

Title

Creation & Configuration of repository using GitHub, Git & GitHubDesktop

Problem Statement

MVC stands for Model View Controller, it's a framework or a design pattern used for the development of Web Application, nowadays its very high in demand for the development of an application with loosely coupled code.

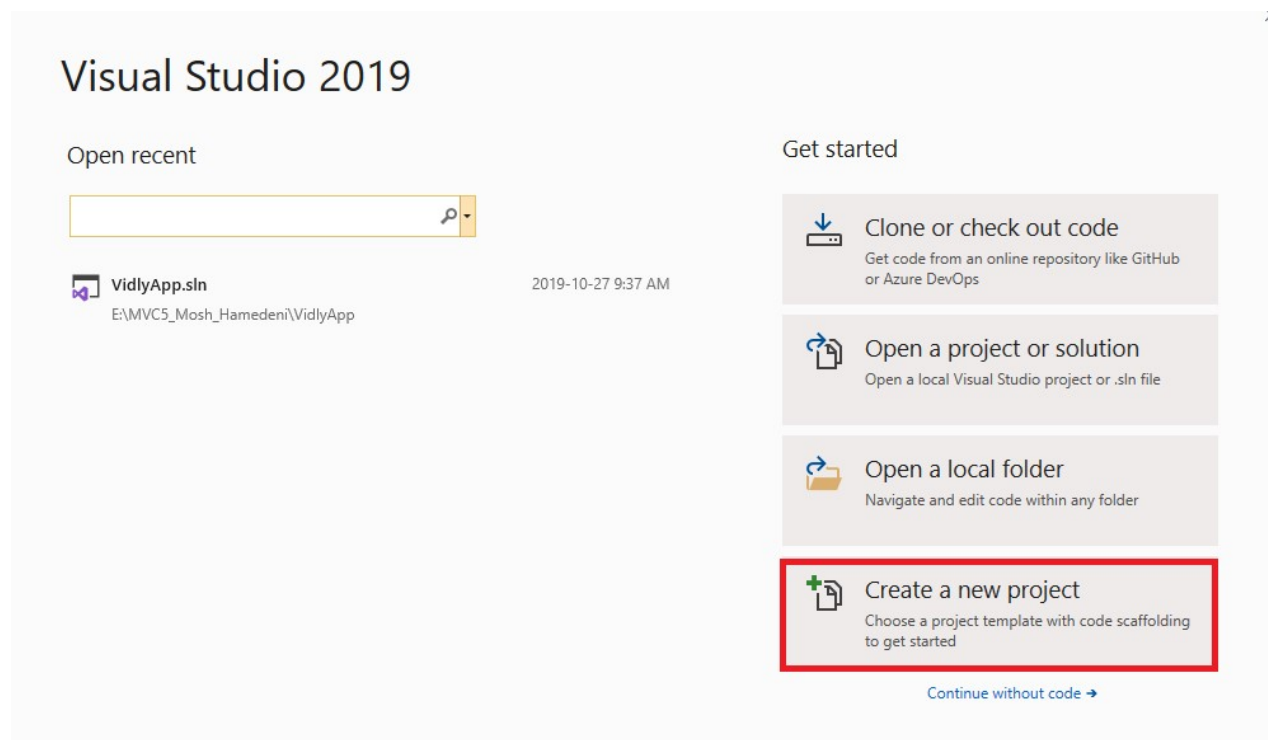
In MVC we have a separation of code that can be developed, tested & maintained easily. In MVC, the application can be separated using three different folders such as Models, Views, & Controller, where the Models folder contains all the Models or data, Controllers folder have all the controllers and Views folder contains all the views or .cshtml files related to the particular application.

Whenever a user requests a webpage, the controller is responsible for handling the HTTP request, the controller interacts with the appropriate model by interacting with the database by fetching the data and stores in the model and return back to the controller. The Controller in turns returns the data to the view for displaying the required data to the user.

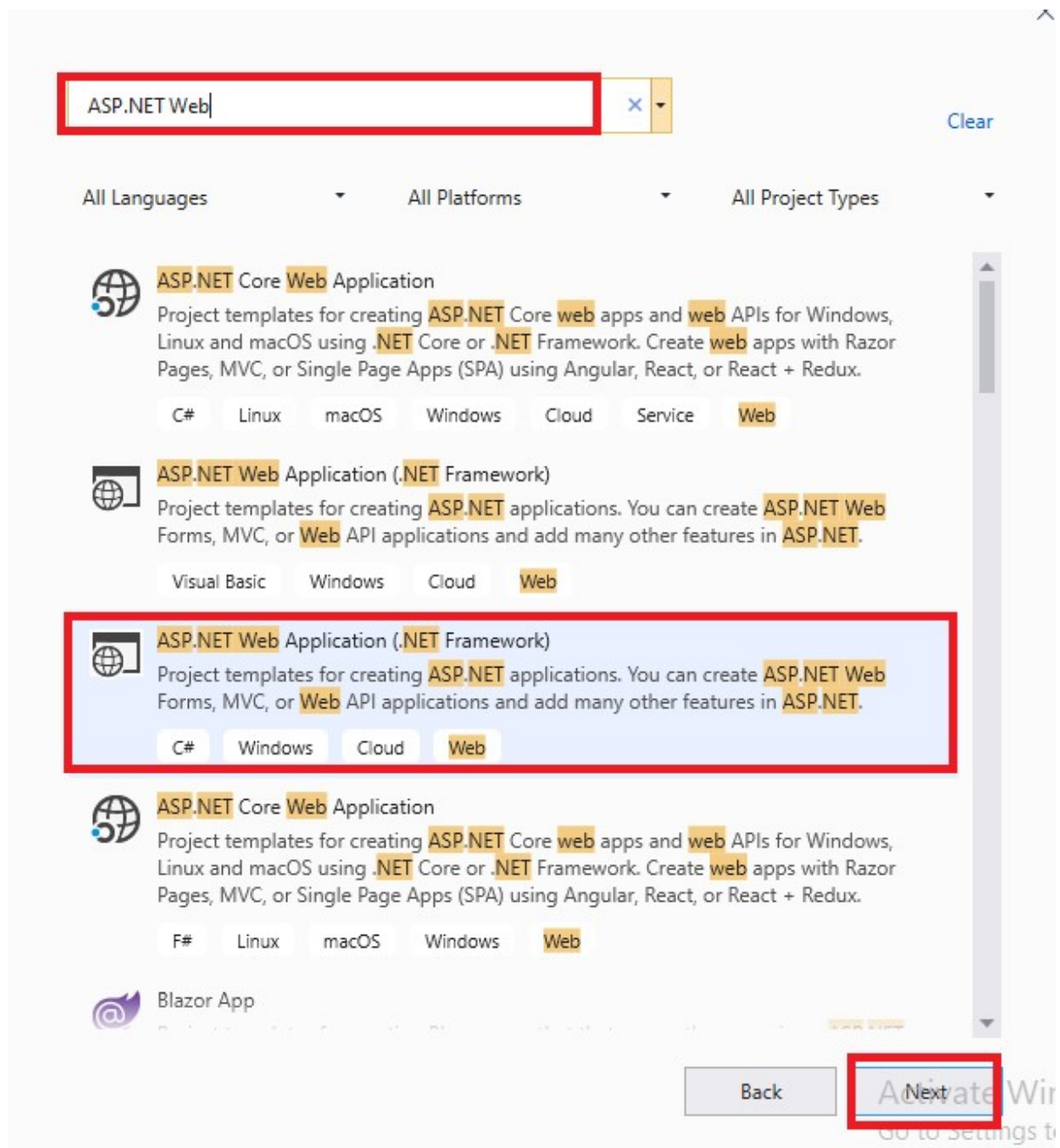
In this article we will see the autogenerated folder structure of MVC, the autogenerated folder structure is created whenever we create a new MVC application. Here I am using visual studio 2019 community edition, you can use any version of your choice, in my case I am going with visual studio 2019.

In order to create MVC 5 application follow the below steps:

Step 1: Open visual studio 2019 and click on Create a new project on the bottom right.



Step 2: Search ASP.NET Web Application in the above-provided textbox and click on the template and click on the Next button as shown below.



Step 3: Provide the project name say MVCFirstProject and click on Create button.

Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name
MVCFirstProject

Location
E:\MVC_Article\

Solution name ⓘ
MVCFirstProject

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Back Create

Step 4: Select the MVC template and click on the Create button, with this a new MVC project is created. Let's see the MVC folder structure which is created when we create the above MVC application. Here we have different folders like

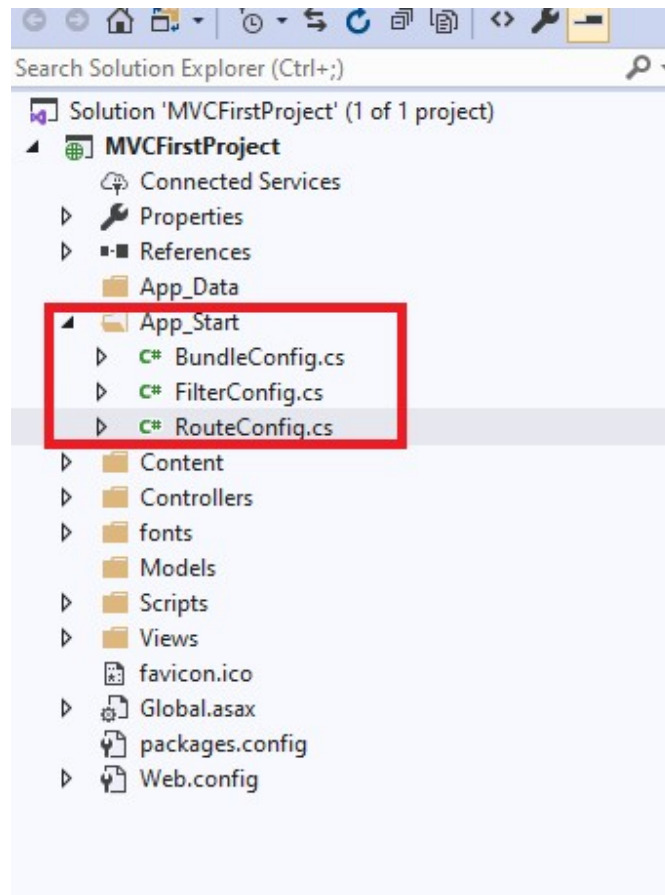
- App_Data
- App_Start
- Content
- Controllers
- fonts
- Models
- Scripts
- Views etc.

App_Data: The App_Data contains all the database related files.

App_Start: This folder contains all the configuration files for the application.

For Example:

BundleConfig.cs
FilterConfig.cs &
RouteConfig.cs



The RouteConfig.cs contains the routing configuration defined by the RouteConfig class and RegisterRoutes static method.

```
namespace MVCFirstProject
{
    1 reference
    public class RouteConfig
    {
        1 reference
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute(url: "{resource}.axd/{*pathInfo}");

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

In the above Url we have a pattern with {Controller}/{action}/{id}. And default Controller is the Home and Action method is Index.

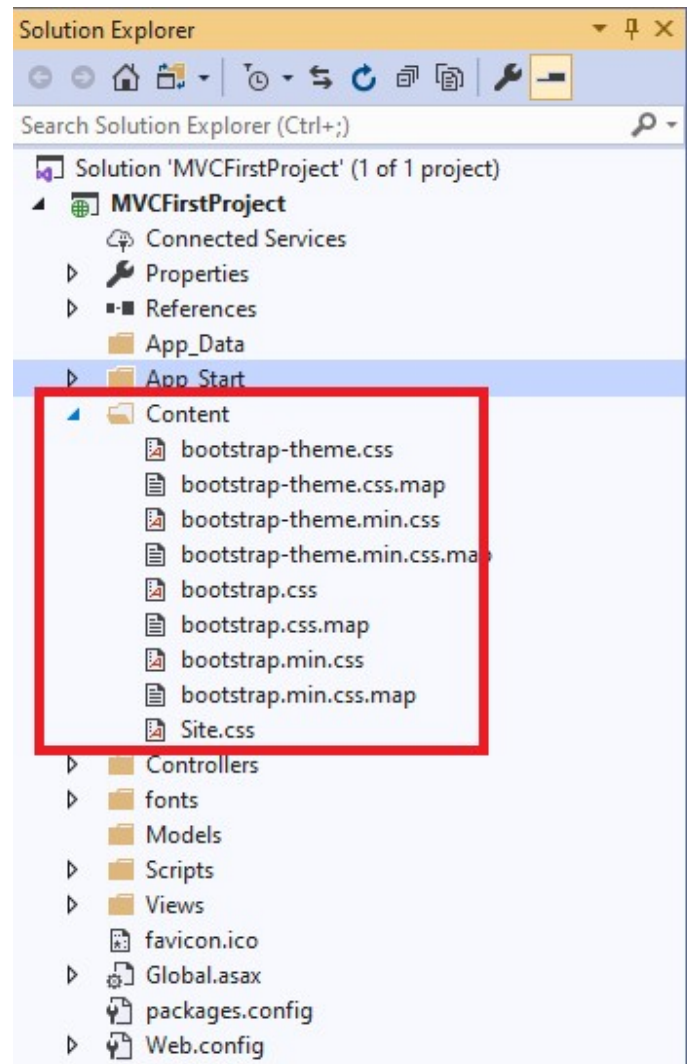
Whenever user request for a webpage by providing the Url say,
localhost:40366/Employee/EmployeeInfo/1

The request will be handled by the EmployeeController and its EmployeeInfo action method by passing the value 1 to the EmployeeInfo method's parameter,

Suppose in a controller's EmployeeInfo action method, if we provide a parametername which is not matching with the Url pattern, then MVC is unable to handle the request by the Employee Controller as there is a mismatch of a parameter name, which gives an error message.

If we provide a Url say, localhost:40366/Employee. Then with this Url, Employee Controller handles the request and the Index method is executed as provided action is Index in defaults field. If we provide a Url, localhost:40366/, without providing the controller name or action method name then by default HomeController and Index action method is called which is defined in defaults field which is defined in RegisterRoutes method.

Content: This folder contains all the CSS related files such as bootstrap.css, bootstrap.min.css, etc.



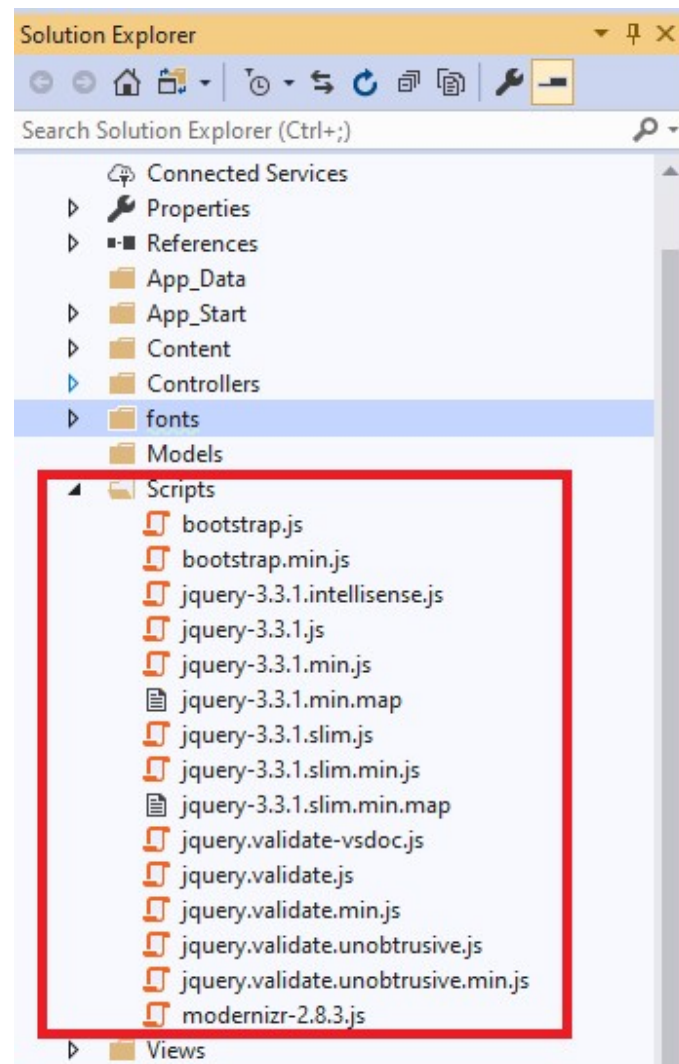
Controllers: This folder contains all the controller for the particular application, by default it will come with HomeController, each controller should be postfix with name Controller.

Fonts: This folder is used for storing images, icons, etc.

Models: Models folder contains all the Models related data for the application, a model is nothing but the properties for storing the data which can be used in the application

for assigning the data and binding it to the view. Based on the functionality or requirement we create multiple Models.

Scripts: Scripts folder contains all the javascript files which are required for the particular application, by default MVC application provides various javascript files such as bootstrap.js, jquery-3.3.1.js, jquery-3.3.1.min.js, etc.



Views: This folder contains all the views for the application, whenever we create a view, it creates a separate folder with the controller's name except the postfix name controller. View folder will have all the views depending on the Controller's name.

For Example: If we have 3 different Controllers say, AccountController, HomeController, LoginController then the views will have 3 different folders with names as Account, Home and Login, etc.

Package.config: This is the configuration file that contains all the dependency files or references of the application. Whenever we add a new reference from NuGet Package Manager, one of the references will be added to the package.config file.

Web.config: This is the configuration file for the application which is used for defining the connectionStrings, AppSettings, etc.

In order to create a sample MVC project, we will add a model to the Models folder and add few properties like below.

Right-click on Models folder--> Add --> select class and give the classname as Employee.cs and click on Add. With this, a new Employee Model is created.

Open Employee.cs and add few properties like below

```
1. namespace MVCFirstProject.Models
2. {
3.     public class Employee
4.     {
5.         public int EmployeeId { get; set; }
6.         public string EmployeeName { get; set; }
7.         public string EmployeeLocation { get; set; }
8.     }
9. }
```

Similarly to add a Controller called EmployeeController, right-click on Controllers folder, click on Add and select the Controller, In Add New Scaffolded Item Window box, select MVC5 Controller - Empty and click on Add, give the Controller name as EmployeeController and click on Add, with this a new controller called EmployeeController is added to Controllers folder.

Whenever we create a new Controller in the Controller's folder, a new folder will be created in views folder which has the same name as controller name except for the postfix Controller. In Employee Controller's EmployeeInfo action method,

create the instance of the Employee model by assigning the value to the properties as shown below.

```
1. namespace MVCFirstProject.Controllers
2. {
3.     public class EmployeeController : Controller
4.     {
5.         // GET: Employee
6.         public ActionResult EmployeeInfo()
7.         {
8.             Employee employee = new Employee()
9.             {
10.                 EmployeeId = 1001,
11.                 EmployeeName = "Khaja Moiz",
12.                 EmployeeLocation = "Hyderabad"
13.             };
14.
15.
16.             return View(employee);
17.         }
18.     }
19. }
```

Right-click on Employee folder in Views folder, click on Add and select Add View. In Add View window box, give the view name as the Action method name that is EmployeeInfo, template as Empty(without model) and click on Add. With this EmployeeInfo.cshtml view is added to the Employee View folder.

Open the EmployeeInfo.cshtml file and paste the below code.

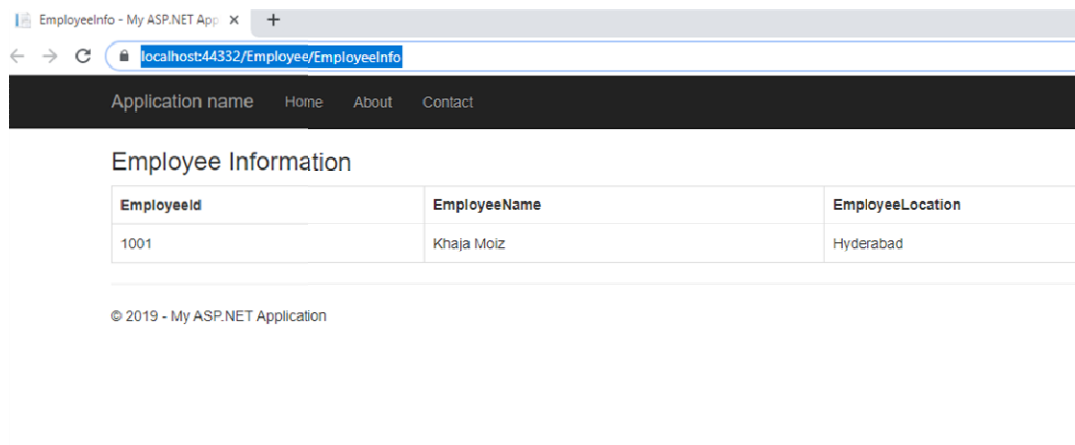
```
1. @model MVCFirstProject.Models.Employee
2.
3. @{
4.     ViewBag.Title = "EmployeeInfo";
5.     Layout = "~/Views/Shared/_Layout.cshtml";
6. }
7.
8. <style>
```

```
9.  table {
10.    font-family: arial, sans-serif;
11.    border-collapse: collapse;
12.    width: 100%;
13.  }
14.
15.  td, th {
16.    border: 1px solid #dddddd;
17.    text-align: left;
18.    padding: 8px;
19.  }
20.</style>
21.
22.<h3>Employee Information</h3>
23.
24.<table>
25.  <tr>
26.    <th>EmployeeId</th>
27.    <th>EmployeeName</th>
28.    <th>EmployeeLocation</th>
29.  </tr>
30.
31.  <tr>
32.    <td>@Model.EmployeeId</td>
33.    <td>@Model.EmployeeName</td>
34.    <td>@Model.EmployeeLocation</td>
35.  </tr>
36.</table>
```

Now build and run the solution by navigating to <https://localhost:44332/Employee/EmployeeInfo>. With the above request, EmployeeInfo of EmployeeController handles the request, In EmployeeInfo Action method we are creating the instance of the Employee model by assigning the hardcoded value and returning the Employee object through the view.

In EmployeeInfo.cshtml, the returned view will be binded to the HTML table and displayed to the user.

The output is shown below.



Assignment

1. Install Git on your local machine.
2. Install GitHubDesktop on your local machine.
3. Go to <https://github.com/join>
4. Type a user name, your email address, and a password.
5. Choose Sign up for GitHub, and then follow the instructions.
6. On the GitHub home page, do one of the following:
7. In Your repositories, choose new repository.
8. On the navigation bar, choose Create new (+), and then choose New repository.
9. In the Create a new repository page, do the following:
10. In the Repository name box, enter GitHubDemo.
11. Select Public.
12. Note
13. Selecting the default Public option means that anyone can see this repository. You can select the Private option to limit that can see and commit to the repository.
14. Check Initialize this repository with a README check box.
15. Choose Create repository.
16. Clone repository with your local machine
17. Copy a sample application to your local repository
18. Push the sample application from the local machine to the GitHub repository using Git & GitHubDesktop