**Boppudi Vivek(916243)**

**Hands-On: Stage 3 - Benefits of .Net Core, .Net Core components and its importance - Day 62 – Hands on 1**

1. **Explain what is .Net core and its benefits over ASP.Net?**

ASP.NET Core is an open source, cross-platform and high-performance web framework to build the web application, which can be run on Windows, Mac or Linux operating system.

* A developer can build modern, cloud optimized, IoT, Mobile Backends application using ASP.NET Core framework.
* We can develop and run ASP.NET Core application on Windows, Mac, Linux operating system.
* We can also deploy our application on the cloud or run on-premises.
* ASP.NET Core runs on .NET Core framework or .NET Standard framework.
* ASP.NET Core framework is a combination of MVC and Web API in a single web framework.

**Prerequisites**

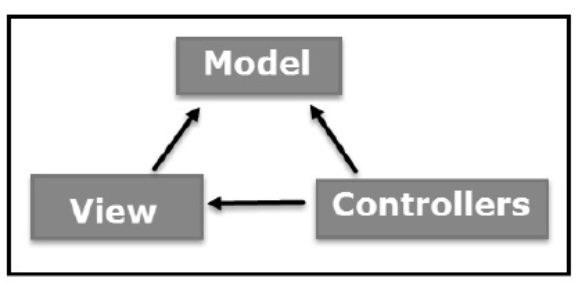
* Basic knowledge of C#
* Visual Studio
* Object Oriented Programming concepts

**Advantages of ASP.NET Core**

* Fast - It is a lightweight, high-performance web framework.
* Integration of Modern UI Framework - ASP.NET Core support modern, a client-side framework like AngularJs, ReactJs and React with Redux etc. ASP.NET framework supports client-side framework templates like AngularJs, ReactJs and React with Redux etc.
* Hosting - It has the ability to host on IIS, Apache, Docker or Self Hosting.
* Cross Platform - ASP.NET Core web application can run on Windows, Mac, Linux development tools.
* Support Built-In Dependency Injection - It supports built-in Dependency Injection.
* Supports Modular - It support modular HTTP request.
* Open-Source - It is an open-source and community-focused web framework.
* Side-by-side app versioning - ASP.NET Core runs on .NET Core which supports the simultaneous running of multiple versions of applications.
* A unified story for building web UI and web APIs.

1. **Explain what is MVC?**

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC Components

**Model**

* The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**View**

* The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**Controller**

* Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

1. **Explain folder structure of the application?**

**Folders Structure**

**wwwroot**

* The wwwroot folder in the ASP.NET Core application is treated as the web root folder and this folder or directory should be present in the root project folder.
* In ASP.NET Core Application, the Static files can be stored in any folder under the web root folder and can be accessed with a relative path to that root.
* Static content is hosted in this folder like CSS, JS files, Bootstrap, jQuery libraries and images.

**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. If your application has data, then you need a model.
* The Models in ASP.NET Core MVC contains a set of classes that are used to represent the domain data (you can also say the business data) as well as it also contains logic to manage the domain/business data.

**Controllers**

* A Controller is used to group actions i.e. Action Methods.
* The Controller is responsible to handle the incoming HTTP Request.
* The Mapping of the HTTP Request is done using Routing. That is for a given HTTP Request, which action method of which controller is going to invoke is handled by the Routing Engine.’
* Many important features such as Caching, Security, etc. can be applied to the controller.

**Views**

* the View is the component that contains logic to represent the model data (the model data provided to it by a controller) as a user interface with which the end-user can interact.
* The Views in MVC are HTML templates with embedded Razor mark-up which generate content that sends to the client.
* We display information about the website on the browser using the views only.
* A user generally performs all the actions on a view such as a button click, form, list, and other UI elements.

**Dependencies**

* It is the place where the necessary dll.s for the applications are stored.

**Properties-launchSettings.json**

* It describes how a project can be launched, whether the browser should be opened, and which environment variable should be set.

**appsettings.json**

* It stores the information of connection strings or application settings.
* It is similar to Web.config

**Startup.cs**

* Startup. cs file contains Startup class which triggers at first when application launches and even in each HTTP request/response.
* Startup. cs file is entry point of application level it handles request pipeline.

Startup class include two public methods Configure and ConfigureServices.

**ConfigureServices()**

* The ConfigureServices method is a place where you can register your dependent classes with the built-in IoC container. After registering dependent class, it can be used anywhere in the application. You just need to include it in the parameter of the constructor of a class where you want to use it.

**Configure()**

* The Configure method is a place where you can configure application request pipeline for your application using IApplicationBuilder instance that is provided by the built-in IoC container.
* The configure method includes three parameters IApplicationBuilder, IHostingEnvironment, and ILoggerFactory by default.

**First Console Application in .Net Core**

**Program.cs**

using System;

namespace ConsoleApp7

{

class Program

{

static void Main(string[] args)

{

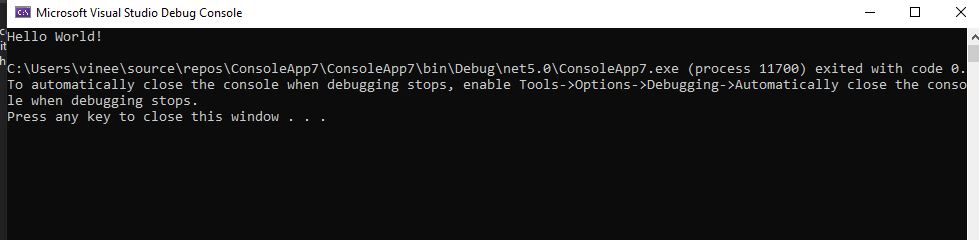
Console.WriteLine("Hello World!");

}

}

}

**OUTPUT**

****