**Tutorial 1**

* 1. What is ASP.NET MVC

ASP.NET MVC stands for Active Server Pages .NET Model-View-Controller. It's a web application framework developed by Microsoft, designed to build dynamic websites and web applications. MVC separates the application into three main components: Model (data), View (user interface), and Controller (logic). This architectural pattern helps in organizing code, enhancing testability, and facilitating parallel development.

* 1. Explain the role of a Controller in an ASP.NET MVC application.

In an ASP.NET MVC application, the Controller acts as the intermediary between the user's interactions (such as clicking a button or submitting a form) and the application's logic. It receives input from the user via HTTP requests, processes that input, interacts with the model to retrieve or update data, and then determines which view should be rendered to present the response back to the user. Essentially, the Controller handles the flow of the application and orchestrates the interaction between the Model and the View.

* 1. What is the purpose of a View in an ASP.NET MVC application?

In an ASP.NET MVC application, the View is responsible for presenting the user interface to the user. It represents the visual presentation of data obtained from the Model. Views are typically comprised of HTML markup combined with embedded server-side code or markup syntax to dynamically render data. The primary purpose of the View is to display information to the user in a format that is understandable and interactive.

* 1. What is Separation of Concerns in ASP.NET MVC?

Separation of Concerns in ASP.NET MVC refers to the architectural principle of dividing a software application into distinct sections, each responsible for a specific aspect of functionality. In MVC, concerns are separated into three main components: Model (data and business logic), View (user interface), and Controller (application logic). This separation enhances maintainability, scalability, and testability of the application by isolating different aspects of functionality, making it easier to manage and modify them independently without affecting other parts of the system.

* 1. In your own words, elaborate what you understand about ASP.NET MVC naming convention.

ASP.NET MVC naming conventions refer to the guidelines and standards used to name various components within an MVC application, such as controllers, actions, views, models, and routes. These conventions help maintain consistency and clarity across the codebase, making it easier for developers to understand and navigate the application.

For example, controllers are typically named with a suffix of "Controller" (e.g., HomeController), actions within controllers are named descriptively to indicate their purpose (e.g., Index, Create, Edit), and views are named to correspond with their associated controller action (e.g., Index.cshtml, Create.cshtml).

Additionally, routing conventions dictate how URLs are mapped to controller actions, often following a pattern of /ControllerName/ActionName/parameters.

Overall, adhering to ASP.NET MVC naming conventions promotes readability, reduces confusion, and facilitates collaboration among developers working on the same project.

* 1. Briefly describe the ASP.NET MVC routing system.

The ASP.NET MVC routing system is responsible for mapping incoming browser requests to controller actions. It uses route tables to define patterns that match incoming URLs to specific controllers and actions within the application. These routes are typically defined in the RouteConfig.cs file within the App\_Start folder.

Routes consist of URL patterns and route parameters, such as controller and action names, along with optional parameters. When a request is received, the routing system matches the URL pattern to the appropriate route and extracts any route parameters. It then invokes the corresponding controller action, passing any extracted parameters as arguments.

This routing system allows for clean and flexible URL structures and enables developers to define custom routes to suit their application's needs.

* 1. Where are routes registered in ASP.NET MVC Application?

In an ASP.NET MVC application, routes are typically registered in the RouteConfig.cs file, which is located within the App\_Start folder. This file contains the configuration for the application's routing system, including the definition of route patterns and the corresponding controller and action mappings. By default, ASP.NET MVC applications use the RegisterRoutes method in the RouteConfig.cs file to define and register routes during application startup.

* 1. What is the purpose of a Razor View Engine in ASP.NET MVC application?

The purpose of the Razor View Engine in an ASP.NET MVC application is to facilitate the creation of dynamic and interactive user interfaces. Razor provides a syntax that combines HTML markup with server-side code (C# or VB.NET) to generate dynamic content. This allows developers to seamlessly integrate server-side logic with the presentation layer, making it easier to produce dynamic web pages and respond to user interactions. Razor views are highly expressive and allow for clean and concise code, enhancing the productivity of developers working on MVC applications.