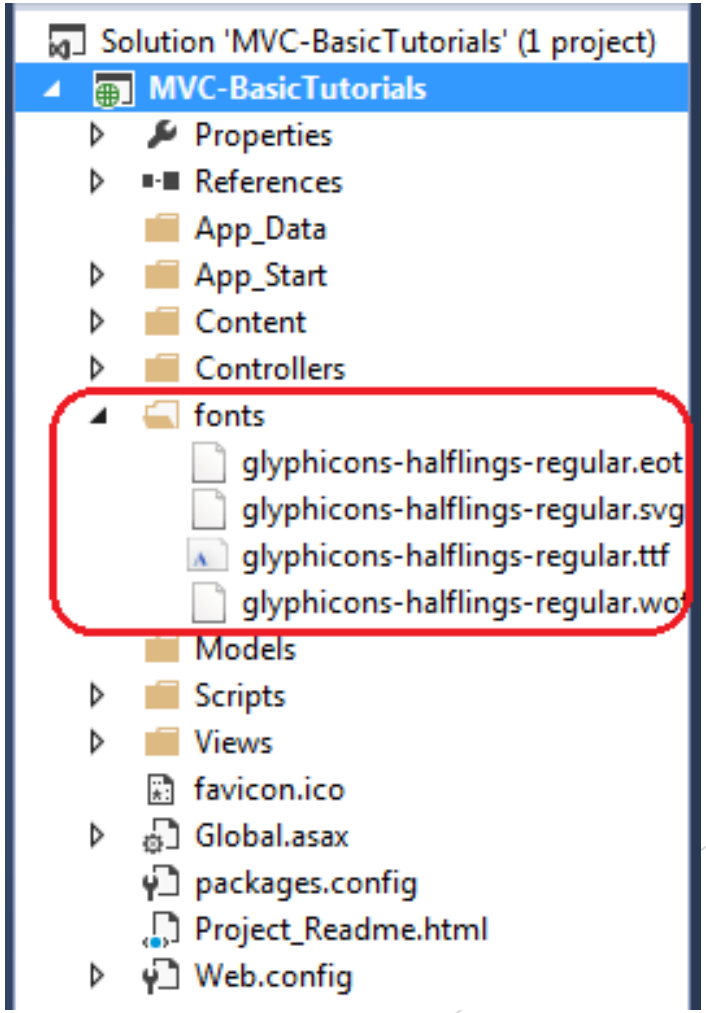
ASP.NET MVC Folder Structure

- Controllers: Controllers folder contains class files for the controllers. Controllers handles users' request and returns a response. MVC requires the name of all controller files to end with "Controller".



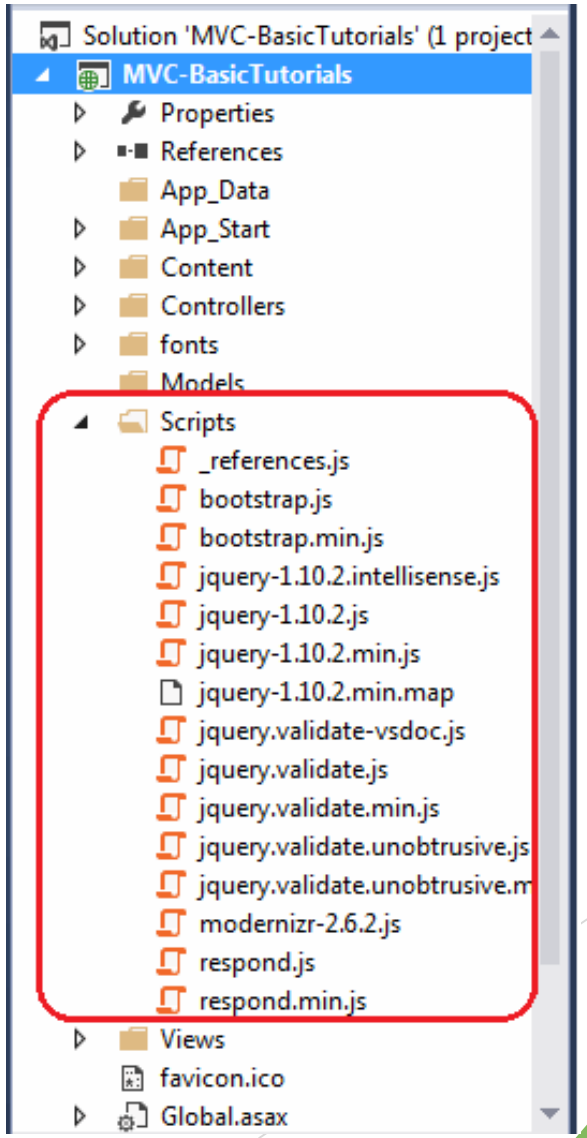
ASP.NET MVC Folder Structure

- fonts: Fonts folder contains custom font files for your application.



ASP.NET MVC Folder Structure

- Models: Models folder contains model class files. Typically model class includes public properties, which will be used by application to hold and manipulate application data.
- Scripts: Scripts folder contains JavaScript or VBScript files for the application. MVC 5 includes javascript files for bootstrap, jquery 1.10 and modernizer by default.

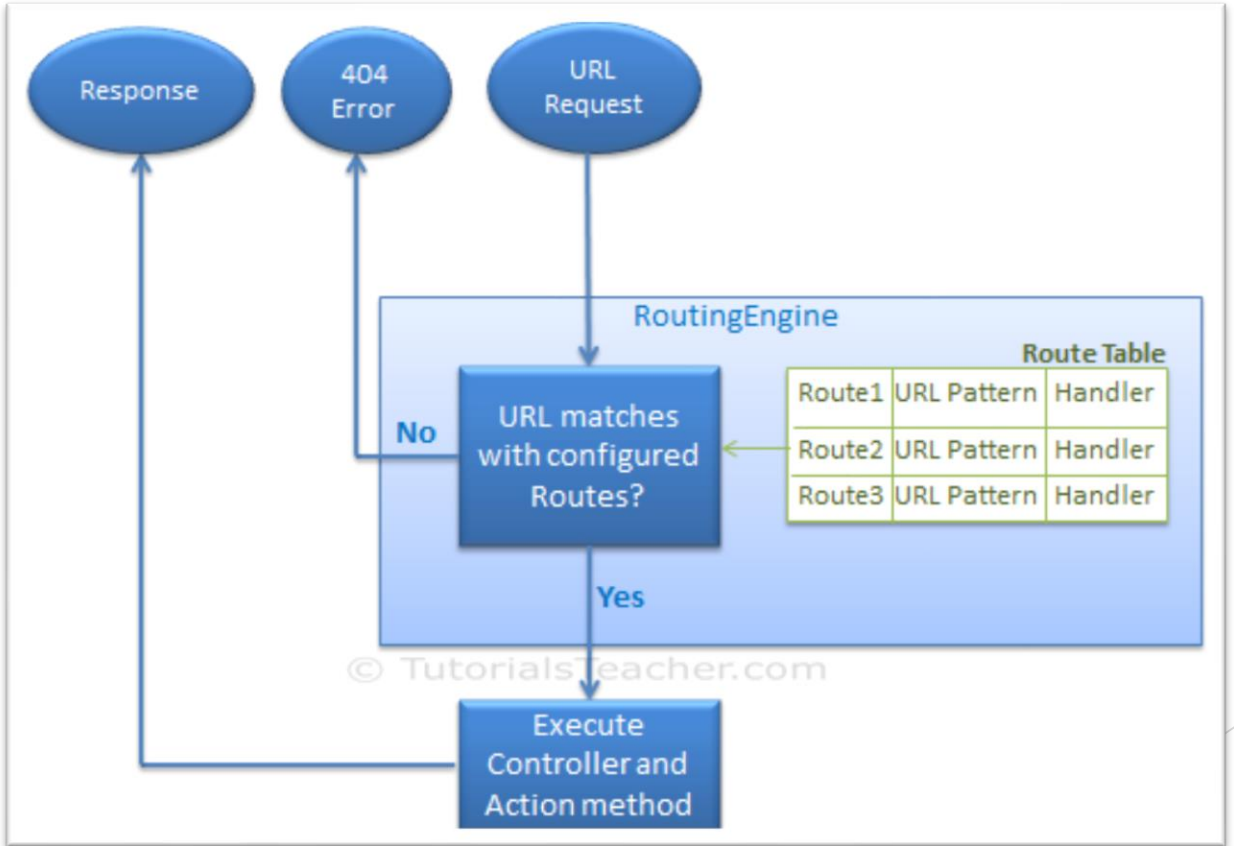


Routing in MVC

- In the ASP.NET Web Forms application, every URL must match with a specific .aspx file. For example, a URL http://domain/studentsinfo.aspx must match with the file studentsinfo.aspx that contains code and markup for rendering a response to the browser.
- ASP.NET introduced **Routing** to eliminate needs of mapping each URL with a physical file. **Routing enable us to define URL pattern that maps to the request handler**. This request handler can be a file or class. In ASP.NET Webform application, request handler is .aspx file and **in MVC, it is Controller class and Action method**.
- For example, http://domain/students can be mapped to http://domain/studentsinfo.aspx in ASP.NET Webforms and the same URL can be mapped to Student Controller and Index action method in MVC.

Routing in MVC

- Route defines the URL pattern and handler information. All the configured routes of an application stored in RouteTable and will be used by Routing engine to determine appropriate handler class or file for an incoming request.
- The following figure illustrates the Routing process.



Routing in MVC

Configure Route:

- Every MVC application must configure (register) at least one route, which is configured by MVC framework by default. You can register a route in **RouteConfig** class, which is in RouteConfig.cs under **App_Start** folder. The following figure illustrates how to configure a Route in the RouteConfig class .

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

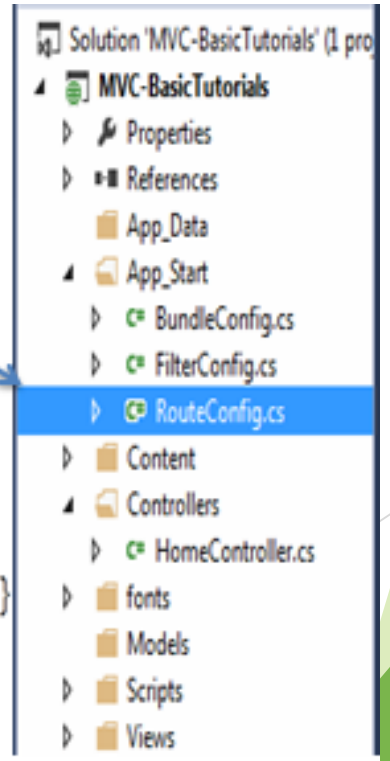
Route to ignore

Route name

URL Pattern

Defaults for Route

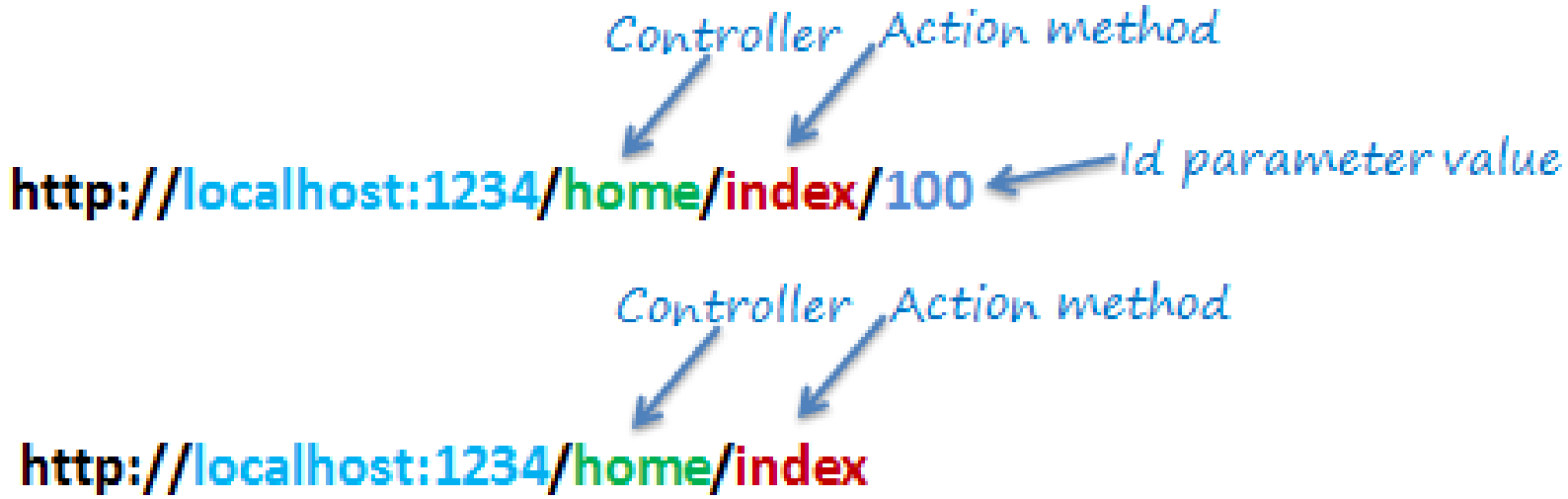
RouteConfig.cs



Routing in MVC

URL Pattern:

- The URL pattern is considered only after domain name part in the URL. For example, the URL pattern *"{controller}/{action}/{id}"* would look like localhost:1234/{controller}/{action}/{id}. Anything after "localhost:1234/" would be considered as controller name. The same way, anything after controller name would be considered as action name and then value of id parameter.



Routing in MVC

URL	Controller	Action	Id
http://localhost/home	?	?	?
http://localhost/home/index/123	HomeController	Index	123
http://localhost/home/about	?	?	?
http://localhost/home/contact	?	?	?
http://localhost/student	?	?	?
http://localhost/student/edit/123	?	?	?

Routing in MVC

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Student",
            url: "students/{id}",
            defaults: new { controller = "Student", action = "Index"}
        );

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

Multiple Routes:

You can also configure a custom route using MapRoute extension method. You need to provide at least two parameters in MapRoute, route name and url pattern. The Defaults parameter is optional.

You can register multiple custom routes with different names. Consider the following example where we register "Student" route.

Routing in MVC

The following table shows how different URLs will be mapped to Student route:

URL	Controller	Action	Id
http://localhost/student/123	StudentController	Index	123
http://localhost/student/index/123	StudentController	Index	123
http://localhost/student?Id=123	StudentController	Index	123

Routing in MVC

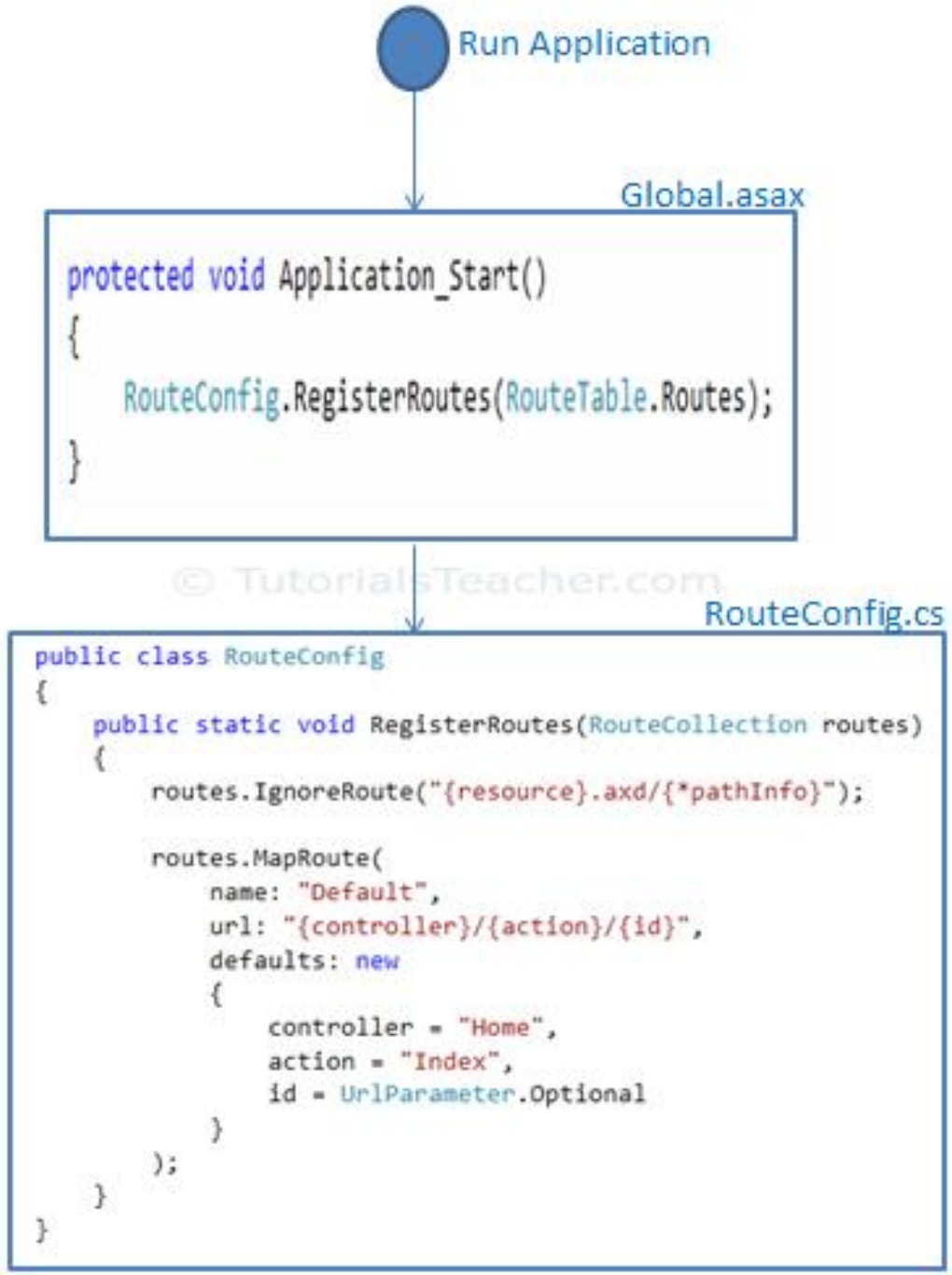
Register Routes:

- Now, after configuring all the routes in RouteConfig class, you need to register it in the Application_Start() event in the Global.asax. So that it includes all your routes into RouteTable

```
public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
```

Routing in MVC

- The figure illustrates Route registration process.



Routing in MVC

Points to Remember

- Routing plays important role in MVC framework. Routing maps URL to physical file or class (controller class in MVC).
- Route contains URL pattern and handler information. URL pattern starts after domain name.
- Routes can be configured in RouteConfig class. Multiple custom routes can also be configured.
- Route constraints applies restrictions on the value of parameters.
- Route must be registered in Application_Start event in Global.ascx.cs file.

Question #1

MVC stands for _____.

1. Model, Vision & Control
2. Model, View & Controller
3. Model, ViewData & Controller
4. Model, Data & Controller

Question #2

Which of following is TRUE?

1. The controller redirects incoming request to model.
2. The controller executes an incoming request.
3. The controller controls the data.
4. The controller render html to view.

Question #3

The model is a _____ .

1.Shape of data.
2.Html content
3.Collection of data
4.Type of data.