Agenda: ASP.NET Core MVC Application

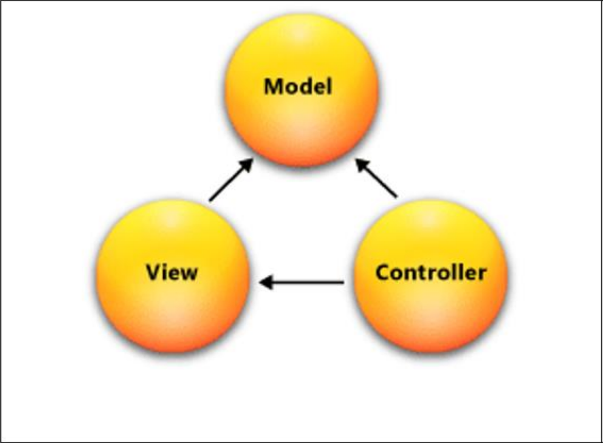
- MVC Pattern
- Understand MVC in context of ASP.NET
- Advantages of ASP.NET MVC
- First ASP.NET Core MVC Application

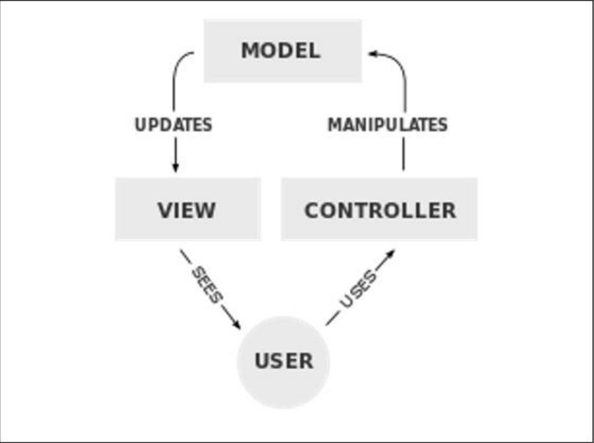
What is MVC Pattern

MVC stands for Model - View – Controller.

It is architecture for developing **interactive** applications where there would be a user interaction involved and event handling would occur.

What is role of Model, View and Controller?





- **Model**: It manages data basically state of application in memory. There is no fixed size or shape of model objects since what we intent to hold in memory in one application may not be the same as that in other application. It includes all of an application’s **validation logic, business logic and data access** logic. For example, an **Employee object (Model)** might retrieve information from a database, operate on it, validate it and then write updated information back to an Employee table in database.
- **View**: It contains logic for rendering Graphical Representation / HTML output. Typically it creates UI with data from Model. An example would be an edit view of a Products table that displays text boxes, drop-down lists, and check boxes based on the current state of a Products object.
- **Controller**: It contains **control flow logic**. It is the one which interacts with both models and views to control the flow of application execution. The controller handles and responds to user input and interaction by creating and passing model to view. A controller can send commands to the model to update the model's state.

Understanding MVC in context of ASP.NET Core

What is a Model?

1. MVC **model** is basically a C# class
2. A **model** is accessible by both **controller** and **view.**
3. A **model** can be used to pass data from **Controller** to **View**
4. A **view** can use model to display data in page (HTML output).

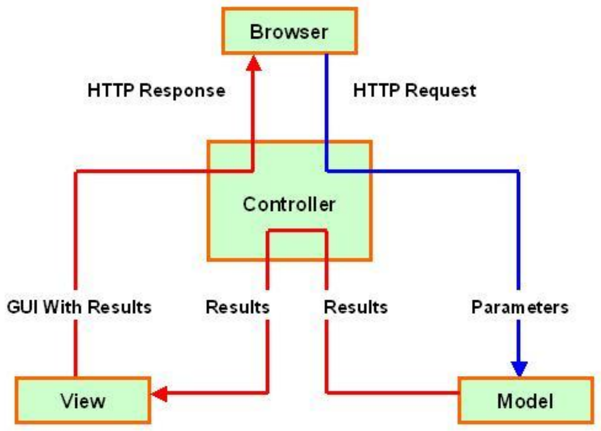
What is a View?

1. View is an CSHTML page without having a code behind file.
2. All page specific HTML generation and formatting can be done inside view.
3. One can use Inline code (server tags) to develop dynamic pages.
4. A request to **view** can be made only from a controller's action method.

What is a Controller?

1. **Controller** is basically a C# class which inherits **Microsoft.AspNetCore.MVC.Controller**
2. **Controller** is a heart of the entire MVC architecture.
3. Inside **Controller's class** action methods can be implemented which are responsible for responding to browser OR calling views.
4. **Controller** can access and use **model** class to pass data to **views**.

Understanding Life Cycle of ASP.NET MVC Request



Advantages of an ASP.NET MVC

- Its **light weight** because it does not use view state or server-based forms or server controls.
- It makes it easier to manage **complexity by dividing** an application into the model, the view, and the controller.
- **Separating the view** from rest of the application logic enables changing of view (technology) in future without affecting the rest of the application. For example you might have views in Silverlight or HTML 5.
- Each developer based on his expertise can **work on different parts** of the application without stepping on each other toes. For example, one developer can work on the view, a second developer can work on the controller logic, and a third developer can focus on the business logic in the model.
- **RESTful / User friendly Url's** and this enables SEO.
- **Clean HTML** and easy integration with JavaScript and JQuery.
- It provides better support for **test-driven development** (TDD). This is because we can focus on one aspect at a time i.e. we can focus on view without worrying about business logic.
- It's created to support **pattern-based** software development.
- It works well for Web applications that are supported by **large teams** of developers and Web designers who need a high degree of control over the application behavior. Easy to maintain since we can clearly pin point what code to open in case there are any changes to be done.

First ASP.NET Core MVC Web Application

1. Start Visual Studio 2017 → File → New Project → Visual C# → .NET Core → ASP.NET Core Web Application.
2. Name = FirstAspNetCoreWebAppl
3. From dropdown choose .NET Core and ASP.NET Core 2.0, Web Application → OK

Note:

1. .NET Core: It's web template for cross platform compatible project that runs on .NET Core framework.
2. .NET Framework: This starts a new project that runs on the standard .NET Framework on Windows.

Writing a Basic Form

1. To the project add the below class

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

2. Right Click on Controller Folder → Add → Controller → Name = "PersonController" → OK
3. Edit PersonController.cs

```
public class HomeController : Controller
{

    public IActionResult Index()
    {
        Person p = new Person();
        p.Name = "Sandeep";
        p.Id = 1;
        return View(p);
    }

    [HttpPost]
    public IActionResult Index(Person person)
    {
        ViewBag.Message = person.Id + " " + person.Name;
        return View(person);
    }
}
```

4. Edit Person/Index.cshtml

```
@model WebApplication5.Models.Person

<form method="POST">
    <div>Id: <input asp-for="Id" /></div>
    <div>Name: <input asp-for="Name" /></div>
    <input type="submit" />
</form>
<hr />
@ViewBag.Message
```

5. Visit http://localhost:5000/Person