ASP .NET Core MVC

**Advantages:**

* Fast and open source
* Cross platform
* Built in dependency injection
* Easy Updates
* Cloud Friendly
* Good Performance

**Prerequisites:**

* C# basics
* SQL

**Tools Needed:**

* Visual studio 2022
* SSMS

**What is Model:**

* The model classes represents 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.
* In the ASP.NET MVC Application, all the Model classes must be created in the Model folder.
* Let's create the model class that should have the required properties for the Student entity.
* In the MVC application in Visual Studio, and right-click on the Model folder, select Add -> and click on Class... It will open the Add New Item dialog box.
* In the Add New Item dialog box, enter the class name Student and click Add.
* This will add a new Student class in model folder. 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.

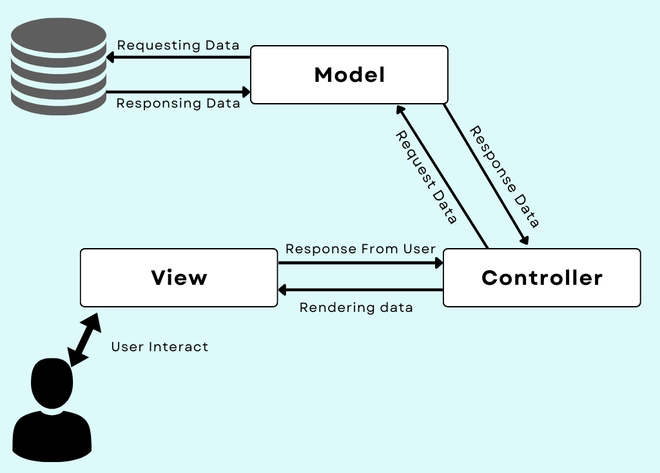
**View:**

* The MVC View is a standard HTML page that may contain script. It is used to create web pages for the application. Unlike ASP.NET Web Pages, MVC Views are mapped to the action and then controller renders the view to the browser.
* MVC has certain conventions for project structure. The view file should be located in the subdirectory of View folder.
* MVC uses Razor view engine so that we can write server side code in HTML as well. Let's create a view and execute it to the browser.

**Controller:**

* 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.

**MVC Architecture :**



* **Passing Data from Controller to View in ASP .NET core :**

You can use following object to pass data between controller and view:

1. ViewData
2. ViewBag
3. TempData
4. Strongly Typed Views
5. ViewData:

* It passes data from controller to a view.
* It is a dictionary of object that is derived from the ViewDataDictionary class.
* Some of the characteristics of ViewData are as follows:

1. The life of a ViewData object exists only during the current request.
2. The value of ViewData is null if the request is redirected.
3. It not give any compile time error checking.

* The general syntax of a ViewData is as follows
* ViewData[“Key”] = <Value>; where Key is a string value to identify the object present in ViewData.
* Value is the object present in ViewData. This object may be a string , object , list , array or a different type such as int , char, float, double etc. Declaration of ViewData

public IActionResult Index()

{

ViewData["data1"] = "kalpana";

ViewData["data2"] = 23;

string[] arr = { "kalpana", "panigrahi" };

ViewData["data3"] =arr;

return View();

}

1. ViewBag :

* Used to pass data from controller to view.
* The general syntax : ViewBag. <PropertyNmae> = <value>;
* Where Property is a string value that represents a property of ViewBag. Value is the property of ViewBag, Value may be string , object , list, array or different types such as int , float, chr etc.
* ViewBag is a dynamic data type property of the base class of all controllers, which is the ControllerBase class.
* ViewBag is a dynamic data type which internally uses Viewdata.
* ViewBag exist only for the current request and becomes null if the request is redirected.
* ViewBag does not require typecasting when you use complex data type.
* It does not provide compile time error checking.
* Declaration:

ViewBag.data1 = "My name";

ViewBag.data2 ="Is Kalpana";

1. TempData :

* Used to pass data from controller to view.
* TempData is used only for current or subsequent request as it is a very short- lived instance.
* Redirecting is the only case when users can rely on TempData.
* TempData stores data as key-value pairs.
* TempData value must be type cast before use.

**MODELS:**

* A model is a class with .cs (for c#) as an extension having both properties and methods.
* Models are used to set or get the data.
* If your application does not have data, then there is no need for a model.
* The models in ASP .NET core MVC contains a set of classes that are used to represent the domain data as well as it also contains logic to manage the data.
* We can perform operations like insert, update, select by using models.
* It is not mandatory but is a good programming practice to store all the model classes within the models folder.

Repository Pattern:

* With the Repository pattern we create an abstraction layer between the data access and the business logic layer of an application.
* By using it we promoting a more loosely coupled approach to access our data from the database.
* Also code is cleaner and easier to maintain and reuse.

1. Strongly Typed View:

* Strongly typed view or strongly typed object is used to pass data from controller to a view.
* The view which binds with any model is called as strongly typed view.
* You can bind any class as a model to view.
* You can access model properties on that view.
* The view that is designed by targeting specific model class object then that view is called “Strongly Typed View”.
* In strongly typed view, view is bind with corresponding model class object or list of objects.

Tag Helper:

* Tag helpers are basically special attributes provided by ASP .NET core.
* Tag helpers enables server-side components to participate in creating and rendering HTML elements in Views.
* Tag helpers are a new feature and similar to HTML helpers which help us render HTML.
* There are many built-in tag helpers for common tasks such as creating forms, hyperlinks, loading assets etc.
* Tag helpers are authored in C# and they target HTML elements based on elements name.
* For example the built in tag helper LabelTagHelper can target the HTML <label> element when the LabelTagHelper attributes are applied.
* The namespace for tag helper is Microsoft.AspNetCore.Mvc.TagHelper.