**1.Â Explain the Model, View, And Controller Components In .NET MVC And Their Responsibilities.**

The MVC architectural pattern in ASP.NET MVC separates web applications into three components:

* The “Model” part is the data and business logic of the .NET MVC web app
* The “View” is responsible for the presentation and the user interface design.
* The “Controller” is made to receive user input, interact with the “Model”, and update the “View” as required.

**2.Â How Does Routing Work In .NET MVC? Describe the Process of Defining and Handling Routes.**

Routing in .NET MVC usually maps URLs to the controller actions in applications. Routes can be defined in the RouteConfig file. The corresponding action method is executed automatically when a URL matches a route pattern.

**3.Â What Is Razor Syntax, And How Is It Used In .NET MVC Views?**

Razor syntax is the compact and expressive markup syntax used in .NET MVC views to seamlessly incorporate HTML and C# code. It allows developers to easily embed C# code within HTML markups using the @ symbol.

**4.Â Describe the Role of Models In .NET MVC And How They Interact with Views and Controllers.**

In .NET MVC, models represent the data and business logic of the application. They are used to encapsulate data, perform validation, as well as interacting with a connected database or other data sources used in a typical .NET MVC web app.

**5.Â How Do You Handle Form Submissions In .NET MVC? Explain the Use Of HTTP Verbs Like GET And POST.**

Form submissions in .NET MVC are handled via the [HttpPost attribute](https://learn.microsoft.com/en-us/answers/questions/497278/how-to-use-(httppost)-(httpget)-in-asp-net-core-we" \t "_blank)Â on the action method as well as the [ValidateAntiForgeryToken](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.validateantiforgerytokenattribute?view=aspnetcore-7.0" \t "_blank)Â attribute for CSRF protection. The form data is bound or linked to the model using model binding in ASP.NET MVC.

**6.Â What Is Data Validation In .NET MVC, And How Is It Implemented?**

Data validation in .NET MVC is normally implemented using attributes like “Required”, “StringLength”, “Range”, etc., added to model properties. Validation is designed to take place automatically during model binding. It can be checked using the “ModelState.IsValid” method.

**7.Â Discuss the Concept of Partial Views and Their Benefits In .NET MVC.**

Partial views enable .NET MVC developers to reuse and modularize “view” components. They can be rendered within other views or controllers via the @Html.Partial or @Html.RenderPartial methods.

**8.Â Explain the Concept of Viewbag And Viewdata In .NET MVC And Their Differences**

“ViewBag” and “ViewData” are special methods used to pass data from controllers to views in .NET MVC apps. By design, “ViewBag” uses dynamic properties, while ViewData utilizes a dictionary-like object. Both methods provide a way to transfer application data, but ViewBag is more convenient and less error-prone.

**9.Â How Do You Handle Authentication and Authorization In .NET MVC Applications?**

Authentication in .NET MVC is implemented using the built-in authentication solutions like Windows Authentication, Forms Authentication, or external auth providers like OAuth or OpenID Connect where needed.

**10.Â Discuss the Role of Filters In .NET MVC And Provide Examples of Their Usage**

In .NET, MVC filters, attributes or classes can be applied globally to specific controllers or actions. They are used to intercept requests and allow custom logic to be executed before, after, or around controller actions in MVC applications.

**11.Â Explain the Concept of Areas In .NET MVC And When You Would Use Them**

Areas in .NET MVC allow developers to logically group related controllers, views, and models into separate logical folders. They help organize large applications and promote the separation of concerns.

**12.Â How Do You Implement Caching In .NET MVC Applications to Improve Performance?**

Caching in .NET MVC can be implemented using the “[OutputCache](https://www.onestopdevshop.io/net-interview-questions/net-mvc-interview-questions/text=The%20output%20cache%20enables%20you,in%20a%20view%20named%20Index." \t "_blank)” attribute. This attribute caches the output of controller actions. As it does this, it improves app performance by serving cached content instead of re-executing the action unnecessarily.

**13.Â Describe the Process of Bundling and Minification Of CSS And JavaScript in .NET MVC**

Bundling and minification combine and compress multiple JS and CSS files into a single “minified” file. This reduces the number of requests made to the server, thus improving the .NET MVC web application performance.

**14.Â How Do You Handle Error Handling and Logging In .NET MVC Applications?**

In .NET MVC applications, error handling is done using the “[customErrors](https://www.tutorialsteacher.com/articles/web-config-customerrors-in-aspnet-mvc" \t "_blank)” element in the “web.config” file. It can also be done through handling exceptions in the “[Application\_Error](https://www.codeguru.com/dotnet/handling-errors-in-asp-net-mvc-applications/" \t "_blank)” event in the “Global.asax” file.

**15.Â What Is Entity Framework, And How Does It Facilitate Data Access In .NET MVC?**

The .NET MVC Entity Framework is an ORM tool that primarily simplifies database access in .NET MVC web applications. It was designed to provide a higher-level abstraction that allows developers to work with databases using C# objects and LINQ queries.

**16.Â Explain the Purpose of Migrations in Entity Framework and How They Are Used**

Migrations are used to manage database schema changes over time in the Entity Framework. They allow ASP.NET MVC developers to make updates to the database schema while preserving existing data.

**17.Â How Do You Integrate External Javascript Libraries Like Jquery In A .NET MVC Application?**

To integrate JavaScript libraries like jQuery in a .NET MVC application, devs can include the library’s script file in their project and reference it in the views as needed according to the design of their application.

**18.Â What Is the Purpose of Nuget in .NET Development, And How Do You Use It?**

NuGet is a package manager used across .NET, including the ASP.NET MVC framework. It simplifies the addition and management of external libraries and dependencies in a .NET MVC project.

**19.Â Describe the Process of Deploying A .NET MVC Application to A Web Server**

Deploying a .NET MVC application to a web server mainly involves publishing the application to a target folder and then transferring the published files to the web server using tools like FTP or Azure App Service deployment.

**20.Â How Do You Use Visual Studio for Debugging and Troubleshooting Issues In .NET MVC Applications?**

Visual Studio is equipped with various debugging features for .NET MVC, such as breakpoints, inspecting variables, stepping through code, and viewing call stacks. The Visual Studio debugger makes identifying and troubleshooting issues in a development environment easier.

**21.Â What Are Some Advantages of Using Git for Version Control In .NET MVC Development?**

Git is a popular version control system that can be used in .NET MVC development. It provides features like branching, merging, and version history, enabling dev teams to collaborate and manage source code efficiently without conflicts or disorganization.

**22.Â Explain the Concept of Continuous Integration (CI) And How It Can Be Achieved In .NET MVC Projects.**

Continuous integration (CI) is a standard application development practice where changes are frequently integrated into a shared repository. In .NET MVC, continuous integration (CI) can be achieved using tools like Azure DevOps Pipelines for automating the build, test, and deployment processes.

**23.Â Discuss the Role of Docker in .NET MVC Development and How It Facilitates Deployment.**

Docker is a popular platform primarily used for application containerization. In .NET MVC development, Docker packages applications and dependencies into containers for easy deployment and scalability.

**24.Â How Do You Use Azure DevOps (Or Other Similar Tools) For Project Management And CI/CD In .NET MVC?**

Azure DevOps is a set of development tools provided by Microsoft that offers features like version control, build pipelines project management, release management, and integration with Azure services, making it suitable for .NET MVC development and CI/CD workflows.

**25.Â What Are Some Benefits of Using A Task Runner Like Gulp or Grunt In .NET MVC Development?**

Visual Studio Code is a lightweight and extensible code editor widely used in .NET MVC development. It provides features like IntelliSense, debugging, and source control integration. It supports various programming languages and extensions.

**26.Â How Would You Use Postman for Testing and Debugging Web APIs in .NET MVC?**

Postman is a popular development tool used for testing and debugging APIs. It allows developers to send HTTP requests, inspect responses, and automate testing scenarios. [Postman](https://www.postman.com/api-platform/api-testing/" \t "_blank)Â is useful for testing Web APIs in .NET MVC applications.

**27.Â Explain the Purpose of Swagger (OpenAPI) In A .NET MVC Project And How It Can Be Used.**

Swagger (OpenAPI) is a specification and tooling ecosystem for designing, building, documenting, and consuming RESTful APIs. In .NET MVC, Swagger can automatically generate interactive API documentation, client SDKs, and server stubs, facilitating API development and integration.

**28.Â Discuss the Role of Automapper in .NET MVC And How It Simplifies Object-To-Object Mapping.**

AutoMapper is a library that simplifies object-to-object mapping in .NET MVC applications. It automatically maps properties between objects with similar names and structures, thus reducing the amount of manual mapping code and improving development productivity.

Load testing involves simulating high levels of user traffic to measure the performance and stability of a .NET MVC application. Developers use tools like [Apache JMeter](https://jmeter.apache.org/)Â or Microsoft’s [Load Testing Service](https://learn.microsoft.com/en-us/azure/load-testing/overview-what-is-azure-load-testing)Â to conduct these tests.

**29.Â How Do You Use Postsharp Or Other Aspect-Oriented Programming (AOP) Frameworks In .NET MVC?**

PostSharp is an aspect-oriented programming (AOP) framework for .NET that developers use to modularize cross-cutting concerns, such as caching, logging, and validation. They do this by applying attributes or aspects to target methods or classes in a .NET MVC application.

**30.Â Discuss the Benefits of Using A Front-End Framework Like React or Angular With .NET MVC.**

React, and Angular are popular front-end JavaScript frameworks that can be integrated with .NET MVC. They provide powerful tools for building dynamic and interactive user interfaces taking advantage of concepts like component-based architecture and virtual DOM diffing.

**31.Â What Are Some Advantages of Using Azure Functions in Conjunction With .NET MVC Applications?**

Azure Functions is a serverless computing platform developed and distributed by Microsoft. It allows developers to build and run small, event-driven code snippets (functions) that can be triggered by various events. It is suitable for microservices and serverless architectures in .NET MVC applications.

**32.Â Explain How You Would Integrate Oauth Or OpenID Connect for Authentication in A .NET MVC App.**

OAuth and OpenID Connect are authentication protocols that enable web application users to authenticate with third-party identity providers (such as Google, Facebook, or Azure AD). This enables them to access resources in a .NET MVC application securely.

**33.Â How Do You Use Application Insights or Other Monitoring Tools to Track Application Performance?**

Application Insights is a monitoring and diagnostics service provided by Microsoft. It is usually integrated into a .NET MVC application to collect telemetry data, including exception logs, performance metrics, and request traces. This helps developers identify and troubleshoot issues.

**34.Â Discuss the Purpose and Benefits of Using Redis Or Other Caching Mechanisms In .NET MVC.**

Redis is an in-memory data structure store used as a caching layer in .NET MVC applications. It provides fast read and write access, thus allowing frequently accessed data to be stored and retrieved quickly to improve application performance.

**35.Â What Are Some Strategies for Optimizing Database Performance in A .NET MVC Application?**

MiniProfiler is a lightweight profiling tool for .NET applications, including .NET MVC apps. It allows developers to measure the performance of specific code sections and identify bottlenecks. The tool provides insights for optimization and performance tuning.

**36.Â Discuss the Purpose and Usage of The Health Checks Feature In .NET MVC**

Health checks in .NET MVC enable developers to monitor the health and availability of their application’s dependencies, such as databases, external services, or system resources. It can help MVC devs detect issues early and also provides insights into the overall health of the application.

**37.Â How Do You Handle Cross-Site Scripting (XSS) Attacks in A .NET MVC Application?**

To handle cross-site scripting (XSS) attacks in a .NET MVC application, developers can sanitize user input by encoding it before displaying it in views. The ASP.NET MVC framework provides built-in features like request validation and output encoding to prevent XSS attacks.

**38.Â Discuss the Role of Action Filters In .NET MVC and Provide Examples of their usage**

Action Filters in .NET MVC are attributes that enable devs to add custom behavior to controller actions or globally to all actions in the application. They are primarily used to perform tasks such as logging, authentication, authorization, or modifying the action’s behavior before or after execution.

**39.Â How Do You Implement Role-Based Authorization in A .NET MVC Application?**

In a .NET MVC application, role-based authorization can be implemented using the built-in ASP.NET Identity system or custom role providers. MVC developers can define roles, assign users to roles, and then apply role-based authorization using attributes like “Authorize (Roles = “RoleName”).”

**40.Â Describe the Process of Implementing Custom Model Binders In .NET MVC**

Custom model binders in .NET MVC allow devs to control how data from HTTP requests is bound to model objects. To implement custom model binders, a developer can create a class that inherits from the ModelBinder base class and override its methods to handle the binding logic.

**41.Â Discuss the Purpose and Usage of The Antiforgerytoken in .NET MVC For CSRF Protection**

The AntiForgeryToken in .NET MVC is a security feature used for protection against Cross-Site Request Forgery (CSRF) attacks.

**42.Â Explain the Concept of Asynchronous Controllers In .NET MVC And Their Benefits**

Asynchronous controllers in .NET MVC allow developers to handle asynchronous operations, such as making external API calls or querying databases without blocking the request thread.

**43.Â How Do You Implement Custom Route Constraints In .NET MVC?**

Custom route constraints in .NET MVC allow developers to define additional conditions for route matching. One can create a class that implements the “[IRouteConstraint](https://learn.microsoft.com/en-us/dotnet/api/system.web.routing.irouteconstraint?view=netframework-4.8.1)” interface and define the logic for evaluating whether a specific route parameter value matches the constraint.

**44.Â How Do You Handle Concurrency Conflicts in A Multi-User .NET MVC Application?**

Concurrency conflicts in a multi-user .NET MVC application can be handled by using optimistic concurrency control techniques. This involves including a version number or timestamp column in the database table representing the entity being modified.

**45.Â Explain the Concept of Output Caching In .NET MVC And Its Impact on Performance**

Output caching in .NET MVC allows MVC developers to cache the output of controller actions or partial views, reducing the need for repeated computations or database queries.

**46.Â Discuss the Benefits of Using Dependency Injection Containers Like Autofac Or Unity In .NET MVC**

Dependency injection containers like[Â Autofac](https://docs.autofac.org/en/stable/integration/aspnetcore.html)Â or [Unity](https://www.onestopdevshop.io/net-interview-questions/net-mvc-interview-questions/text=Dependency%20Injection%20is%20defined%20as,and%20Dependency%20Injection%20is%20implementation%E2%80%9D." \t "_blank)Â are used in .NET MVC to help with the implementation of the dependency injection pattern. These containers manage the creation and lifetime of objects and resolve dependencies between different components of the applications.

**47.Â How Do You Handle File Downloads In .NET MVC Applications?**

File downloads in .NET MVC applications can be handled by returning a “FileResult” from a controller action. The FileResult represents the file to be downloaded, and the developer can specify the file’s content, content type, and file name as needed.

**48.Â Describe the purpose and usage of the @helper syntax in Razor views**

The @helper syntax is used to define reusable helper functions that can be invoked within the view. It provides a way to encapsulate commonly used markup or logic into a helper method. Doing this improves code organization and reduces duplication.

**49.Â What are the different types of ActionResult in .NET MVC, and when would you use each?**

ActionResult is the base class for all result types returned from controller actions in .NET MVC. The different types of [ActionResult](https://www.onestopdevshop.io/net-interview-questions/net-mvc-interview-questions/text=An%20action%20result%20is%20what,redirection%20to%20a%20new%20URL." \t "_blank)Â are ViewResult (renders a view), [PartialViewResult](https://www.c-sharpcorner.com/UploadFile/0c1bb2/partialviewresult-type-in-mvc-5/" \t "_blank)Â (renders a partial view), JsonResult (returns JSON data), RedirectResult (redirects to a specified URL), and FileResult (returns a file for download). .

**50.Â How would you use Docker Compose to orchestrate multiple containers in a .NET MVC application?**

Docker Compose is used to orchestrate multiple containers in a .NET MVC application. Used correctly, it allows one to define and manage a multi-container application using a YAML file. With Docker Compose, one can specify the services, their dependencies, network configurations, and volumes required for the application to run.

## **Q1. What is MVC?**

MVC stands for [Model-View-Controller.](https://www.scholarhat.com/tutorial/mvc) It is a software design pattern that was introduced in the 1970s. Also, the MVC pattern forces a separation of concerns, which means the domain model and controller logic are decoupled from the user interface (view). As a result, maintenance and testing of the application become simpler and easier.

MVC framework generally works by separating the business logic and presentation layer from each other and by doing it, the accessibilities may enhance. It was traditionally used for the desktop graphical user interfaces previously but, nowadays, the MVC architectural pattern in web technology has become popular and evolved for the designing & architecting the web applications as well as mobile apps while having a robust application structure.

## **Q2. What is ASP.NET MVC?**

[ASP.NET MVC](https://www.scholarhat.com/tutorial/mvc) is an open-source framework built on the top of the Microsoft .NET Framework to develop a web application that enables a clean separation of code. ASP.NET MVC framework is the most customizable and extensible platform shipped by Microsoft.

MVC architectural pattern is related to the ASP.NET in an almost similar way as it is related to an application. the ASP.NET is integrated into the [HTML](https://www.scholarhat.com/tutorial/html)template that is used to create or develop most of the webpages we see on the Internet. In other words, we can say that the view layer of the MVC. ASP.NET is executed on the remote server which is why it's called a server-side framework which is the controller layer of the MVC framework.

## **Q3. What are the advantages of ASP.NET MVC?**

There are the following advantages of ASP.NET MVC over Web Forms (ASP.NET):

* **Separation of concern:** MVC design pattern divides the ASP.NET MVC application into three main aspects Model, View, and Controller which make it easier to manage the application complexity.
* **TDD:** The MVC framework brings better support to test-driven development. Using the TDD, writing clean and robust code which is structured along with well commented as well as testing your code. This is where the ASP.Net MVC puts a special emphasis on testing the all development activities while doing it. TDD approaches the further capabilities of testing the individual component during the software development process and ensures the highest quality development activity for the project.
* **Extensible and pluggable:** MVC framework components were designed to be pluggable and extensible and therefore can be replaced or customized easier than Web Forms.
* **Full control over application behavior:** MVC framework doesn’t use View State or server-based forms like Web Forms. This gives the application developer more control over the behaviors of the application and also reduces the bandwidth of requests to the server.
* **ASP.NET features are supported**MVC framework is built on top of ASP.NET and therefore can use most of the features that ASP.NET includes such as the provider's architecture, authentication, authorization scenarios, membership, and roles, caching session, and more.
* **URL routing mechanism:** MVC framework supports a powerful URL routing mechanism that helps to build a more comprehensible and searchable URL in your application. This mechanism helps the application to be more addressable in the eyes of search engines and clients and can help in search engine optimization.

## **Q4.What is Routing in ASP.NET MVC?**

[Routing](https://www.scholarhat.com/tutorial/mvc/routing-in-aspnet-mvc-with-example) is a pattern matching system that monitors the incoming request and figures out what to do with that request. At runtime, the Routing engine uses the Route table for matching the incoming request's URL pattern against the URL patterns defined in the Route table. You can register one or more URL patterns to the Route table at the Application\_Start event. When the routing engine finds a match in the routing table for the incoming request's URL, it forwards the request to the appropriate controller and action. If there is no match in the routing table for the incoming request's URL, it returns a 404 HTTP status code.

## **Q5. When to use Attribute Routing?**

The convention-based routing is complex to support certain URI patterns that are common in RESTful APIs. But by using [attribute routing](https://www.scholarhat.com/tutorial/mvc/understanding-attribute-routing-in-aspnet-mvc) you can define these URI patterns very easily.

**For example,**resources often contain child resources like Clients have ordered, movies have actors, books have authors, and so on. It’s natural to create URIs that reflect these relations like as /clients/1/orders

This type of URI is difficult to create using convention-based routing. Although it can be done, the results don’t scale well if you have many controllers or resource types. With attribute routing, it’s pretty much easy to define a route for this URI. You simply add an attribute to the controller action as:

[](https://www.scholarhat.com/training/aspnet-core-certification-training?utm_source=banner&utm_medium=display&utm_campaign=https://www.scholarhat.com/tutorial/mvc/mvc-interview-questions-answers)

[Route("clients/{clientId}/orders")]

**public** IEnumerable<Order> **GetOrdersByClient**(int clientId)

{

//TO DO

}

## **Q6. How to enable Attribute Routing in ASP.NET MVC?**

Enabling attribute routing in your ASP.NET MVC5 application is simple, just add a call to routes.MapMvcAttributeRoutes() method within RegisterRoutes() method of RouteConfig.cs file.

## **Q7. What is the difference between Routing and URL Rewriting?**

Many developers compare routing to URL rewriting since both look similar and can be used to make SEO friendly URLs. But both the approaches are very much different. The main difference between routing and URL rewriting is given below :

* URL rewriting is focused on mapping one URL (new URL) to another URL (old URL) while routing is focused on mapping a URL to a resource.
* URL rewriting rewrites your old URL to a new one while routing never rewrite your old URL to the new one but it maps to the original route.

## **Q8. What is View Engine?**

A [View Engine](https://www.scholarhat.com/tutorial/mvc/removing-the-web-form-view-engine-for-better-performance-of-razor-view-engine) is an MVC subsystem that has its own markup syntax. It is responsible for converting the server-side template into HTML markup and rendering it to the browser. Initially, ASP.NET MVC ships with one view engine, web forms (ASPX), and from ASP.NET MVC3 a new view engine, Razor is introduced. With ASP.NET MVC, you can also use other view engines like Spark, NHaml, etc.

The view engine helps to render the views into HTML form in the browser. By default, the Asp.net MVC provides two different flavors of the view engines such as ASPX and the Razor View Engine.

## **Q9. What is Razor View Engine?**

[Razor Engine](https://www.scholarhat.com/tutorial/mvc/removing-the-web-form-view-engine-for-better-performance-of-razor-view-engine) is an advanced view engine that was introduced with MVC3. This is not a new language but it is a new markup syntax. Razor has a new and advanced syntax that is compact, expressive, and reduces typing. Razor syntax is easy to learn and much cleaner than Web Form syntax. Razor uses @ symbol to write markup as:

@Html.ActionLink("SignUp", "SignUp")

## **Q10. What are HTML Helpers in ASP.NET MVC?**

An [HTML Helper](https://www.scholarhat.com/tutorial/mvc/understanding-html-helpers-in-aspnet-mvc) is just a method that returns an HTML string. The string can represent any type of content that you want. For example, you can use HTML Helpers to render standard HTML tags like HTML <input>, <button> and <img> tags etc. You can also create your own HTML Helpers to render more complex content such as a menu strip or an HTML table for displaying database data.

The HtmlHelper is a class that is used to render the HTML controls into the razor view pages. Helper binds the model object to the HTML controls of the templates to display the value of the respective model properties into those controls as well as it also assigns the value of the controls to the model properties while the user submits the webform while interacting with the form elements. Its always recommended to always use the Html Helpers classes in the razor view instead of writing all the HTML tags manually.

## **Q11. What are Url Helpers?**

Url helpers allow you to render HTML links and raw URLs. The output of these helpers is dependent on the routing configuration of your ASP.NET MVC application.

## **Q12. When to use \_ViewStart?**

When a set of views shares common settings, the \_ViewStart.cshtml file is a great place to put these common view settings. If any view needs to override any of the common settings then that view can set new values in common settings.

## **Q13. Can you change the action method name?**

You can also change the action method name by using the ActionName attribute. Now the action method will be called by the name defined by the ActionName attribute.

[ActionName("DoAction")]

**public** ActionResult **DoSomething**()

{

//**TODO:**

**return** View();

}

Now, the DoSomething action will be identified and called by the name DoAction.

## **Q14. What are Data Annotations in ASP.NET MVC?**

[Data validation](https://www.scholarhat.com/tutorial/mvc/mvc-data-annotations-for-model-validation) is a key aspect of developing a web application. In Asp.net MVC, we can easily apply validation to the web application by using Data Annotation attribute classes to the model class. [Data Annotation](https://www.scholarhat.com/tutorial/mvc/mvc-data-annotations-for-model-validation) attribute classes are present in System.ComponentModel. DataAnnotations namespace and are available to Asp.net projects like Asp.net web application & website, Asp.net MVC, Web forms, and also to Entity framework ORM models. Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end-users.

## **Q15. How to determine there is no error in Model State?**

When server-side model validation fails, errors are included in the ModelState. Hence, by using ModelState.IsValid property you can verify model state. It returns true if there is no error in ModelState else returns false.

[HttpPost]

**public** ActionResult **DoSomething**(UserViewModel model)

{

**if** (ModelState.IsValid)

{

//**TODO:**

}

**return** View();

}

## **Q16. What is the jQuery Validation Unobtrusive plugin?**

Microsoft introduced the jquery.validate.unobtrusive.js plugin with ASP.NET MVC3 to apply data model validations to the client side using a combination of jQuery Validation and HTML 5 data attributes.

## **Q17. Can we use Bundling and Minification in ASP.NET MVC3 or ASP.NET4.0?**

System.Web.The optimization class offers the [bundling and minification](https://www.scholarhat.com/tutorial/mvc/bundling-and-minification-in-mvc3-and-aspnet-40) techniques that exist within Microsoft.Web.Optimization dll. Using this dll you can also use this technique with ASP.NET MVC3 and ASP.NET 4.0.

## **Q18. What is Scaffolding?**

[Scaffolding](https://www.scholarhat.com/tutorial/mvc/understanding-aspnet-mvc-scaffolding) is a technique used by many MVC frameworks like ASP.NET MVC, Ruby on Rails, Cake PHP and Node.JS, etc., to generate code for basic CRUD (create, read, update, and delete) operations against your database effectively. Further, you can edit or customize this auto-generated code according to your need. Scaffolding consists of page templates, entity page templates, field page templates, and filter templates. These templates are called Scaffold templates and allow you to quickly build a functional data-driven Web site.

## **Q19. How to persist data in TempData?**

The life of [TempData](https://www.scholarhat.com/tutorial/mvc/persisting-data-with-tempdata" \t "_blank) is very short and lies only until the target view is fully loaded. But you can persist data in TempData by calling the Keep() method after request completion

## **Q20. Can you change the action method name?**

You can also change the action method name by using the ActionName attribute. Now the action method will be called by the name defined by the ActionName attribute.

[ActionName("DoAction")]

**public** ActionResult **DoSomething**()

{

//**TODO:**

**return** View();

}

Now, the DoSomething action will be identified and called by the name DoAction.

## **Q21. Mention the types of results in MVC?**

In MVC, a total of 12 types of results are found. "ActionResult" class is the main class whereas the 11 are their sub-types. These types are covered when you **learn MVC step by step.**Here is the list of these sub-types:

* ViewResult
* EmptyResult
* PartialViewResult
* RedirectResult
* RedirectToRouteResult
* JavaScriptResult
* JsonResult
* FileContentResult
* FileStreamResult
* FilePathResult
* ContentResult

## **Q22. What do the 3 logic layers define about the MVC Pattern?**

The MVC model defines the web applications through 3 logic layers:

1. The business layer (Model logic)
2. The display layer (View logic)
3. The input control (Controller logic)

The Model logic is the portion of the application that only deals with the logic for the application data. Commonly, the model objects to access data and even store data from a database.

The View logic is the portion of the application that looks after the display of the data. Frequently, the model data create views. Some other complex methods of creating views are available.

The Controller logic is the portion of the application that deals with user interaction.

## **Q23. What does Database first approach in MVC through Entity Framework?**

Database First Approach works as a substitute to the Code First as well as Model First approaches to the Entity Data Model. Moreover, the Entity Data Model makes model codes including properties, classes, DbContext, etc. from the database within the project. The particular class works as the link between the controller and the database. When you undergo [**MVC Certification training,**](https://www.scholarhat.com/course/aspnet-mvc-webapi-course)you may be made familiar with the Database first approach.

The following ways are useful to connect the database with the application.

* Database First
* Code First
* Model First

## **Q24. Explain GET and POST Action types:**

Both these action types are described in the[MVC tutorial](https://www.scholarhat.com/tutorial/mvc)**.**

GET Action Type:

GET is employed to request data from a specific resource. Through all the GET requests, the URL is passed. The process is mandatory and it can take up a few overloads.

POST Action Type: The POST Action type is useful to submit data that needs to be processed to a specific resource. Through all the POST requests, the URL is passed that is essential and the data.

## **Q25. How to execute validation in MVC?**

It is easy to execute validation in MVC applications through the validators defined in theSystem.ComponentModel.DataAnnotationsnamespace. Various types of validators are as follows:

* Required
* Range
* DataType
* StringLength

## **Q26. How do Views and Partial Views differ?**

The view contains the layout page. Before rendering any view, the view start page is rendered. Moreover, a view may have markup tags such as HTML, body, title, head, and meta, etc.

Partial View does not contain the layout page. It does not validate for a viewstart.cshtml. It is not allowed to place common code for a partial view inside the view start.cshtml.page. A partial view is specially designed to render inside the view and therefore, it does not include any markup. It is allowed to pass a regular view to the RenderPartial method.

## **Q27. What basic folders use the MVC template without Areas in the ASP.NET Core project?**

The following folders make use of the MVC template without Areas:

* Controllers- Default folder for application controllers
* Views- It is a folder containing a folder for each controller as well as a particular folder shared for views utilized by multiple views or controllers.
* wwwroot- It is a publicly accessible folder of a site comprising of subfolders of static files

## **Q28. Why is WebAPI technology introduced?**

Formerly, HTTP was used as a protocol for all types of clients. Gradually, the client variety began to increase and extend in various directions. The widespread use of Windows applications, JavaScript, and mobile demanded the huge consumption of HTTP. Therefore, the REST approach was introduced. This is the reason why WebAPI technology is implemented to the REST principles to render the data over HTTP.

Web API as the name suggests is an API over the web that can be used using the HTTP protocol. The web API is a concept and it should not be considered technology. By using the web API, we can build the Web APIs using different technologies such as Java, .NET, NodeJs, Python, etc. For example, The student's record details can be used by the universities and university details can be consumed via the Web API URLs in any of the web applications.

## **Q29. What steps need to be followed when you have an ASP.NET Core MVC application wherein you have to use some cache solution as well as support running across multiple servers?**

The ASP.NET Core is a cutting-edge platform and it supports various approaches. The user needs to use certain types of distributed cache, for example, Redis. Moreover, Microsoft offers various packages to assist you with this. When using Redis, Microsoft.Extensions.Caching.Redis offers the middleware and employs IDistributedCache to present a standard approach to function with it.

## **Q30. Discuss the vital namespaces used in ASP.NET MVC?**

For an [MVC developer](https://www.scholarhat.com/course/aspnet-mvc-webapi-course)**,**it is better to know the significant namespaces used in the ASP.NET MVC application. Here are they:

1. System.Web.Mvc:

It comprises interfaces and classes which support the MVC pattern for ASP.NET Web applications. Moreover, it contains classes that depict controller factories, controllers, views, partial views, auction results, and model binders.

1. System.Web.Mvc.Ajax:

It includes classes that support Ajax scripting within an ASP.NET MVC application.

1. System.Web.Mvc.Html:

It includes classes that facilitate the rendering of HTML controls in an MVC application. It contains those classes that support input controls, forms, partial views, links, and validation.

## **Q31. How does ASP.NET MVC differ from Web Forms?**

|  |  |
| --- | --- |
| **ASP.NET MVC** | **Web Forms** |
| The architecture is based on Model-View-Controller pattern. | It follows the event-driven programming model. |
| It gives more control to the developers over generating HTML. | It provides comparatively less control over the HTML output. |
| It focuses more on separation of concerns among the layers. | It has tightly coupled code that can mix up the concerns. |

## **Q32. Explain the Model-View-Controller (MVC) architectural pattern in ASP.NET MVC?**

The Model-View-Controller architectural pattern consists of three components:

1. **Model-**It consists of the data and business logic of the application and also the data that is stored and retrieved from the database.
2. **View-**It consists of the application's User Interface where all the data is displayed and the user input is handled.
3. **Controller-**It works like a bridge between the model and the view where it takes the user input and returns the processed data back to the user as view.

## **Q33. Describe the components of an ASP.NET MVC application?**

There are basically three components of an ASP.NET MVC application that are:

* **Models-**The data and business logic of the application are present here.
* **Views-**The User Interface of the application comes under views.
* **Controllers-**User requests are handled with the help of Controllers.

## **Q34. What are the different types of ActionResult in ASP.NET MVC?**

There are several types of ActionResult in ASP.NET MVC such as:

1. ViewResult
2. PartialViewResult
3. RedirectResult
4. RedirectToRouteResult
5. JsonResult
6. ContentResult
7. FileResult

## **Q35. What is the role of TempData in ASP.NET MVC?**

In ASP.NET MVC, the TempData is used to temporarily store data when an HTTP request is under process and pass the data from one controller to another controller during that process.

## **Q36. Explain the concept of Action Filters in ASP.NET MVC?**

Whenever an action method is being executed, a certain set of attributes are applied to the controller actions so that they can add on some processing logic to the execution, whether it be before or after the process. These attributes are known as Action [Filters](https://www.scholarhat.com/tutorial/mvc/understanding-aspnet-mvc-filters-and-attributes)in ASP.NET MVC.

## **Q37. How does TempData differ from ViewBag and ViewData?**

* [TempData](https://www.scholarhat.com/tutorial/mvc/viewdata-vs-viewbag-vs-tempdata-vs-session)and ViewData are both dictionary objects and both of them need typecasting while ViewBag is a dynamic property that does not need any typecasting to pass data.
* TempData temporarily stores data for the next HTTP request while ViewBag and ViewData passes the data within the same request only.

## **Q38. What are the different types of routing constraints available in ASP.NET MVC?**

The different types of [routing constraints](https://www.scholarhat.com/tutorial/mvc/route-constraints-in-aspnet-mvc-with-example) that are available in ASP.NET MVC are as follows:

1. Regular Expression Constraint
2. Range Constraint
3. Length Constraint
4. Min/Max Constraint
5. Compound Constraint
6. Custom Constraint

## **Q39. Discuss the usage of Partial Views in ASP.NET MVC applications.**

In ASP.NET MVC, [partial views](https://www.scholarhat.com/tutorial/mvc/partial-view-in-aspnet-mvc3-razor) are basically small sections of the complete parent view which help the developers to manage the complex UI by breaking them down into small parts like mini views. Moreover, partial views are also reusable across multiple views and they provide features like modularity, encapsulation and dynamic data binding.

## **Q40. What is Dependency Injection, and how is it used in ASP.NET MVC applications?**

Dependency Injection can be defined as those loose coupling design pattern which gives the freedom to not create dependencies of a class within the class but instead they are injected from the outside.

Dependency Injection are very commonly used in ASP.NET MVC application DI has built in support in ASP.NET Core for injecting dependencies like services into the views, controllers and many components.

## **Q41. Explain the difference between Html.RenderPartial() and Html.Partial() methods in ASP.NET MVC.**

|  |  |
| --- | --- |
| **Html.RenderPartial()** | **Html.Partial()** |
| This is used to render the partial view to the output directly. | This is used to store the content which is already rendered in a variable. |
| There is nothing to return. | A string that has the partial view displayed in it is returned. |
| This is mostly used when there is no need for changes to the content and can be directly rendered. | This is mostly used when you want changes to be made to the content before rendering it. |

## **Q42. How does Model Binding work in ASP.NET MVC?**

Model Binding in ASP.NET MVC is a process where data of HTTP request is mapped onto the model properties or action method parameters. The raw data of the HTTP request is complex to understand so model binding enables converting them into readable .NET objects that are strongly typed.

## **Q43. Describe the concept of Areas in ASP.NET MVC?**

The concept of Areas in ASP.NET MVC can be referred to as separate modules which contain logical group of controllers, views and other assets. These are different small parts of a web application which makes managing the application less complicated and more organizable.

## **Q44. How do you handle errors and exceptions in ASP.NET MVC applications?**

To [handle errors and exceptions](https://www.scholarhat.com/tutorial/mvc/exception-or-error-handling-and-logging-in-mvc4) in ASP.NET MVC applications, we can use:

* try-catch blocks in controllers
* global error handling mechanisms like custom error pages
* frameworks like Serilog, Elmah
* 'HandleError' attribute in controller

## **Q45. Discuss the various ways to implement client-side validation in ASP.NET MVC applications.**

The various ways to implement [client-side validation](https://www.scholarhat.com/tutorial/mvc/how-to-enable-and-disable-client-side-validation-in-mvc) in ASP.NET MVC applications are:

1. **HTML5 Validation-**'required', 'min', 'max', 'pattern', etc are some of the HTML5 attributes that can be used to implement client side validation.
2. **jQuery Validation Plugin-**We can use jQuery for integrating jQuery Validation Plugin that provides extensive validation rules and customizable error messages.
3. **Custom JavaScript Validation-**We can also manually write JavaScript code so that form inputs can be validated through customized code.
4. **Third-party Libraries-**There are some third party JavaScript libraries such as Validate.js, Parsley.js that allow implementing client-side validation.
5. **Built-in Client-Side Validation-**ASP.NET Core has built-in client-side validation as well through 'jQuery-validation-unobtrusive' package.

## **Q46. How can you handle multiple submit buttons in a single ASP.NET MVC form?**

To handle multiple submit buttons in a single ASP.NET MVC form, what you can do is for each button used in the form, give it a unique name or another method is using JavaScript or jQuery to see which button was clicked and the correct action can be performed after clicking.

## **Q47. What is the purpose of the Web.config file in an ASP.NET MVC application?**

In an ASP.NET MVC application, a Web.config file is used for the purpose of configuring the settings that are in that specific application. They can be database connections, security configurations, application wide settings or other parameters which are required so that the application can run in the correct manner as it should.

## **Q48. How do you handle file uploads in ASP.NET MVC?**

In ASP.NET MVC, you can easily handle file uploads using the 'HttpPostedFileBase' class that will help you to access the file data that has been uploaded. The file can be saved later or if you want process it however you want.

## **Q49. What are the advantages of using asynchronous controllers in ASP.NET MVC?**

The advantages of using asynchronous controllers in ASP.NET MVC are as follows:

* It reduces blocking and waiting time to load the pages faster.
* It allows to free up the threads and there is more time for handling other requests while I/O operations are going on.
* It doesn't block the thread during I/O operations and application becomes more responsive.
* It handles long running operations like network requests or database queries very well.
* It gives the server the power to handle more concurrent requests.

## **Q50. What is the purpose of the Layout property in Razor views?**

The main purpose of the Layout property in Razor views is specifying a shared layout page which will have some common HTML structure and the elements that will help to achieve multiple views in an ASP.NET Core MVC application.

**1. Describe the MVC architecture and explain how it works.**

The Model-View-Controller (MVC) architecture is an architectural pattern used in software engineering to separate the application logic from the user interface. It divides an application into three interconnected parts, which are the Model, the View, and the Controller.  
  
The Model is responsible for managing the data of the application. It contains the business logic and handles the data access layer. It is the core of the application and is independent of the user interface.  
  
The View is responsible for displaying the data to the user. It is the user interface of the application and is responsible for displaying the data to the user in a format that is easy to understand.  
  
The Controller is responsible for handling the user input. It is the middleman between the Model and the View. It receives the user input, processes it, and then passes it to the Model or the View.  
  
The MVC architecture is beneficial because it allows for the separation of concerns. The Model is responsible for the data, the View is responsible for the user interface, and the Controller is responsible for handling the user input. This separation of concerns makes it easier to maintain and extend the application. Additionally, it allows for the reuse of code, as the same Model can be used in multiple Views.

**2. What is the purpose of the Model-View-Controller (MVC) pattern?**

The Model-View-Controller (MVC) pattern is an architectural software design pattern that separates an application into three main components: the model, the view, and the controller. The purpose of the MVC pattern is to separate the application logic from the user interface, allowing for easier maintenance and scalability.  
  
The model is responsible for managing the data of the application. It is the core component of the MVC pattern and is responsible for maintaining the data and business logic of the application.  
  
The view is responsible for displaying the data to the user. It is the user interface of the application and is responsible for displaying the data to the user in a format that is easy to understand.  
  
The controller is responsible for handling user input and updating the model and view accordingly. It is the middleman between the model and the view and is responsible for handling user input and updating the model and view accordingly.  
  
The MVC pattern is a great way to structure an application as it allows for easier maintenance and scalability. It also allows for better separation of concerns, as the model, view, and controller are all separate components. This makes it easier to maintain and debug the application.

**3. What are the benefits of using the MVC pattern?**

The Model-View-Controller (MVC) pattern is a popular software design pattern used to create applications that separate the application’s data, business logic, and user interface. This separation of concerns makes it easier to maintain and extend the application.  
  
The main benefits of using the MVC pattern are:  
  
1. Separation of Concerns: MVC separates the application into three distinct components: the model, the view, and the controller. This separation of concerns makes it easier to maintain and extend the application. The model handles the data, the view handles the user interface, and the controller handles the business logic. This separation of concerns makes it easier to debug and test the application.  
  
2. Reusability: The MVC pattern allows developers to reuse code and components. This makes it easier to create new applications and features.  
  
3. Flexibility: The MVC pattern allows developers to easily modify the application’s user interface without having to change the underlying business logic. This makes it easier to make changes to the application without having to rewrite the entire application.  
  
4. Scalability: The MVC pattern makes it easier to scale an application. The separation of concerns makes it easier to add new features and components without having to rewrite the entire application.  
  
5. Testability: The MVC pattern makes it easier to test an application. The separation of concerns makes it easier to test each component of the application separately. This makes it easier to identify and fix bugs.

**4. How do you handle authentication and authorization in an MVC application?**

Authentication and authorization are two important aspects of any MVC application. Authentication is the process of verifying the identity of a user, while authorization is the process of determining what a user is allowed to do.  
  
To handle authentication and authorization in an MVC application, I typically use the ASP.NET Identity framework. This framework provides a secure and extensible way to manage user accounts and roles. It also provides a number of features such as password reset, two-factor authentication, and account lockout.  
  
To implement authentication and authorization, I first create a database to store user accounts and roles. I then create a custom user store that implements the ASP.NET Identity interfaces. This user store is responsible for managing user accounts and roles.  
  
Next, I create a custom authentication manager that implements the ASP.NET Identity interfaces. This authentication manager is responsible for verifying user credentials and issuing authentication tokens.  
  
Finally, I create a custom authorization manager that implements the ASP.NET Identity interfaces. This authorization manager is responsible for determining what a user is allowed to do.  
  
By using the ASP.NET Identity framework, I can easily implement authentication and authorization in an MVC application.

**5. What is the difference between a View and a Partial View in MVC?**

The main difference between a View and a Partial View in MVC is that a View is a complete page, while a Partial View is a reusable component that can be embedded in multiple pages.  
  
A View is a page that is rendered by the controller and is composed of HTML, CSS, and JavaScript. It is the main page that the user interacts with and is responsible for displaying the content of the application.  
  
A Partial View is a reusable component that can be embedded in multiple pages. It is a reusable piece of HTML code that can be used to render a portion of the page. It is typically used to render a portion of the page that is common across multiple pages, such as a header or footer. Partial Views are typically rendered by the controller and can be passed data from the controller.  
  
In summary, a View is a complete page, while a Partial View is a reusable component that can be embedded in multiple pages.

**6. How do you handle routing in an MVC application?**

Routing in an MVC application is handled by the routing engine. The routing engine is responsible for mapping incoming requests to the appropriate controller and action. It is also responsible for generating URLs for links and redirects.  
  
The routing engine uses a set of rules defined in the application's configuration file (usually called routes.config) to determine which controller and action should be invoked for a given request. The rules are defined using a combination of URL patterns and HTTP verbs.  
  
When a request is received, the routing engine parses the URL and matches it against the defined rules. If a match is found, the appropriate controller and action are invoked. If no match is found, the routing engine will return a 404 error.  
  
The routing engine also provides a way to generate URLs for links and redirects. This is done by passing a set of parameters to the routing engine, which will then generate the appropriate URL. This allows for more flexibility when creating links and redirects, as the URLs can be generated dynamically based on the parameters passed.  
  
In summary, the routing engine is responsible for mapping incoming requests to the appropriate controller and action, as well as generating URLs for links and redirects. It is an essential part of any MVC application.

**7. What is the purpose of the ViewBag in MVC?**

The ViewBag is a dynamic object in ASP.NET MVC that allows you to pass data from the controller to the view. It is a wrapper around the ViewData dictionary and provides a convenient way to access data stored in the ViewData dictionary. The ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0. It provides a way to store and retrieve values using a dynamic property syntax, rather than the traditional dictionary syntax.  
  
The ViewBag is useful for passing data from the controller to the view without having to explicitly define the data type. This makes it easier to pass data from the controller to the view without having to create a strongly-typed view model. It also allows you to pass data from the controller to the view without having to explicitly define the data type.  
  
The ViewBag is also useful for passing data from the view to the controller. This is done by using the ViewBag.SetValue() method. This allows you to pass data from the view to the controller without having to explicitly define the data type.  
  
Overall, the ViewBag is a powerful tool for passing data between the controller and the view in ASP.NET MVC. It provides a convenient way to access data stored in the ViewData dictionary and allows you to pass data from the controller to the view without having to explicitly define the data type.

**8. How do you handle data validation in an MVC application?**

Data validation in an MVC application is an important part of ensuring the integrity of the data that is being stored and used. There are several ways to handle data validation in an MVC application.  
  
The first way is to use the built-in validation features of the MVC framework. This includes using the ModelState object to validate user input, using the DataAnnotations attributes to validate model properties, and using the ValidationAttribute class to create custom validation rules.  
  
The second way is to use a third-party library such as FluentValidation or DataAnnotationsExtensions. These libraries provide additional features such as custom validation rules, client-side validation, and more.  
  
The third way is to create custom validation logic in the controller. This can be done by using the TryValidateModel() method to validate the model, or by writing custom validation logic in the controller action.  
  
Finally, you can also use a combination of the above methods to create a comprehensive data validation strategy. This can include using the built-in validation features of the MVC framework, using third-party libraries, and writing custom validation logic in the controller.

**9. What is the difference between a ViewModel and a Model in MVC?**

The Model in MVC is the layer that contains the application's business logic and data access logic. It is responsible for retrieving and storing data from the database, and for performing any business logic operations. The ViewModel is a layer that sits between the Model and the View. It is responsible for providing the data and logic needed by the View to render the user interface. The ViewModel is typically used to transform data from the Model into a format that is more suitable for the View. It can also contain additional logic that is specific to the View, such as validation logic or formatting logic. The ViewModel is also responsible for mapping user input from the View back to the Model.

**10. How do you handle caching in an MVC application?**

Caching is an important part of any MVC application, as it can help improve performance and reduce server load. In an MVC application, caching can be handled in a few different ways.  
  
The first way is to use the built-in caching features of the MVC framework. Most MVC frameworks have built-in caching mechanisms that allow developers to easily cache data and objects. This can be done by using the caching API provided by the framework, or by using a third-party caching library.  
  
The second way to handle caching in an MVC application is to use a caching layer. This is a layer of code that sits between the application and the database, and is responsible for caching data and objects. This layer can be implemented using a variety of technologies, such as memcached, Redis, or even a custom caching solution.  
  
The third way to handle caching in an MVC application is to use a content delivery network (CDN). A CDN is a network of servers that are located around the world, and are responsible for delivering content to users. By using a CDN, developers can ensure that content is delivered quickly and reliably to users, regardless of their location.  
  
Finally, developers can also use a combination of the above methods to handle caching in an MVC application. By combining the built-in caching features of the MVC framework, a caching layer, and a CDN, developers can ensure that their application is performing optimally.

<https://dotnettutorials.net/lesson/mvc-basic-interview-questions-answers/>

https://www.codeproject.com/Articles/556995/ASP-NET-MVC-Interview-Questions-with-Answers