- ## What is MVC?

  MVC stands for Model-View-Controller. It is a design pattern used for developing web applications. The main components are:

  - **Model:** Represents the data and business logic.
  - **View:** Responsible for displaying the data.
  - **Controller:** Handles user input and interactions, updating the model and selecting the view to render.

- ## How does routing work in ASP.NET MVC?

  Routing in ASP.NET MVC maps incoming HTTP requests to controller actions. It is configured in the RouteConfig.cs file, typically found in the App_Start folder. Routes are added to the RouteCollection and specify the URL pattern and the default controller, action, and parameters.

  Example:
  C#

```
public class RouteConfig
{
   public static void RegisterRoutes(RouteCollection routes)
   {
      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

      routes.MapRoute(
         name: "Default",
         url: "{controller}/{action}/{id}",
                 defaults: new { controller = "Home", action =
        "Index", id = UrlParameter.Optional }
      );
   }
}
```

- **What is the role of the controller in ASP.NET MVC?**

  The controller in ASP.NET MVC is responsible for handling user input and interactions. It processes incoming requests, performs operations on the model (such as retrieving data), and determines which view to render. Controllers contain action methods that map to user interactions.

  Example:

  C#

  ```
  public class HomeController : Controller

  {

     public ActionResult Index()

     {

                                           var model = new HomeModel {
              Message = "Welcome to ASP.NET MVC!"

      };

        return View(model);

     }

  }
  ```

- **How do you pass data from a controller to a view in ASP.NET MVC?**

  Data can be passed from a controller to a view using several methods:

  - **ViewBag:** A dynamic object for passing data.
  - **ViewData:** A dictionary object for passing data.
  - **Strongly-Typed Views:** Using a model class.

Example using ViewBag:

C#

```
public ActionResult Index()

{

    ViewBag.Message = "Hello, World!";

    return View();

}
```

Example using a strongly-typed view:

C#

```
public ActionResult Index()

{

    var model = new HomeModel { Message = "Hello, World!" };

    return View(model);

}
```

View

C#

```
@model HomeModel

<p>@Model.Message</p>
```

- **What is Razor in ASP.NET MVC?**

  Razor is a markup syntax used in ASP.NET MVC for embedding server-based code into web pages. It allows developers to write HTML and C# code in the same file, making it easy to render dynamic content.

  Example:

  Html

```
@model HomeModel

<!DOCTYPE html>

<html>

<head>

   <title>@Model.Title</title>

</head>

<body>

   <h1>@Model.Message</h1>

</body>

</html>
```

- **How do you handle errors in ASP.NET MVC?**

  Errors in ASP.NET MVC can be handled using several approaches:

  - **Try-Catch Blocks:** Handle exceptions in controller actions.
  - **Custom Error Pages:** Configure in Web.config to redirect users to custom error pages.
  - **Global Error Handling:** Use the Application_Error method in Global.asax or create a custom error handling attribute.

- **What is the purpose of a ViewModel in ASP.NET MVC?**

  A ViewModel in ASP.NET MVC is a class that represents the data and logic required for a view. It is used to pass data from the controller to the view, encapsulating multiple data models if necessary. ViewModels help to keep views clean and focused on presentation logic.

  Example:

  C#

  ```
  public class HomeViewModel

  {

      public string Title { get; set; }

      public string Message { get; set; }

  }
  ```

  Controller

  C#

  ```
  public ActionResult Index()

  {

      var model = new HomeViewModel { Title = "Home", Message =
      "Welcome to the Home Page" };

      return View(model);

  }
  ```

  View

  Html

  ```
  @model HomeViewModel

  <h1>@Model.Title</h1>
  ```

```
    <p>@Model.Message</p>
```