UNIT I: Introduction to .NET Core and MVC 6

1.1 Introduction to .NET Core 6.0

ASP.NET Core (.NET) is a free, open-source, and cloud-optimized framework that can run on Windows, Linux, or macOS.

It is the new version of ASP.NET. The framework was completely rewritten to be open-source, modular, and cross-platform.

.NET Core is the new version of the .NET Framework, a free, open-source, general-purpose development platform maintained by Microsoft.

NET Core aimed to provide a unified platform for developing various applications, including web applications, desktop applications, microservices, and more.

.NET Core is written from scratch to be a modular, lightweight, fast, and cross-platform framework.

It includes the core features required to run a basic .NET Core app.

.NET Core Characteristics:

Open-Source Framework:

.NET Core is an open-source framework maintained by Microsoft and available on GitHub under MIT and Apache 2 Licenses.

CLI Tools:

.NET Core includes CLI tools (Command Line Interface) for development and continuous integration.

Flexible Deployment:

.NET Core applications can be deployed user-wide or system-wide or with Docker Containers.

Compatibility: Compatible with .NET Framework and Mono APIs using .NET Standard Specification.

1.2 Introduction to MVC 6

MVC stands for Model View and Controller.

It is an Architectural Design Pattern, which means it is used at the application's architecture level.

MVC is not a programming language, not a Framework. It is a Design Pattern.

When we design an application, we first create its architecture, and MVC plays an important role in designing that architecture.

The MVC (Model-View-Controller) Design Pattern was introduced in the 1970s.

The main objective of the MVC Design Pattern is the separation of concerns. This means the Domain Model and Business Logic are separated from the User Interface (i.e., View). As a result

The Controller is the Component in the MVC design pattern that handles the incoming request.

The controller components do several things to handle the request.

The controller component creates the model that is required by a view.

The model is the component in the MVC design pattern, which basically contains classes that are used to store the domain data.

In the MVC design pattern, the Model component also contains the required logic to retrieve data from a database.

The Model in an MVC application represents the application's state and business logic. That means the Model is the component in the MVC Design pattern used to manage the data, i.e., the application's state in memory.

The View is the Component in the MVC Design pattern that contains the logic to represent the model data as a user interface with which the end-user can interact.

The Controller is the component in an MVC application that handles the incoming HTTP Request.

Based on the user action, the respective controller might work with the model, select a view to render the information, and then send the response back to the user who initially made the request.

1.3 NET Web Forms (vs.) ASP.NET MVC

Net Web Forms

Asp.Net Web Form follows a traditional event-driven development model.

Asp.Net Web Form has server controls.

Asp.Net Web Form supports view state for state management on the client side.

Asp.Net Web Form has file-based URLs means the file name exist in the URLs must have its physical existence.

Asp.Net Web Form follows Web Forms Syntax

In Asp.Net Web Form, Web Forms(ASPX) i.e. views are tightly coupled to Code behind(ASPX.CS) i.e. logic.

Asp.Net Web Form has Master Pages for a consistent look and feel.

Asp.Net Web Form has User Controls for code re-usability.

Asp.Net Web Form has built-in data controls and is best for rapid development with powerful data access.

Asp.Net Web Form is not Open Source.

ASP.NET MVC

Asp.Net MVC is lightweight and follows the MVC (Model, View, Controller) pattern-based development, model.

Asp.Net MVC has HTML helpers.

Asp.Net MVC does not support view state.

Asp.Net MVC has route-based URLs means URLs are divided into controllers and actions and it is based on the controller not on the physical file.

Asp.Net MVC follows customizable syntax (Razor as default)

In Asp.Net MVC, Views and logic are kept separately.

Asp.Net MVC has Layouts for a consistent look and feel.

Asp.Net MVC has Partial Views for code re-usability.

Asp.Net MVC is lightweight, provides full control over markup, and supports many features that allow fast & agile development. Hence it is best for developing an interactive web application with the latest web standards.

Asp.Net Web MVC is an Open Source.

1.4 List of Versions of ASP.NET MVC

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.

Microsoft had introduced ASP.NET MVC in .NET 3.5, since then lots of new features have been added.

Microsoft made ASP.NET MVC framework open-source in April 2009.

Following are the versions of ASP .NET MVC

1. ASP.NET MVC 1.0

It support Visual Studio 2008 and .NET Framework 3.5 Released on 13 march 2009.

2. ASP.NET MVC 2.0

It support Visual Studio 2008 and .NET Framework 3.5/4.0 Released on 10-Mar-2010.

3. ASP.NET MVC 3.0

It support Visual Studio 2010 and .NET Framework 4.0 Released on 13-Jan-2011. Razor view engine.

4. ASP.NET MVC 4.0

It support Visual Studio 2012 and .NET Framework 4.0/4.5 Released on 15-Aug-2012. Mobile project template.

5. ASP.NET MVC 5.0

It support Visual Studio 2013 and .NET Framework 4.5 Released on 17-oct-2013. Authentication filters

6. ASP.NET MVC 5.2

It support Visual Studio 2013 and .NET Framework 4.5 Released on 28-Aug-2014. Bug fixes and minor features update.

7. ASP.NET MVC 5.2.9 – Current

It support Visual Studio 2022 and .NET Framework 4.5 Released on June-2022

1.5 Differences between versions of ASP.NET MVC

Asp.Net MVC1

Released on Mar 13, 2009

Runs on .Net 3.5 and with Visual Studio 2008 & Visual Studio 2008 SP1

MVC Pattern architecture with WebForm Engine

Html Helpers

Ajax helpers

Asp.Net MVC2

Released on Mar 10, 2010

Runs on .Net 3.5, 4.0 and with Visual Studio 2008 & 2010

Strongly typed HTML helpers mean lambda expression HTML Helpers

UI helpers with automatic scaffolding & customizable templates

Attribute-based model validation on both client and server.

Overriding the HTTP Method Verb including GET, PUT, POST, and DELETE

Areas for partitioning large applications into modules

Asynchronous controllers

Asp.Net MVC3

Released on Jan 13, 2011

Runs on .Net 4.0 and with Visual Studio 2010

The Razor View engine

Improved Support for Data Annotations

ViewBag dynamic property for passing data from controller to view

jQuery Validation, and JSON binding

Use of NuGet to deliver software and manage dependencies throughout the platform

Good Intellisense support for Razor in Visual Studio

Asp.Net MVC4

Released on Aug 15, 2012

Runs on .Net 4.0, 4.5 and with Visual Studio 2010SP1 & Visual Studio 2012

ASP.NET Web API

Enhancements to default project templates

Mobile project template using jQuery Mobile

Task support for Asynchronous Controllers

Support for the Windows Azure SDK

Asp.Net MVC5

Released on 17 October 2013

Runs on .Net 4.5, 4.5.1 and with Visual Studio 2013

Authentication filters - run prior to authorization filters in the ASP.NET MVC pipeline

Bootstrap in the MVC template

ASP.NET Web API2

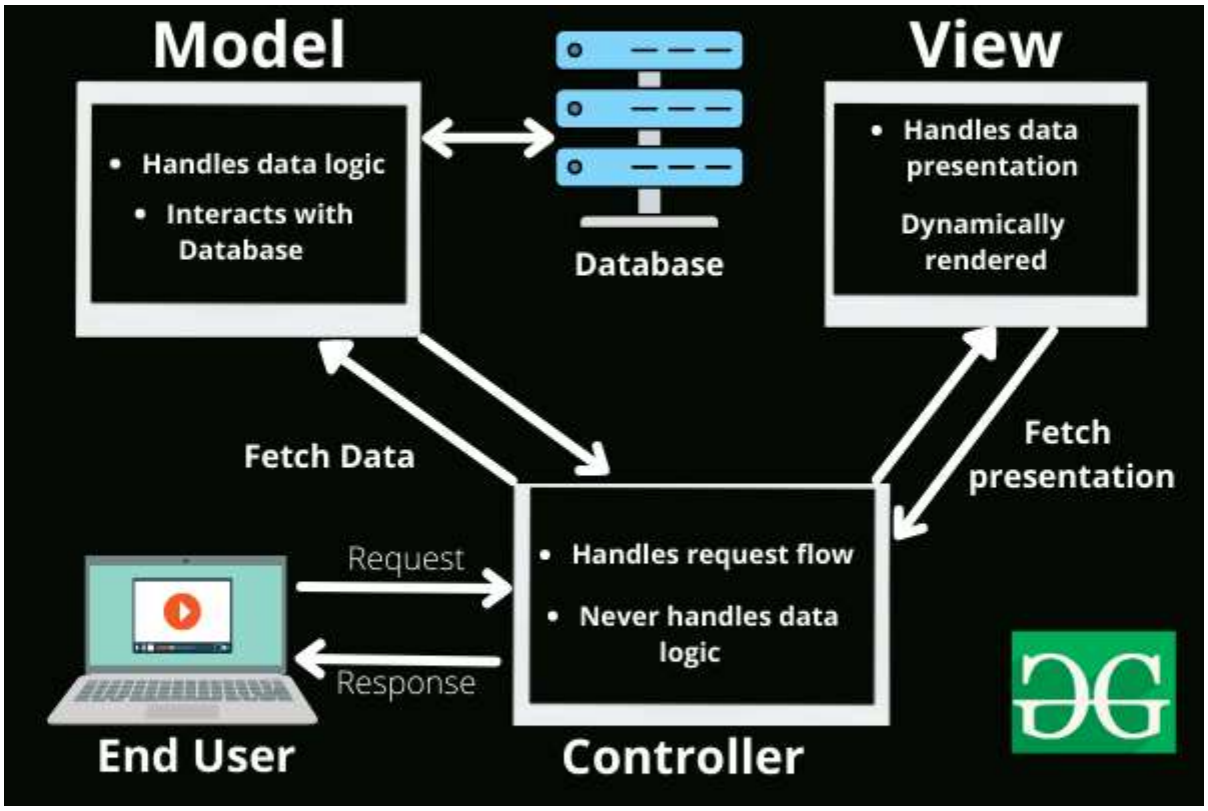
1.6 MVC Architecture

The MVC framework includes the following 3 components:

1. Controller

2. Model

3. View



1. Controller

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary.

The controller doesn't have to worry about handling data logic, it just tells the model what to do.

It processes all the business logic and incoming requests, manipulates data using the Model component, and interact with the View to render the final output.

Responsibilities:

Receiving user input and interpreting it.

Updating the Model based on user actions.

Selecting and displaying the appropriate View.

Example:

In a bookstore application, the Controller would handle actions such as searching for a book, adding a book to the cart, or checking out.

2. Model

The Model component corresponds to all the data-related logic that the user works with.

Model represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

It can add or retrieve data from the database.

It responds to the controller's request because the controller can't interact with the database by itself.

Responsibilities:

Managing data: CRUD (Create, Read, Update, Delete) operations.

Enforcing business rules.

Notifying the View and Controller of state changes.

Example:

In a bookstore application, the Model would handle data related to books, such as the book title, author, price, and stock level.

3. View/Action Method.

The View component is used for all the UI logic of the application.

It generates a user interface for the user.

Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller.

It only interacts with the controller.

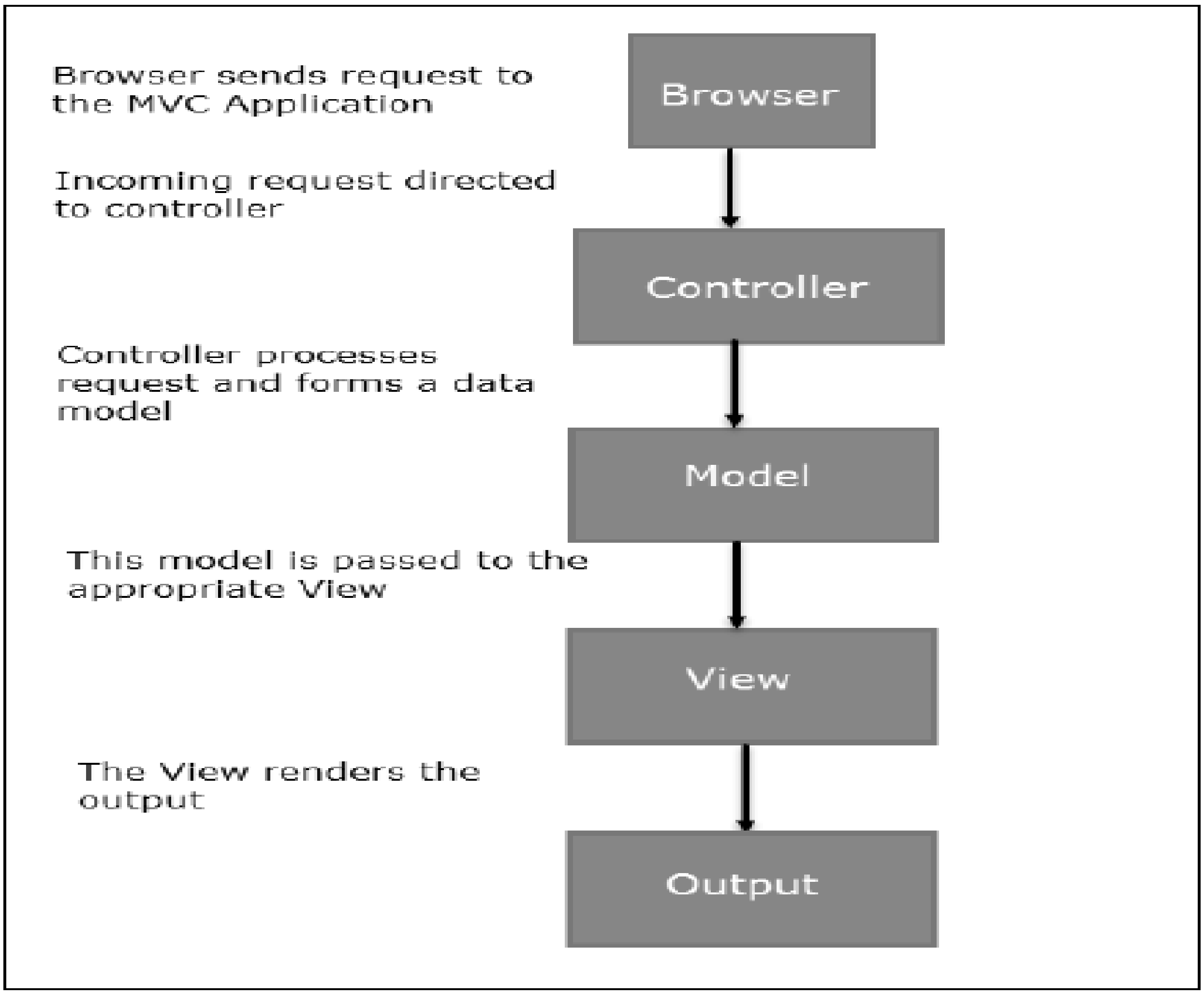
Responsibilities:

Rendering data to the user in a specific format.

Displaying the user interface elements.

Updating the display when the Model changes.

1.8 Request Flow in ASP.NET MVC


Browser sends request to the MVC Application
Browser
Incoming request directed to controller
Controller
Controller processes request and forms a data model
Model
This model is passed to the appropriate View
View
The View renders the output
Output

Request flow handles the request from the clients and passes it to the server.

Request is being taken from User to controller.

-Controller processes the request from the user and creates a data Model of that particular request.

-Data model that is being created is then passed to View that handles the frontend or the design.

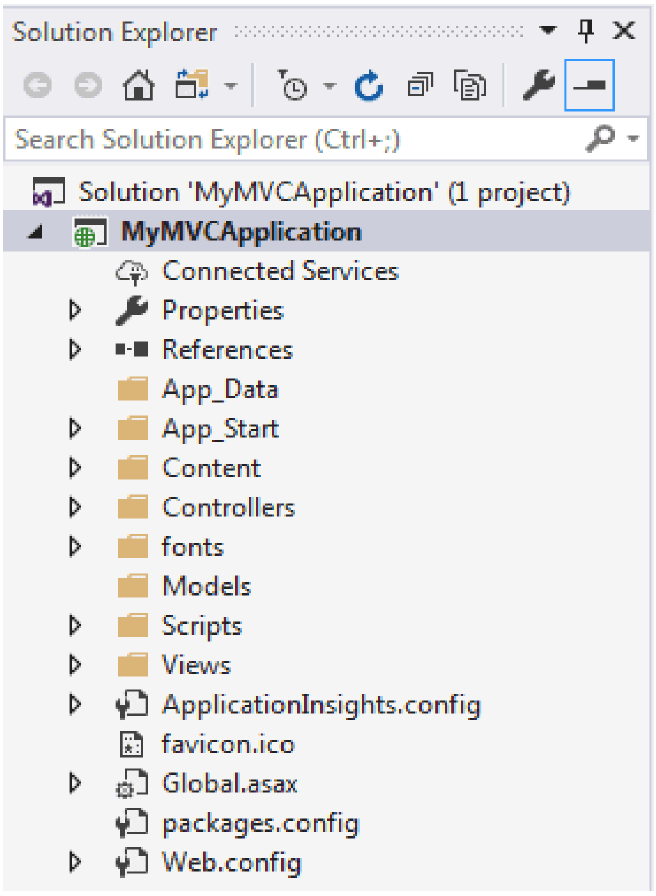
-View then transforms the Data Model by using its own functions in an appropriate output format.

-The output format that is being given by the View is then gets rendered to the Browser and the View will be seen by the user.

1.9 Overview of Folders and files of MVC project

The ASP.NET MVC project has a structure. We are using the IDE (Integrated Development Environment that is Visual Studio 2017 or latest.

Visual Studio creates the following folder structure of the ASP.NET MVC application by default.



```
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
   ▷  ApplicationInsights.config
      favicon.ico
   ▷  Global.asax
      packages.config
   ▷  Web.config
```

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. MVC 5 includes BundleConfig.cs, FilterConfig.cs and RouteConfig.cs by default.

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.

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.

Views : The Views folder contains HTML files for the application. Typically view file is a .cshtml file where you write HTML and C#.The Views folder includes a separate folder for each controller.

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.

Thank You