RKIT Module – 5

Name: Adhyaru Keyur D.

ASP.NET MVC Tutorials

- ASP.NET is a free web framework for building websites and web applications on .NET Framework using HTML, CSS, and JavaScript.
- ASP.NET MVC 5 is a web framework based on Model-View-Controller (MVC) architecture. Developers can build dynamic web applications using ASP.NET MVC framework that enables a clean separation of concerns, fast development, and TDD friendly.

ASP.NET MVC Architecture

- The MVC architectural pattern has existed for a long time in software engineering.
- All most all the languages use MVC with slight variation, but conceptually it remains the same.
- MVC stands for Model, View, and Controller. MVC separates an application into three components - Model, View, and Controller.

Model: Model represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database.

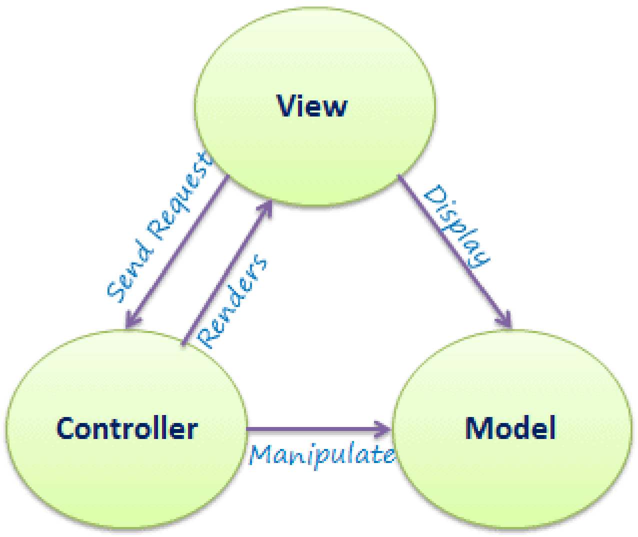
View: View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS, and some special syntax (Razor syntax) that makes it easy to communicate with the model and the controller.

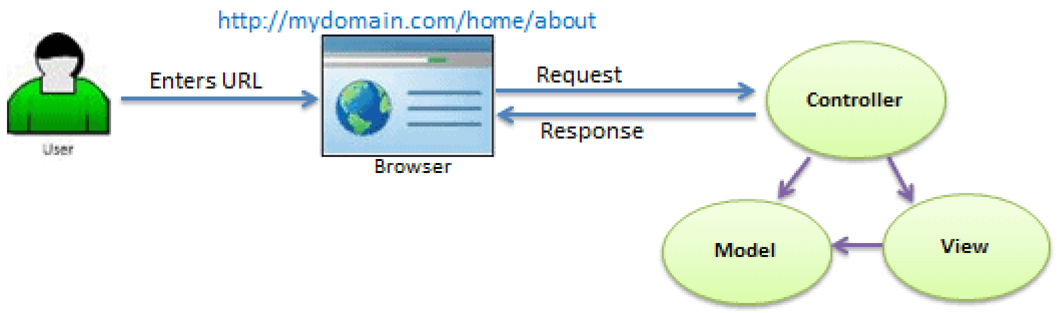
Controller: The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.

Model represents the data.

View is the User Interface.

Controller is the request handler.





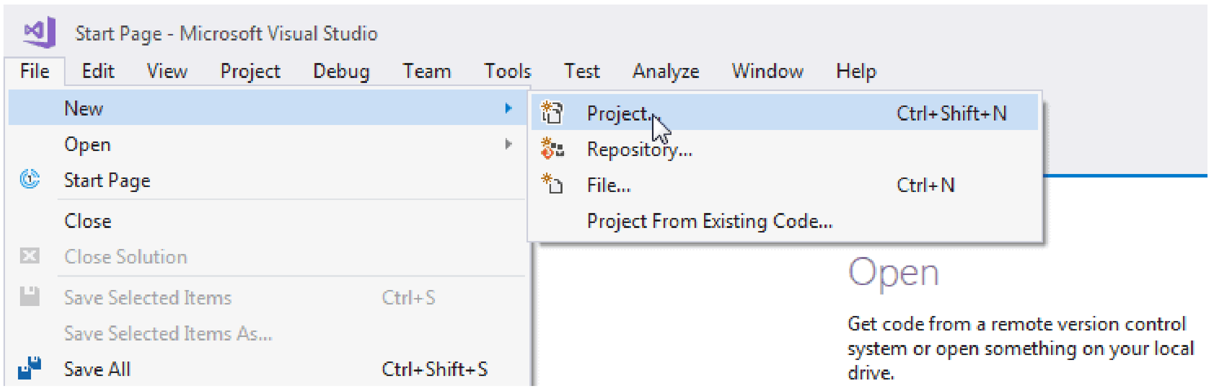
- As per the above figure, when a user enters a URL in the browser, it goes to the webserver and routed to a controller.
- A controller executes related view and models for that request and create the response and sends it back to the browser.

Points to Remember

1. MVC stands for Model, View and Controller.
2. Model represents the data
3. View is the User Interface.
4. Controller is the request handler.

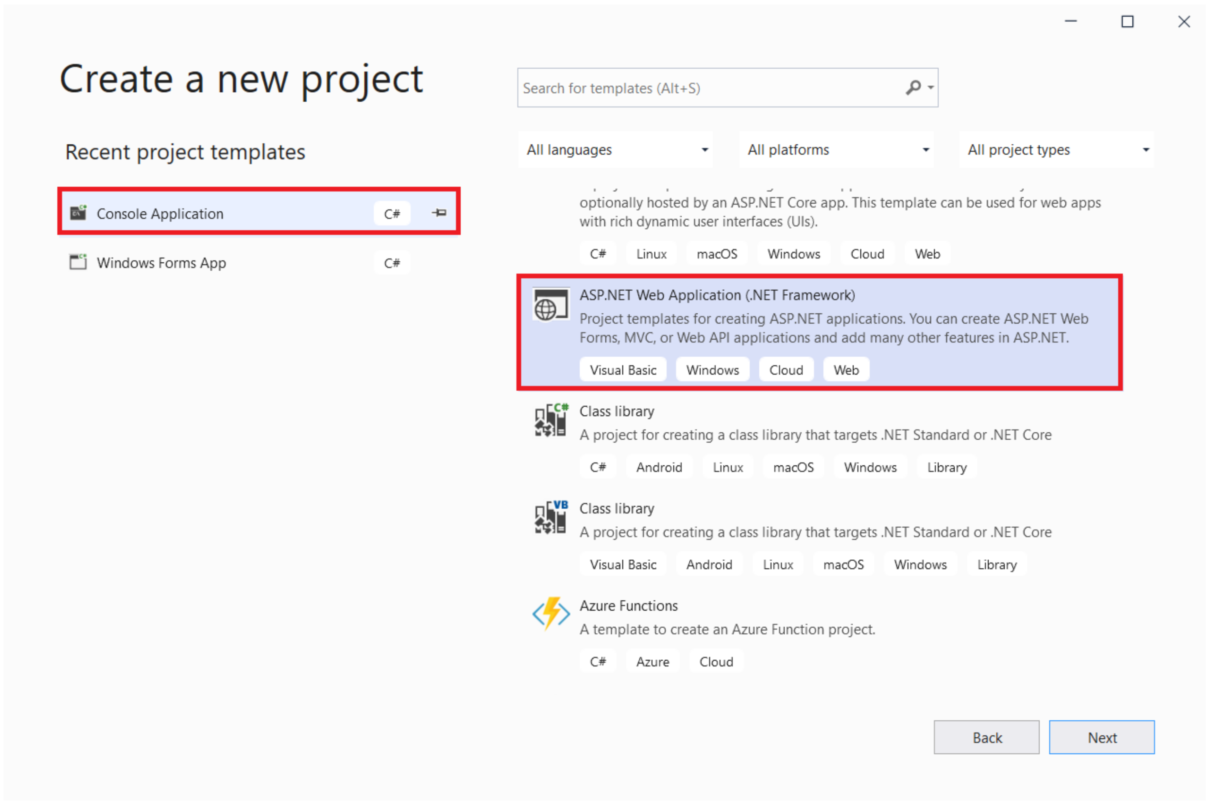
Create ASP.NET MVC Application

Step 1: Open Visual Studio 2017 and select **File menu -> New -> Project**, as shown below.


Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Project... Ctrl+Shift+N
Repository...
File... Ctrl+N
Project From Existing Code...
Open

Get code from a remote version control system or open something on your local drive.

Step 2: Now Select **Console Application** and Choose **Asp.net web application** and click on the **next** button.



optionally hosted by an ASP.NET Core app. This template can be used for web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Visual Basic Windows Cloud Web

Class library

A project for creating a class library that targets .NET Standard or .NET Core

C# Android Linux macOS Windows Library

Class library

A project for creating a class library that targets .NET Standard or .NET Core

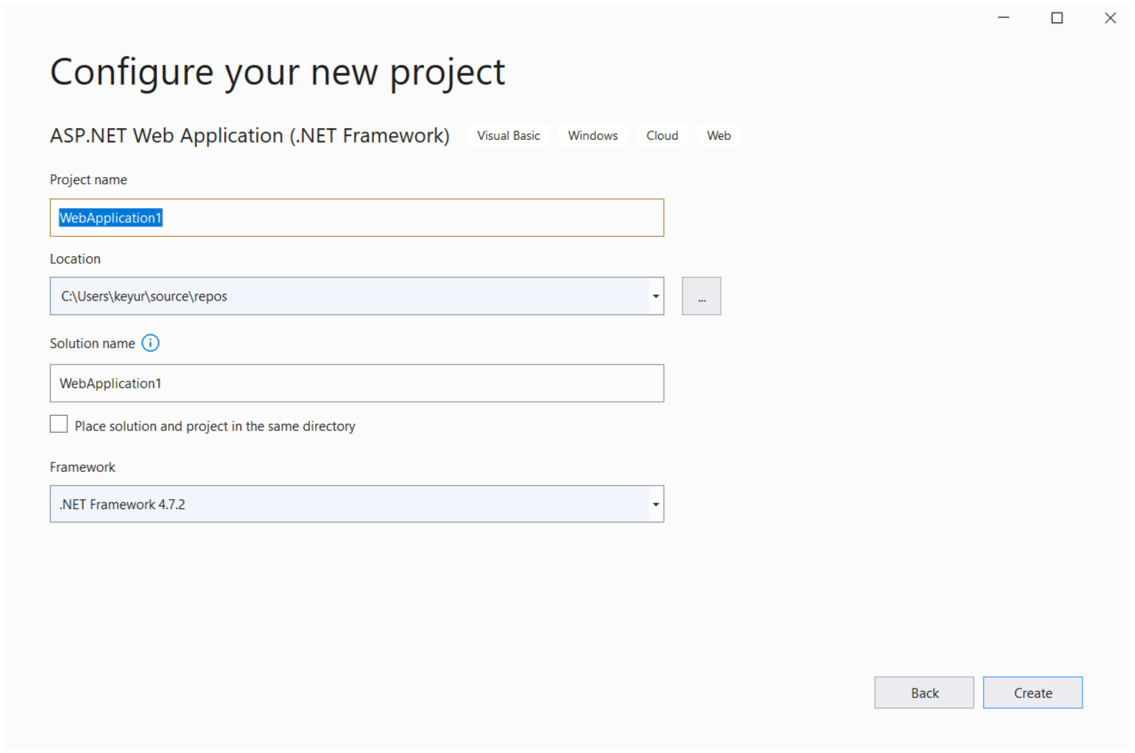
Visual Basic Android Linux macOS Windows Library

Azure Functions

A template to create an Azure Function project.

C# Azure Cloud

Step 3: Now Write the Project name, choose Project Solution, Write Solution name and the Framework.



Configure your new project

ASP.NET Web Application (.NET Framework) Visual Basic Windows Cloud Web

Project name: WebApplication1

Location: C:\Users\keyur\source\repos

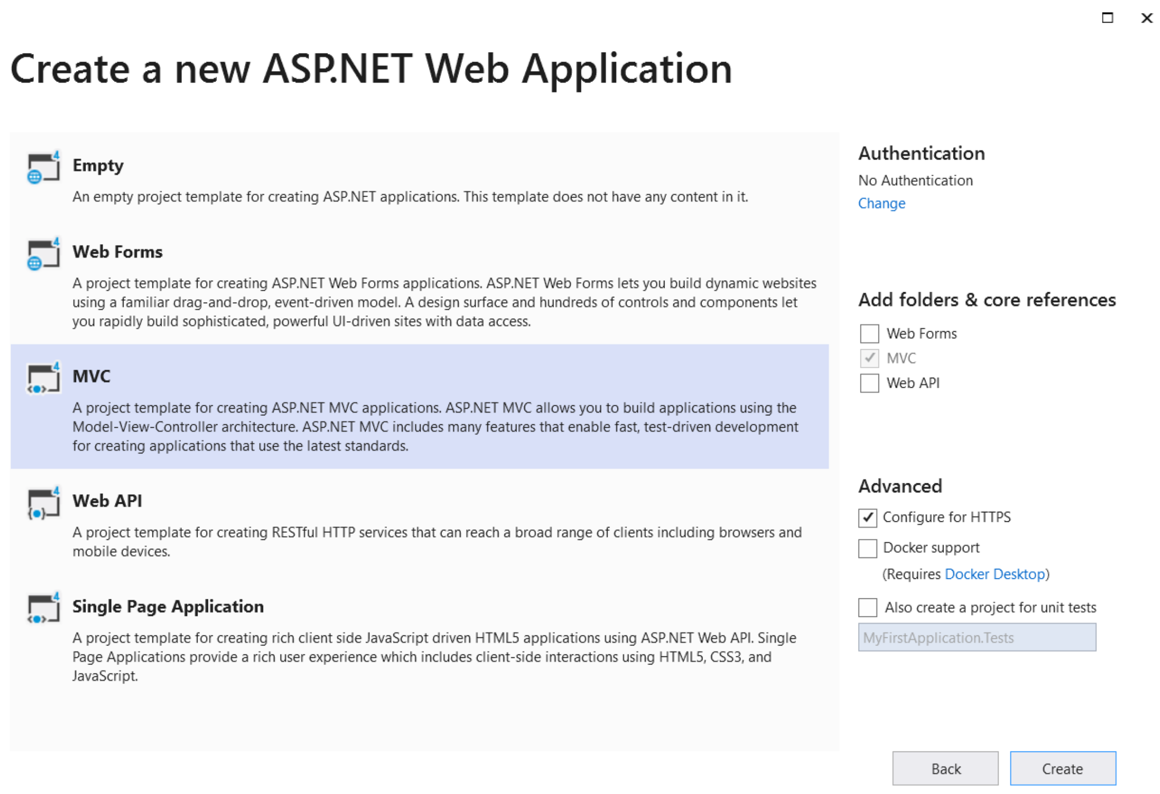
Solution name: WebApplication1

☐ Place solution and project in the same directory

Framework: .NET Framework 4.7.2

Step 4: Now We have to Choose MVC Application. And Click on the **Create** Button

You can also change the authentication by clicking on **Change** button. You can select appropriate authentication mode for your application.



Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

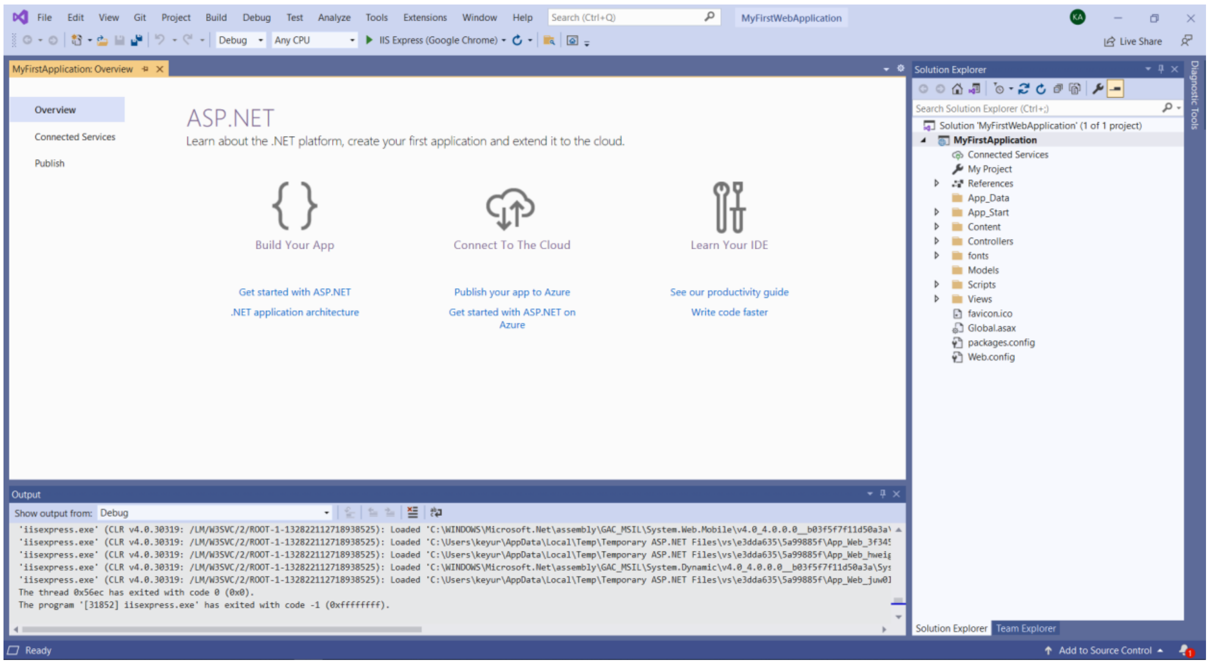
Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication
No Authentication
Change

Add folders & core references
☐ Web Forms
✓ MVC
☐ Web API

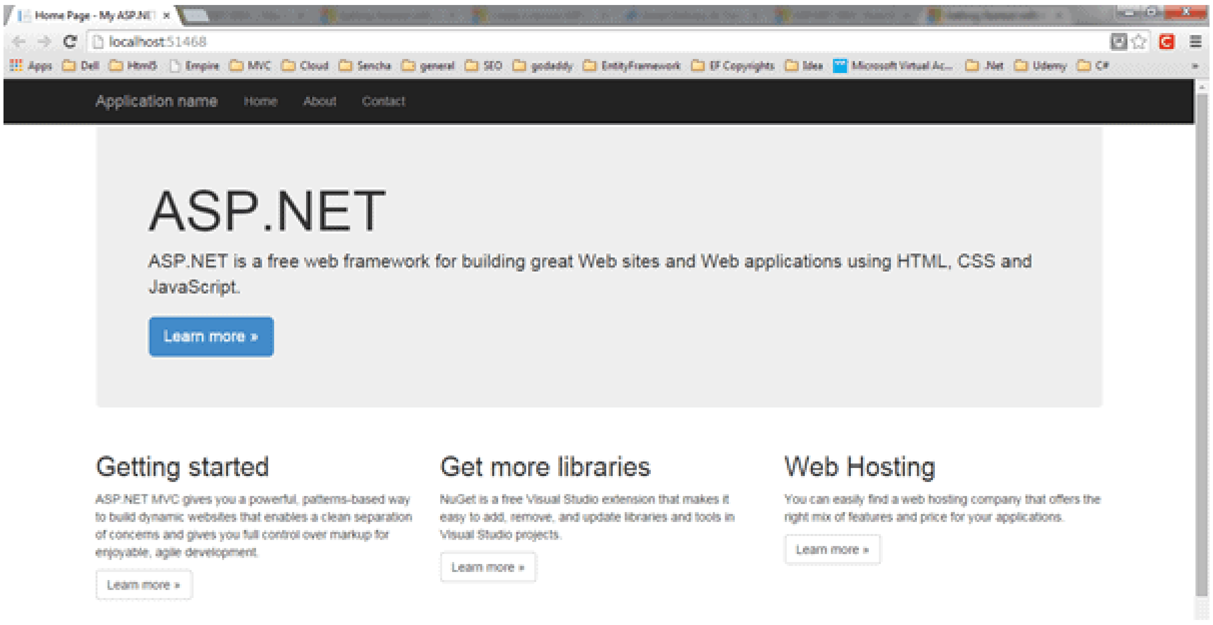
Advanced
✓ Configure for HTTPS
☐ Docker support
(Requires Docker Desktop)
☐ Also create a project for unit tests
MyFirstApplication.Tests

So Here Our Project is Created.



```
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132822112718938525): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Web.Mobile\v4.0_4.0.0.0__b03f5f7f11d50a3a\
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132822112718938525): Loaded 'C:\Users\keyur\AppData\Local\Temp\Temporary ASP.NET Files\vs\e3dda635\5a99885f\App_Web_3f345
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132822112718938525): Loaded 'C:\Users\keyur\AppData\Local\Temp\Temporary ASP.NET Files\vs\e3dda635\5a99885f\App_Web_hweig
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132822112718938525): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Dynamic\v4.0_4.0.0.0__b03f5f7f11d50a3a\Sys
'iisexpress.exe' (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132822112718938525): Loaded 'C:\Users\keyur\AppData\Local\Temp\Temporary ASP.NET Files\vs\e3dda635\5a99885f\App_Web_juw0l
The thread 0x56ec has exited with code 0 (0x0).
The program '[31852] iisexpress.exe' has exited with code -1 (0xffffffff).
```

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



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

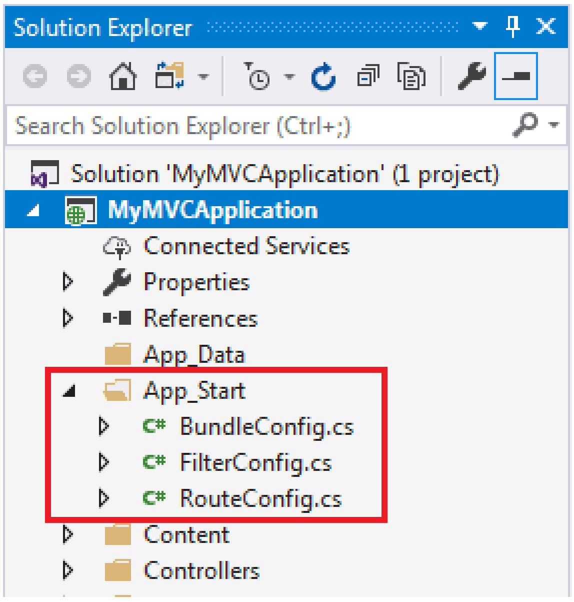
Folder Structure

App_Data

- The App_Data folder can contain application data files like LocalDB, .mdf files, XML files, and other data related files. IIS will never serve files from App_Data folder.

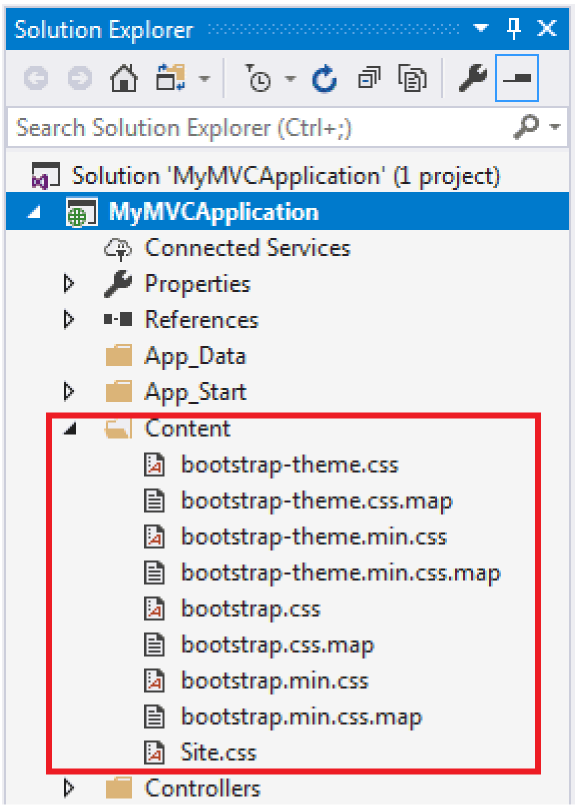
App_Start

- The App_Start folder can contain class files that 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. We will see the significance of these files later.



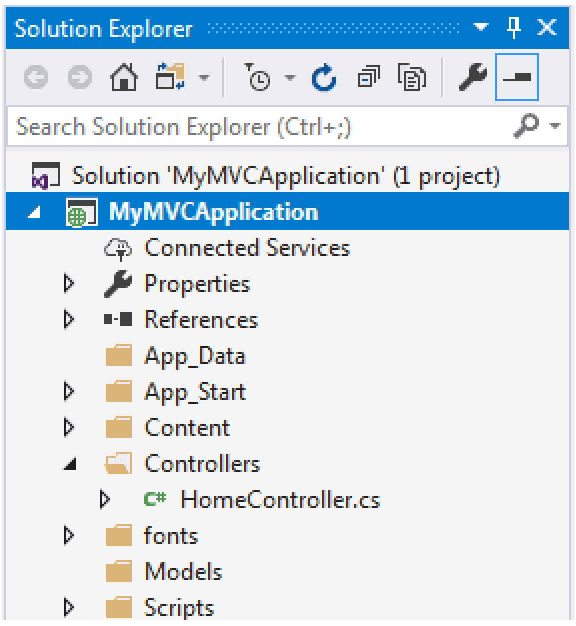
Content

- The 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.



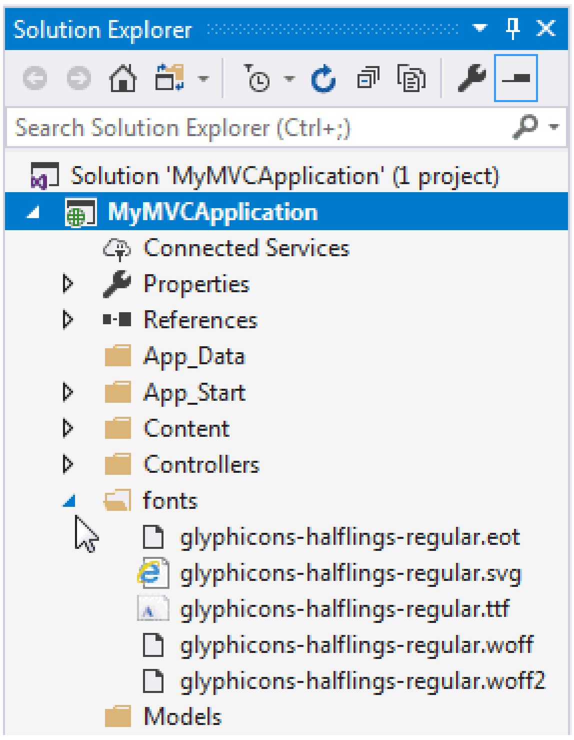
Controllers

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



fonts

- The Fonts folder contains custom font files for your application.

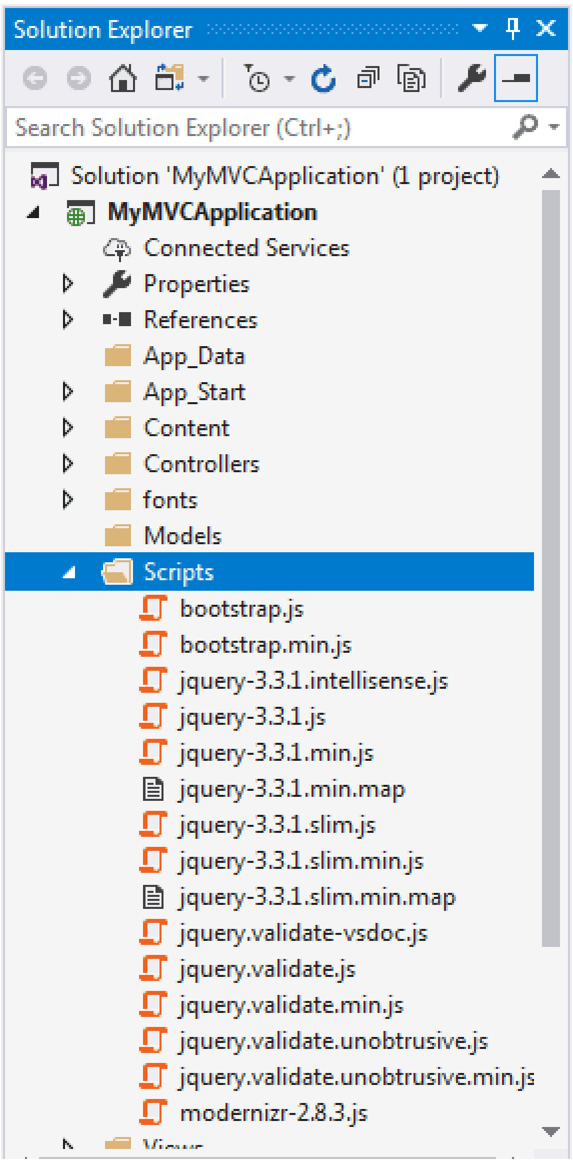

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MyMVCApplication' (1 project)
MyMVCApplication
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
fonts
glyphicons-halflings-regular.eot
glyphicons-halflings-regular.svg
glyphicons-halflings-regular.ttf
glyphicons-halflings-regular.woff
glyphicons-halflings-regular.woff2
Models

Models

- The Models folder contains model class files.
- Typically, model class includes public properties, which will be used by the application to hold and manipulate application data.

Scripts

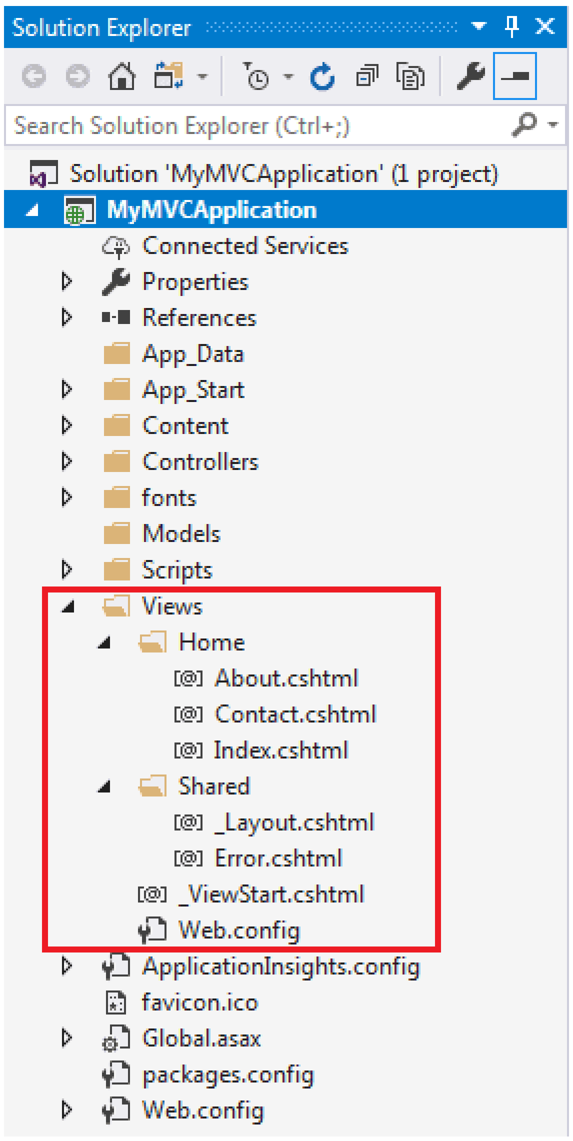
- The Scripts folder contains JavaScript or VBScript files for the application.
- MVC 5 includes JavaScript files for bootstrap, jQuery 1.10, and modernizer by default.



```
Solution 'MyMVCApplication' (1 project)
  MyMVCApplication
    Connected Services
    Properties
    References
    App_Data
    App_Start
    Content
    Controllers
    fonts
    Models
    Scripts
      bootstrap.js
      bootstrap.min.js
      jquery-3.3.1.intellisense.js
      jquery-3.3.1.js
      jquery-3.3.1.min.js
      jquery-3.3.1.min.map
      jquery-3.3.1.slim.js
      jquery-3.3.1.slim.min.js
      jquery-3.3.1.slim.min.map
      jquery.validate-vsdoc.js
      jquery.validate.js
      jquery.validate.min.js
      jquery.validate.unobtrusive.js
      jquery.validate.unobtrusive.min.js
      modernizr-2.8.3.js
```

<u>Views</u>

- The Views folder contains HTML files for the application. Typically view file is a .cshtml file where you write HTML and C# or VB.NET code.
- The Views folder includes a separate folder for each controller. For example, all the .cshtml files, which will be rendered by HomeController will be in View > Home folder.
- The Shared folder under the View folder contains all the views shared among different controllers e.g., layout files.


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'MyMVCApplication' (1 project)
  MyMVCApplication
      Connected Services
      Properties
      References
      App_Data
      App_Start
      Content
      Controllers
      fonts
      Models
      Scripts
      Views
        Home
          About.cshtml
          Contact.cshtml
          Index.cshtml
        Shared
          _Layout.cshtml
          Error.cshtml
        _ViewStart.cshtml
        Web.config
      ApplicationInsights.config
      favicon.ico
      Global.asax
      packages.config
      Web.config
```

- Additionally, MVC project also includes the following configuration files:

Global.asax

- Global.asax file allows you to write code that runs in response to application-level events, such as Application_BeginRequest, application_start, application_error, session_start, session_end, etc.

Packages.config

- Packages.config file is managed by NuGet to track what packages and versions you have installed in the application.

Web.config

- Web.config file contains application-level configurations.

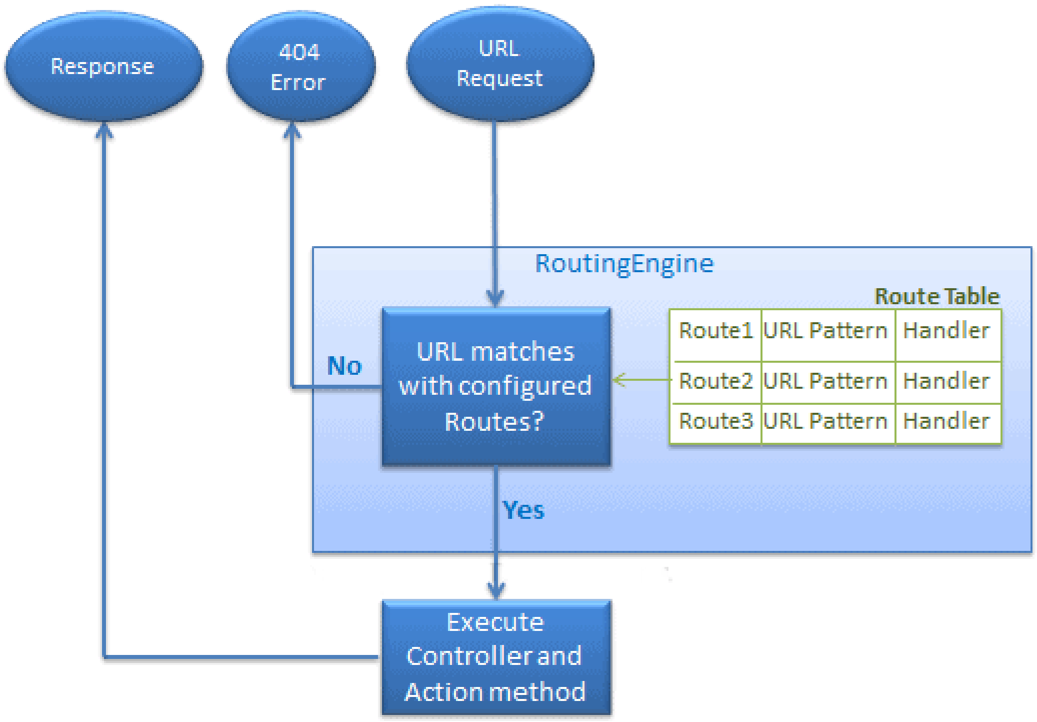
Routing in MVC

- 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.

Note: Routing is not specific to the MVC framework. It can be used with ASP.NET Webform application or MVC application

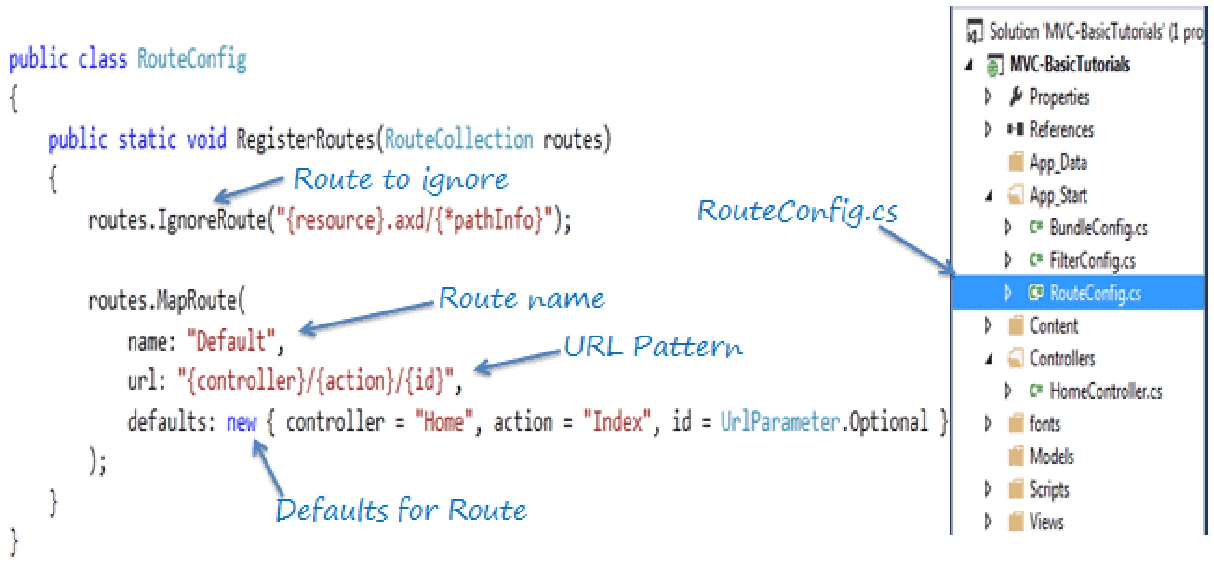
Route

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


Response
404 Error
URL Request
RoutingEngine
Route Table
URL matches with configured Routes?
No
Yes
Route1 URL Pattern Handler
Route2 URL Pattern Handler
Route3 URL Pattern Handler
Execute Controller and Action method

Configure a Route

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



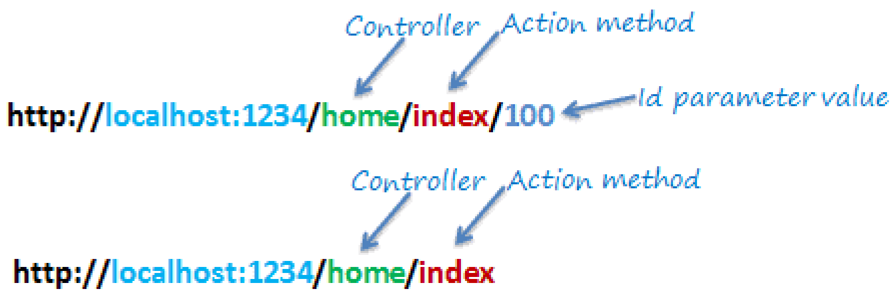
```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

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

- As you can see in the above figure, the route is configured using the **MapRoute()** extension method of **RouteCollection**, where name is "**Default**", url pattern is **"{controller}/{action}/{id}"** and defaults parameter for controller, action method and id parameter.
- Defaults specify which controller, action method, or value of id parameter should be used if they do not exist in the incoming request URL.
- In the same way, you can configure other routes using the MapRoute() method of the RouteCollection class.
- This RouteCollection is actually a property of the RouteTable class.

URL Pattern

- The URL pattern is considered only after the 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 a controller name.
- The same way, anything after the controller name would be considered as action name and then the value of id parameter.



Controller
Action method
http://localhost:1234/home/index/100
Id parameter value
Controller
Action method
http://localhost:1234/home/index

- If the URL doesn't contain anything after the domain name, then the default controller and action method will handle the request.
- For example, http://localhost:1234 would be handled by the HomeController and the Index() method as configured in the default parameter.

Multiple Routes

- You can also configure a custom route using the 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.

Example

```
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 }
        );
    }
}
```

Route Constraints

- You can also apply restrictions on the value of the parameter by configuring route constraints.
- For example, the following route applies a limitation on the id parameter that the id's value must be numeric.

Example

```
routes.MapRoute(
        name: "Student",
        url: "student/{id}/{name}/{standardId}",
        defaults: new { controller = "Student", action = "Index",
id = UrlParameter.Optional, name = UrlParameter.Optional, standardId
= UrlParameter.Optional },
        constraints: new { id = @"\d+" }
    );
```

- So, if you give non-numeric value for id parameter, then that request will be handled by another route or, if there are no matching routes, then "**The resource could not be found**" error will be thrown.

Register Routes

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

Example

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

Points to Remember:

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

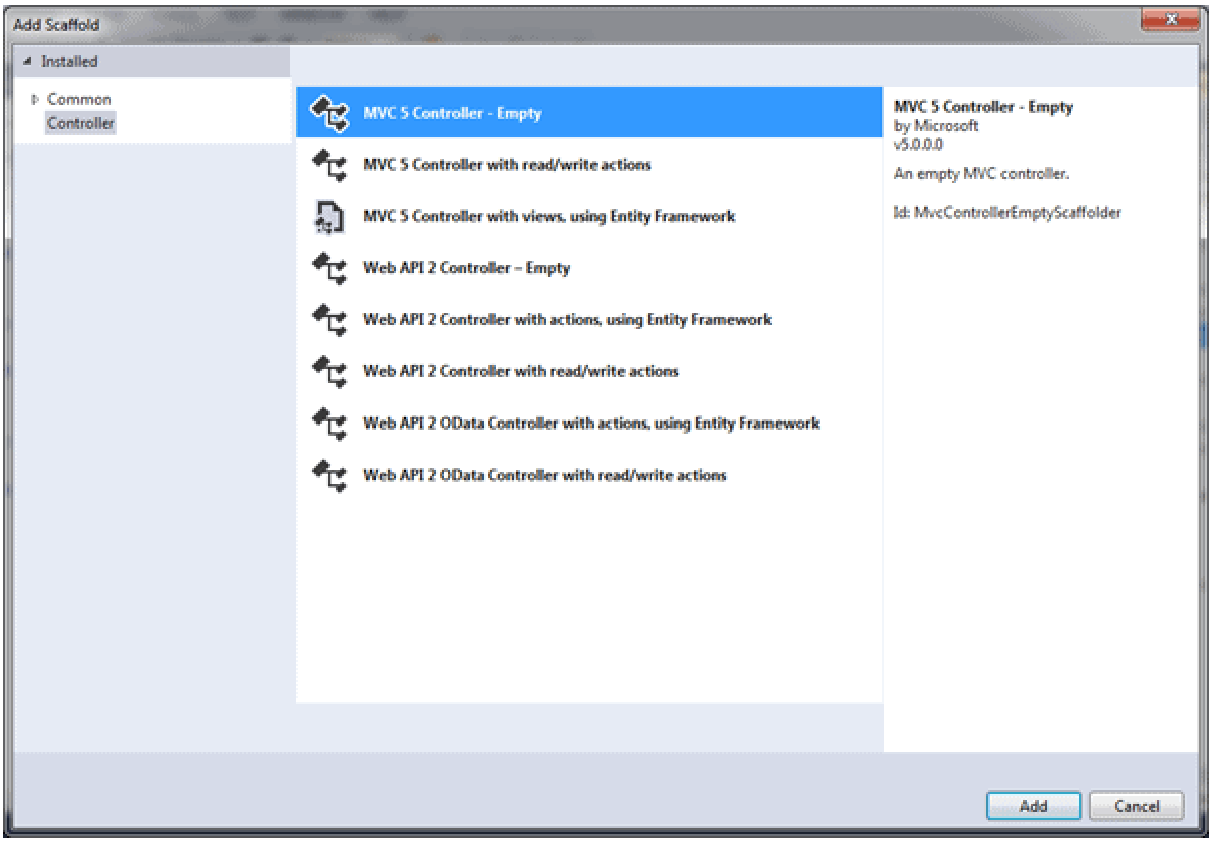
Controllers in ASP.NET MVC

- The Controller in MVC architecture handles any incoming URL request.
- The Controller is a class, derived from the base class System.Web.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, the home page controller name must be HomeController, and for the student page, it must be the StudentController.
- Also, every controller class must be located in the Controller folder of the MVC folder structure.

Adding a New Controller

Step 1: In the Visual Studio, right click on the Controller folder -> select Add -> click on Controller..

This opens Add Scaffold dialog, as shown below.



MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData Controller with actions, using Entity Framework
Web API 2 OData Controller with read/write actions

MVC 5 Controller - Empty
by Microsoft
v5.0.0.0
An empty MVC controller.
Id: MvcControllerEmptyScaffolder

Note: Scaffolding is an automatic code generation framework for ASP.NET web applications. Scaffolding reduces the time taken to develop a controller, view, etc. in the MVC framework. You can develop a customized scaffolding template using T4 templates as per your architecture and coding standards.

Step 2: Select the Empty template and Click on the **Add** Button.

Step 3: Given you **Controller Name** ended with the **Controller** Keywords

Points to Remember:

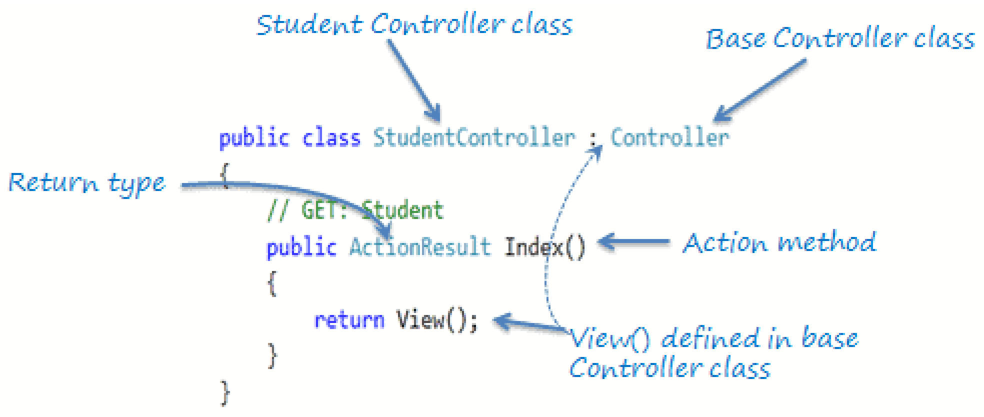
1. The Controller handles incoming URL requests. MVC routing sends requests to the appropriate controller and action method based on URL and configured Routes.
2. All the public methods in the Controller class are called Action methods.
3. The Controller class must be derived from System.Web.Mvc.Controller class.
4. The Controller class name must end with "Controller".
5. A new controller can be created using different scaffolding templates. You can create a custom scaffolding template also.

Action method

All the public methods of the Controller class are called Action methods.

They are like any other normal methods with the following restrictions:

1. Action method must be public. It cannot be private or protected
2. Action method cannot be overloaded
3. Action method cannot be a static method.



```
public class StudentController : Controller
{
    // GET: Student
    public ActionResult Index()
    {
        return View();
    }
}
```

- As you can see in the above figure, the Index() method is public, and it returns the ActionResult using the View() method.
- The View() method is defined in the Controller base class, which returns the appropriate ActionResult.

Default Action Method

- Every controller can have a default action method as per the configured route in the RouteConfig class.
- By default, the Index() method is a default action method for any controller, as per configured default root, as shown below.

Example

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

However, you can change the default action name as per your requirement in the **RouteConfig** class.

ActionResult

- MVC framework includes various Result classes, which can be returned from an action method.
- The result classes represent different types of responses, such as HTML, file, string, JSON, javascript, etc. The following table lists all the result classes available in ASP.NET MVC.

Result Class	Description
ViewResult	Represents HTML and markup.
EmptyResult	Represents No response.
ContentResult	Represents string literal.
FileContentResult/ FilePathResult/ FileStreamResult	Represents the content of a file.
JavaScriptResult	Represent a JavaScript script.
JsonResult	Represent JSON that can be used in AJAX.
RedirectResult	Represents a redirection to a new URL.
RedirectToRouteResult	Represent another action of same or other controller.
PartialViewResult	Returns HTML from Partial view.
HttpUnauthorizedResult	Returns HTTP 403 status.

- The ActionResult class is a base class of all the above result classes, so it can be the return type of action method that returns any result listed above.
- However, you can specify the appropriate result class as a return type of action method.
- The Index() method of the StudentController in the above figure uses the View() method to return a ViewResult (which is derived from the ActionResult class).
- The base Controller class includes the View() method along with other methods that return a particular type of result, as shown in the below table.

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	Represents a JavaScript script.	JavaScript()
JsonResult	Represents JSON that can be used in AJAX.	Json()
RedirectResult	Represents a redirection to a new URL.	Redirect()
RedirectToRouteResult	Represents redirection to another route.	RedirectToRoute()
PartialViewResult	Represents the partial view result.	PartialView()
HttpUnauthorizedResult	Represents HTTP 403 response.	

Action Method Parameters

- Every action method can have input parameters as normal methods.
- It can be primitive data type or complex type parameters, as shown below.

Example

```
[HttpPost]
public ActionResult Edit(Student std)
{
    // update student to the database

    return RedirectToAction("Index");
}

[HttpDelete]
public ActionResult Delete(int id)
{
    // delete student from the database whose id matches with
specified id

    return RedirectToAction("Index");
}
```

Points to Remember:

1. All the public methods in the Controller class are called Action methods.
2. The Action method has the following restrictions.
   - Action method must be public. It cannot be private or protected.
   - Action method cannot be overloaded.
   - Action method cannot be a static method.
3. ActionResult is a base class of all the result type which returns from Action method.
4. The base Controller class contains methods that returns appropriate result type e.g. View(), Content(), File(), JavaScript() etc.
5. The Action method can include Nullable type parameters.

Action Selectors

Action selector is the attribute that can be applied to the action methods.

It helps the routing engine to select the correct action method to handle a particular request.

1. ActionName
2. NonAction
3. ActionVerbs

Action Name:

The ActionName attribute allows us to specify a different action name than the method name, as shown below.

Example

```
public class StudentController : Controller
{
    public StudentController()
    {
    }

    [ActionName("Find")]
    public ActionResult GetById(int id)
    {
        // get student from the database
        return View();
    }
}
```

- In the above example, we have applied ActioName("find") attribute to the GetById() action method. So now, the action method name is Find instead of the GetById.
- So now, it will be invoked on http://localhost/student/find/1 request instead of http://localhost/student/getbyid/1 request.

NonAction:

- Use the NonAction attribute when you want public method in a controller but do not want to treat it as an action method.
- In the following example, the Index() method is an action method, but the GetStudent() is not an action method.
-

Example

```
public class StudentController : Controller
{
    public string Index()
    {
            return    "This    is    Index    action    method    of
StudentController";
    }

    [NonAction]
    public Student GetStudent(int id)
    {
        return    studentList.Where(s    =>    s.StudentId    ==
id).FirstOrDefault();
    }
}
```

ActionVerbs: HttpGet, HttpPost, HttpPut

- The ActionVerbs selector is to handle different type of Http requests.
- The MVC framework includes HttpGet, HttpPost, HttpPut, HttpDelete, HttpOptions, and HttpPatch action verbs.
- You can apply one or more action verbs to an action method to handle different HTTP requests.
- If you don't apply any action verbs to an action method, then it will handle HttpGet request by default.
- The following table lists the usage of HTTP methods:

Http method	Usage
GET	To retrieve the information from the server. Parameters will be appended in the query string.
POST	To create a new resource.
PUT	To update an existing resource.
HEAD	Identical to GET except that server do not return the message body.
OPTIONS	It represents a request for information about the communication options supported by the web server.
DELETE	To delete an existing resource.
PATCH	To full or partial update the resource.

Example

```
public class StudentController : Controller
{
    public ActionResult Index() // handles GET requests by default
    {
        return View();
    }

    [HttpPost]
    public ActionResult PostAction() // handles POST requests by default
    {
        return View("Index");
    }


    [HttpPut]
    public ActionResult PutAction() // handles PUT requests by default
    {
        return View("Index");
    }

    [HttpDelete]
    public ActionResult DeleteAction() // handles DELETE requests by
default
    {
        return View("Index");
    }

    [HttpHead]
    public ActionResult HeadAction() // handles HEAD requests by default
    {
        return View("Index");
    }

    [HttpOptions]
    public ActionResult OptionsAction() // handles OPTION requests by
default
    {
        return View("Index");
    }

    [HttpPatch]
    public ActionResult PatchAction() // handles PATCH requests by default
    {
        return View("Index");
    }
}
```

You can also apply multiple action verbs using the AcceptVerbs attribute, as shown below.

Example

```
[AcceptVerbs(HttpVerbs.Post | HttpVerbs.Get)]
public ActionResult GetAndPostAction()
{
    return RedirectToAction("Index");
}
```

Model in ASP.NET MVC

- The model classes represent domain-specific data and business logic in the MVC application.
- It represents the shape of the data as public properties and business logic as methods.
- All the Model classes must be created in the **Model** folder.

Step to Create Model Class

Step 1: Right click on the **Model** Folder.

Step 2: Select **New** and Click on the **class**.

Step 3: Now in this New Dialog. Select Class and give the class name. Click **Add** Button.

- We want this model class to store id, name, and age of the students.
- So, we will have to add public properties for Id, Name, and Age, as shown below.

Example

```
public class Student
{
    public int StudentId { get; set; }
    public string StudentName { get; set;  }
    public int Age { get; set;  }
}
```

The model class can be used in the view to populate the data, as well as sending data to the controller.

ASP.NET Web API Tutorials

- ASP.NET Web API is a framework for building HTTP services that can be accessed from any client including browsers and mobile devices.
- It is an ideal platform for building RESTful applications on the .NET Framework.

What is Web API?

- In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.
- To put it in simple terms, API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.
- Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol.
- It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc.
- The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc.
- It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view.
- It is like a web service or WCF service but the exception is that it only supports HTTP protocol.

ASP.NET Web API Characteristics

1. ASP.NET Web API is an ideal platform for building RESTful services.
2. ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
3. ASP.NET Web API maps HTTP verbs to method names.
4. ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
5. ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
6. ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

ASP.NET Web API vs WCF

Web API	WCF
Open source and ships with .NET framework.	Ships with .NET framework
Supports only HTTP protocol.	Supports HTTP, TCP, UDP and custom transport protocol.
Maps http verbs to methods	Uses attributes based programming model.
Uses routing and controller concept similar to ASP.NET MVC.	Uses Service, Operation and Data contracts.
Does not support Reliable Messaging and transaction.	Supports Reliable Messaging and Transactions.
Web API can be configured using HttpConfiguration class but not in web.config.	Uses web.config and attributes to configure a service.
Ideal for building RESTful services.	Supports RESTful services but with limitations.

Web API Controllers

- Web API Controller is similar to ASP.NET MVC controller.
- It handles incoming HTTP requests and send response back to the caller.
- Web API controller is a class which can be created under the Controllers folder or any other folder under your project's root folder.
- The name of a controller class must end with "Controller" and it must be derived from System.Web.Http.ApiController class.
- All the public methods of the controller are called action methods.

Example

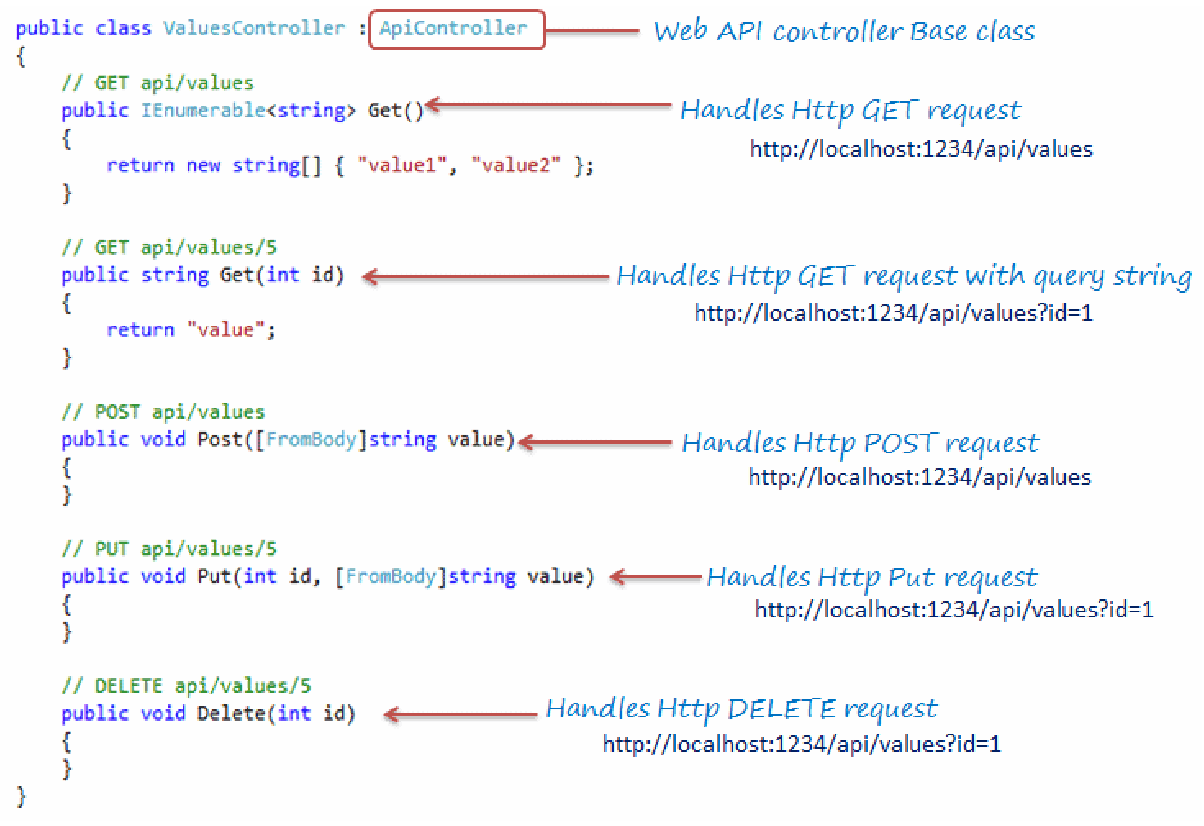
```
namespace MyWebAPI.Controllers
{
    public class ValuesController : ApiController
    {
        // GET: api/values
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET: api/values/5
        public string Get(int id)
        {
            return "value";
        }

        // POST: api/values
        public void Post([FromBody]string value)
        {
        }

        // PUT: api/values/5
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE: api/values/5
        public void Delete(int id)
        {
        }
    }
}
```



```
public class ValuesController : ApiController
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    public string Get(int id)
    {
        return "value";
    }

    // POST api/values
    public void Post([FromBody]string value)
    {
    }

    // PUT api/values/5
    public void Put(int id, [FromBody]string value)
    {
    }

    // DELETE api/values/5
    public void Delete(int id)
    {
    }
}
```

Web API controller Base class
Handles Http GET request
http://localhost:1234/api/values
Handles Http GET request with query string
http://localhost:1234/api/values?id=1
Handles Http POST request
http://localhost:1234/api/values
Handles Http Put request
http://localhost:1234/api/values?id=1
Handles Http DELETE request
http://localhost:1234/api/values?id=1

Web API Controller Characteristics

- It must be derived from System.Web.Http.ApiController class.
- It can be created under any folder in the project's root folder. However, it is recommended to create controller classes in the Controllers folder as per the convention.
- Action method name can be the same as HTTP verb name or it can start with HTTP verb with any suffix (case in-sensitive) or you can apply Http verb attributes to method.
- Return type of an action method can be any primitive or complex type.

Action Method Naming Conventions

- Action method name can be the same as HTTP verbs like Get, Post, Put, Patch or Delete as shown in the Web API Controller example above.
- However, you can append any suffix with HTTP verbs for more readability.
- For example, Get method can be GetAllNames(), GetStudents() or any other name which starts with Get.

HTTP Method	Possible Web API Action Method Name	Usage

GET	Get()
	get()
	GET()
	GetAllStudent()
	*any name starting with Get *
		Retrieves data.

POST	Post()
	post()
	POST()
	PostNewStudent()
	any name starting with Post
		Inserts new record.

PUT	Put()
	put()
	PUT()
	PutStudent()
	any name starting with Put
		Updates existing record.

PATCH	Patch()
	patch()
	PATCH()
	PatchStudent()
	any name starting with Patch
		Updates record partially.

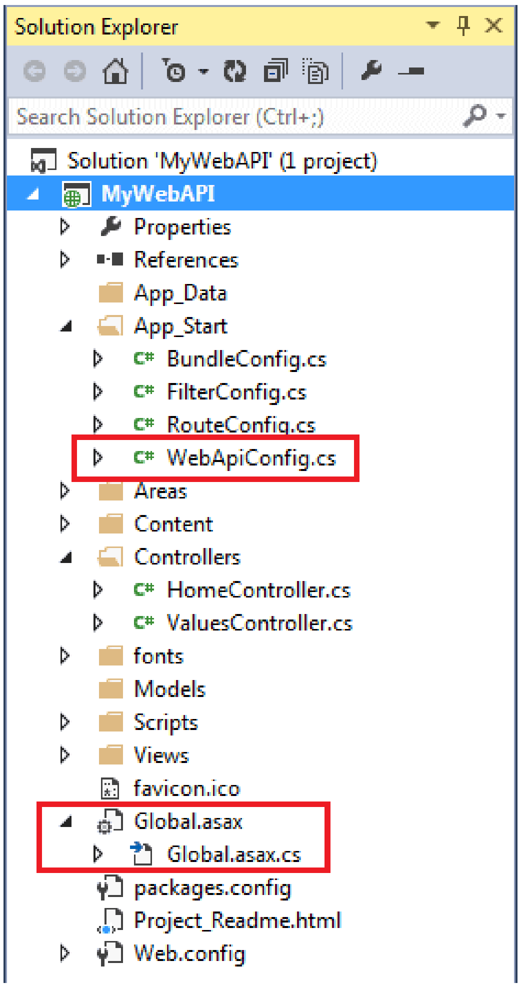
DELETE	Delete()
	delete()
	DELETE()
	DeleteStudent()
	any name starting with Delete
		Deletes record.

Difference between Web API and MVC controller

Web API Controller	MVC Controller
Derives from System.Web.Http.ApiController class	Derives from System.Web.Mvc.Controller class.
Method name must start with Http verbs otherwise apply http verbs attribute.	Must apply appropriate Http verbs attribute.
Specialized in returning data.	Specialized in rendering view.
Return data automatically formatted based on Accept-Type header attribute. Default to json or xml.	Returns ActionResult or any derived type.
Requires .NET 4.0 or above	Requires .NET 3.5 or above

Configure Web API

- Web API supports code based configuration. It cannot be configured in web.config file.
- We can configure Web API to customize the behaviour of Web API hosting infrastructure and components such as routes, formatters, filters, DependencyResolver, MessageHandlers, ParamterBindingRules, properties, services etc.
- Web API project includes default WebApiConfig class in the App_Start folder and also includes Global.asax as shown below.



Global.asax

```
public class WebAPIApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        GlobalConfiguration.Configure(WebApiConfig.Register);

        //other configuration
    }
}
```

WebApiConfig.cs

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {

        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );

        // configure additional webapi settings here..
    }
}
```

- Web API configuration process starts when the application starts.
- It calls GlobalConfiguration.Configure(WebApiConfig.Register) in the Application_Start method.
- The Configure() method requires the callback method where Web API has been configured in code. By default this is the static WebApiConfig.Register() method.
- As you can see above, WebApiConfig.Register() method includes a parameter of HttpConfiguration type which is then used to configure the Web API.
- The HttpConfiguration is the main class which includes following properties using which you can override the default behaviour of Web API.

Property	Description
DependencyResolver	Gets or sets the dependency resolver for dependency injection.
Filters	Gets or sets the filters.
Formatters	Gets or sets the media-type formatters.
IncludeErrorDetailPolicy	Gets or sets a value indicating whether error details should be included in error messages.
MessageHandlers	Gets or sets the message handlers.
ParameterBindingRules	Gets the collection of rules for how parameters should be bound.
Properties	Gets the properties associated with this Web API instance.
Routes	Gets the collection of routes configured for the Web API.
Services	Gets the Web API services.

Web API Routing

It routes an incoming HTTP request to a particular action method on a Web API controller.

Web API supports two types of routing:

1. Convention-based Routing
2. Attribute Routing

Convention-based Routing

- In the convention-based routing, Web API uses route templates to determine which controller and action method to execute.
- At least one route template must be added into route table in order to handle various HTTP requests.

Attribute Routing

- Attribute routing is supported in Web API 2.
- As the name implies, attribute routing uses [Route()] attribute to define routes.
- The Route attribute can be applied on any controller or action method.
- In order to use attribute routing with Web API, it must be enabled in WebApiConfig by calling config.MapHttpAttributeRoutes() method.

Example

```
public class StudentController : ApiController
{
    [Route("api/student/names")]
                public IEnumerable<string> Get()
    {
                return new string[] { "student1", "student2" };
    }
}
```

- In the above example, the Route attribute defines new route "api/student/names" which will be handled by the Get() action method of StudentController.
- Thus, an HTTP GET request http://localhost:1234/api/student/names will return list of student names.