Asp.net MVC Interview Questions

1. **What is MVC (Model View Controller)**

MVC is a software architecture pattern for developing web application. It is handled by three objects Model-View-Controller. Below is how each one of them handles the task.

* The View is responsible for the look and feel.
* Model represents the real world object and provides data to the View.
* The Controller is responsible for taking the end user request and loading the appropriate Model and View

1. **What is MVC Application Life Cycle**

Any web application has two main execution steps first understanding the request and depending on the type of the request sending out the appropriate response. MVC application life cycle is not different it has two main phases first creating the request object and second sending our response to the browser.

Creating the request object: -The request object creation has four major steps. Below is the detail explanation.

**Step 1 Fill route**:  MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the route table with routes collection. This filling of route table happens in the global.asax file.

**Step 2 Fetch route:**  Depending on the URL sent “UrlRoutingModule” searches the route table to create “RouteData” object which has the details of which controller and action to invoke.

**Step 3 Request context created**: The “RouteData” object is used to create the “RequestContext” object.

**Step 4 Controller instance created:**  This request object is sent to “MvcHandler” instance to create the controller class instance. Once the controller class object is created it calls the “Execute” method of the controller class.

**Step 5 Creating Response object:**  This phase has two steps executing the action and finally sending the response as a result to the view.

**Explain in which assembly is the MVC framework is defined?**

The MVC framework is defined in System.Web.Mvc.

**What are the advantages of MVC**

* A main advantage of MVC is separation of concern. Separation of concern means we divide the application Model, Control and View.
* We can easily maintain our application because of separation of concern.
* In the same time we can split many developers work at a time. It will not affects one developer work to another developer work.
* It supports TTD (test-driven development). We can create an application with unit test. We can write won test case.
* Latest version of MVC Support default responsive web site and mobile templates.

1. **List out different return types of a controller action method?**

There are total nine return types we can use to return results from controller to view.

**ViewResult (View):** This return type is used to return a webpage from an action method.

* **PartialviewResult (Partialview):** This return type is used to send a part of a view which will be rendered in another view.
* **RedirectResult (Redirect):** This return type is used to redirect to any other controller and action method depending on the URL.
* **RedirectToRouteResult (RedirectToAction, RedirectToRoute):** This return type is used when we want to redirect to any other action method.
* **ContentResult (Content):** This return type is used to return HTTP content type like text/plain as the result of the action.
* **jsonResult (json):** This return type is used when we want to return a JSON message.
* **javascriptResult (javascript):** This return type is used to return JavaScript code that will run in browser.
* **FileResult (File):** This return type is used to send binary output in response.
* **EmptyResult:** This return type is used to return nothing (void) in the result.
* **What is the difference between each version of MVC 2, 3 , 4, 5 and 6?**

**MVC 6**

ASP.NET MVC and Web API has been merged in to one.

Side by side - deploy the runtime and framework with your application

No need to recompile for every change. Just hit save and refresh the browser.

Dependency injection is inbuilt and part of MVC.

Everything packaged with NuGet, Including the .NET runtime itself.

New JSON based project structure.

Compilation done with the new Roslyn real-time compiler.

**MVC 5**

Asp.Net Identity

Attribute based routing

Bootstrap in the MVC template

Filter overrides

Authentication Filters

**MVC 4**

ASP.NET Web API

New mobile project template

Refreshed and modernized default project templates

Many new features to support mobile apps

**MVC 3**

Razor

HTML 5 enabled templates

JavaScript and Ajax

Support for Multiple View Engines

Model Validation Improvements

**MVC 2**

Templated Helpers

Client-Side Validation

Areas

Asynchronous Controllers

Html.ValidationSummary Helper Method

DefaultValueAttribute in Action-Method Parameters

Binding Binary Data with Model Binders

DataAnnotations Attributes

Model-Validator Providers

New RequireHttpsAttribute Action Filter

1. **What are Filters in MVC?**

In MVC, many times we would like to perform some action before or after a particular operation. For achieving this functionality, ASP.NET MVC provides feature to add pre and post action behaviors on controller's action methods.

**Types of Filters:**

 ASP.NET MVC framework supports the following action filters:

* **Action Filters:** Action filters are used to implement logic that gets executed before and after a controller action executes.
* **Authorization Filters:** Authorization filters are used to implement authentication and authorization for controller actions.
* **Result Filters:** Result filters contain logic that is executed before and after a view result is executed. For example, you might want to modify a view result right before the view is rendered to the browser.
* **Exception Filters:** You can use an exception filter to handle errors raised by either your controller actions or controller action results. You can also use exception filters to log errors.

1. **What are Action Filters in MVC?**

Action Filters are additional attributes that can be applied to either a controller section or the entire controller to modify the way in which action is executed. These attributes are special .NET classes derived from System.Attribute which can be attached to classes, methods, properties and fields.

 ASP.NET MVC provides the following action filters:

**Output Cache:** This action filter caches the output of a controller action for a specified amount of time.

**Handle Error:** This action filter handles errors raised when a controller action executes.

**Authorize:** This action filter enables you to restrict access to a particular user or role.

Now we will see the code example to apply these filters on an example controller ActionFilterDemoController. (ActionFilterDemoController is just used as an example. You can use these filters on any of your controllers.)

1. **What are HTML helpers in MVC?**

HTML helpers help you to render HTML controls in the view. For instance if you want to display a HTML textbox on the view , below is the HTML helper code.

<%= Html.TextBox("FirstName") %>

For checkbox below is the HTML helper code. In this way we have HTML helper methods for every HTML control that exists.

<%= Html.CheckBox("Yes") %>

1. **What is the difference between “HTML.TextBox” and “HTML.TextBoxFor”?**

  Both provide the same HTML output, “HTML.TextBoxFor” is strongly typed while “HTML.TextBox” isn’t.     Below is a simple HTML code which just creates a simple textbox with “FirstName” as name.

Html.TextBox("FirstName ")

Below is “Html.TextBoxFor” code which creates HTML textbox using the property name ‘FirstName” from object “m”.

Html.TextBoxFor(m => m.CustomerCode)

In the same way, we have for other HTML controls like for checkbox we have “Html.CheckBox” and “Html.CheckBoxFor”.

1. **What is Route in MVC? What is Default Route in MVC?**

A route is a URL pattern that is mapped to a handler. The handler can be a physical file, such as a .aspx file in a Web Forms application. A handler can also be a class that processes the request, such as a controller in an MVC application. To define a route, you create an instance of the Route class by specifying the URL pattern, the handler, and optionally a name for the route.

You add the route to the application by adding the Route object to the static Routes property of the RouteTable class. The Routesproperty is a RouteCollection object that stores all the routes for the application.

You typically do not have to write code to add routes in an MVC application. Visual Studio project templates for MVC include preconfigured URL routes. These are defined in the Mvc Application class, which is defined in the Global.asax file.

|  |  |
| --- | --- |
| **Route definition** | **Example of matching URL** |
| {controller}/{action}/{id} | /Products/show/beverages |
| {table}/Details.aspx | /Products/Details.aspx |
| blog/{action}/{entry} | /blog/show/123 |
| {reporttype}/{year}/{month}/{day} | /sales/2008/1/5 |
| {locale}/{action} | /US/show |
| {language}-{country}/{action} | /en-US/show |

**Default Route**

The default ASP.NET MVC project templates add a generic route that uses the following URL convention to break the URL for a given request into three named segments.

**URL: "{controller}/{action}/{id}"**

This route pattern is registered via call to the MapRoute() extension method of RouteCollection.

1. **Where is the route mapping code written?**

The route mapping code is written in "RouteConfig.cs" file and registered using "global.asax" application start event.

1. **What is the difference between Temp data, View, and View Bag?**

In ASP.NET MVC there are three ways to pass/store data between the controllers and views.

**ViewData**

* ViewData is used to pass data from controller to view.
* It is derived from ViewDataDictionary class.
* It is available for the current request only.
* Requires typecasting for complex data type and checks for null values to avoid error.
* If redirection occurs, then its value becomes null.

**ViewBag**

* ViewBag is also used to pass data from the controller to the respective view.
* ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0
* It is also available for the current request only.
* If redirection occurs, then its value becomes null.
* Doesn’t require typecasting for complex data type.

**TempData**

* TempData is derived from TempDataDictionary class
* TempData is used to pass data from the current request to the next request
* It keeps the information for the time of an HTTP Request. This means only from one page to another. It helps to maintain the data when we move from one controller to another controller or from one action to another action
* It requires typecasting for complex data type and checks for null values to avoid error. Generally, it is used to store only one time messages like the error messages and validation messages

1. **What is Partial View in MVC?**

 Partial view is a reusable view (like a user control) which can be embedded inside other view. For example let’s say all your pages of your site have a standard structure with left menu, header, and footer as shown in the image below.

For every page you would like to reuse the left menu, header, and footer controls. So you can go and create partial views for each of these items and then you call that partial view in the main view.

1. **How did you create a partial view and consume it?**

When you add a view to your project you need to check the “Create partial view” check box.

Figure: Create partial view

Once the partial view is created you can then call the partial view in the main view using the Html.RenderPartial method as shown in the below code snippet:

<body>

<div>

<% Html.RenderPartial("MyView"); %>

</div>

</body>

1. **Explain what is the difference between View and Partial View?**

**View:**

It contains the layout page.

Before any view is rendered, viewstart page is rendered.

View might have markup tags like body, html, head, title, meta etc.

View is not lightweight as compare to Partial View.

**Partial View:**

It does not contain the layout page.

Partial view does not verify for a viewstart.cshtml.We cannot put common code for a partial view within the viewStart.cshtml.page.

Partial view is designed specially to render within the view and just because of that it does not consist any mark up.

We can pass a regular view to the RenderPartial method.

1. **Explain attribute based routing in MVC?**

In ASP.NET MVC 5.0 we have a new attribute route, By using the "Route" attribute we can define the URL structure. For example in the below code we have decorated the "GotoAbout" action with the route attribute. The route attribute says that the "GotoAbout" can be invoked using the URL structure "Users/about".

public class HomeController: Controller

{

[Route("Users/about")]

publicActionResultGotoAbout()

{

return View();

}

}

1. **What is Razor in MVC?**

Razor is not a new programming language itself, but uses C# syntax for embedding code in a page without the ASP.NET delimiters: <%= %>. It is a simple-syntax view engine and was released as part of ASP.NET MVC 3. The Razor file extension is "cshtml" for the C# language. It supports TDD (Test Driven Development) because it does not depend on the System.Web.UI.Page class.

1. **How to implement Forms authentication in MVC?**

ASP.NET forms authentication occurs after IIS authentication is completed. You can configure forms authentication by using forms element with in web.config file of your application. The default attribute values for forms authentication are shown below:

<system.web>

<authenticationmode="Forms">

<formsloginUrl="Login.aspx" protection="All" timeout="30" name=".ASPXAUTH" path="/" requireSSL="false" slidingExpiration="true" defaultUrl="default.aspx" cookieless="UseDeviceProfile" enableCrossAppRedirects="false" />

</authentication>

</system.web>

The FormsAuthentication class creates the authentication cookie automatically when SetAuthCookie() or RedirectFromLoginPage() methods are called. The value of authentication cookie contains a string representation of the encrypted and signed FormsAuthenticationTicket object.

1. **What is Areas in MVC?**

From ASP.Net MVC 2.0 Microsoft provided a new feature in MVC applications, Areas. Areas are just a way to divide or “isolate” the modules of large applications in multiple or separated MVC. like:

When you add an area to a project, a route for the area is defined in an AreaRegistration file. The route sends requests to the area based on the request URL. To register routes for areas, you add code to the Global.asax file that can automatically find the area routes in the AreaRegistration file.

1. **Explain the concept of MVC Scaffolding?**

ASP.NET Scaffolding is a code generation framework for ASP.NET Web applications. Visual Studio 2013 includes pre-installed code generators for MVC and Web API projects. You add scaffolding to your project when you want to quickly add code that interacts with data models. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

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 Website.

**Scaffolding Templates:**

**Create:** It creates a View that helps in creating a new record for the Model. It automatically generates a label and input field for each property in the Model.

**Delete:** It creates a list of records from the model collection along with the delete link with delete record.

**Details:** It generates a view that displays the label and an input field of the each property of the Model in the MVC framework.

**Edit:** It creates a View with a form that helps in editing the current Model. It also generates a form with label and field for each property of the model.

**List:** It generally creates a View with the help of a HTML table that lists the Models from the Model Collection. It also generates a HTML table column for each property of the Model.

1. **What is the difference between ActionResult and ViewResult?**

* **ActionResult** is an abstract class while **ViewResult** derives from the ActionResult class.
* **ActionResult** has several derived classes like **ViewResult**, **JsonResult**, **FileStreamResult**, and so on.
* **ActionResult** can be used to exploit polymorphism and dynamism. So if you are returning different types of views dynamically, **ActionResult** is the best thing. For example in the below code snippet, you can see we have a simple action called **DynamicView**. Depending on the flag (**IsHtmlView**) it will either return a **ViewResult** or **JsonResult.**

public ActionResult DynamicView()

{

if (IsHtmlView)

return View(); // returns simple ViewResult

else

return Json(); // returns JsonResult view

}

1. **What is Route Constraints in MVC?**

This is very necessary for when we want to add a specific constraint to our URL.  So, we want to set some constraint string after our host name. Fine, let's see how to implement it.

It's very simple to implement, just open the RouteConfig.cs file and you will find the routing definition in that. And modify the routing entry as in the following. We will see that we have added “abc” before.

Controller name, now when we browse we need to specify the string in the URL, as in the following:

1. **What is Output Caching in MVC?**

The main purpose of using Output Caching is to dramatically improve the performance of an ASP.NET MVC Application. It enables us to cache the content returned by any controller method so that the same content does not need to be generated each time the same controller method is invoked. Output Caching has huge advantages, such as it reduces server round trips, reduces database server round trips, reduces network traffic etc.

OutputCache label has a "Location" attribute and it is fully controllable. Its default value is "Any", however there are the following locations available; as of now, we can use any one.

* Any
* Client
* Downstream
* Server
* None
* ServerAndClient

1. **What is the use of Keep and Peek in “TempData”?**

Once “TempData” is read in the current request it’s not available in the subsequent request. If we want “TempData” to be read and also available in the subsequent request then after reading we need to call “Keep” method as shown in the code below.

@TempData["MyData"];

TempData.Keep("MyData");

 The more shortcut way of achieving the same is by using “Peek”. This function helps to read as well advices MVC to maintain “TempData” for the subsequent request.

string str = TempData.Peek("MyData ").ToString();

1. **What is Bundling and Minification in MVC?**

Bundling and minification are two new techniques introduced to improve request load time. It improves load time by reducing the number of requests to the server and reducing the size of requested assets (such as CSS and JavaScript).

**Bundling:** It lets us combine multiple JavaScript (.js) files or multiple cascading style sheet (.css) files so that they can be downloaded as a unit, rather than making individual HTTP requests.

**Minification:** It squeezes out whitespace and performs other types of compression to make the downloaded files as small as possible. At runtime, the process identifies the user agent, for example IE, Mozilla, etc. and then removes whatever is specific to Mozilla when the request comes from IE.

1. **What is Validation Summary in MVC?**

The ValidationSummary helper method generates an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

The ValidationSummary can be used to display all the error messages for all the fields. It can also be used to display custom error messages. The following figure shows how ValidationSummary displays the error messages.

1. **What are the Folders in MVC application solutions?**

When you create a project a folder structure gets created by default under the name of your project which can be seen in solution explorer. Below I will give you a brief explanation of what these folders are for.

**Model:** This folder contains classes that is used to provide data. These classes can contain data that is retrieved from the database or data inserted in the form by the user to update the database.

**Controllers:** These are the classes which will perform the action invoked by the user. These classes contains methods known as "Actions" which responds to the user action accordingly.

**Views:** These are simple pages which uses the model class data to populate the HTML controls and renders it to the client browser.

**App\_Start:** Contains Classes such as FilterConfig, RoutesConfig, WebApiConfig. As of now we need to understand the RouteConfig class. This class contains the default format of the URL that should be supplied in the browser to navigate to a specified page.

1. **If we have multiple filters, what’s the sequence for execution?**

* Authorization filters
* Action filters
* Response filters
* Exception filters

1. **What is ViewStart?**

Razor View Engine introduced a new layout named \_ViewStart which is applied on all view automatically. Razor View Engine firstly executes the \_ViewStart and then start rendering the other view and merges them.

Example of Viewstart:

@ {

Layout = "~/Views/Shared/\_v1.cshtml";

} < !DOCTYPE html >

< html >

< head >

< meta name = "viewport"

content = "width=device-width" / >

< title > ViewStart < /title> < /head> < body >

< /body> < /html>

1. **What is JsonResultType in MVC?**

Action methods on controllers return JsonResult (JavaScript Object Notation result) that can be used in an AJAX application. This class is inherited from the "ActionResult" abstract class. Here Json is provided one argument which must be serializable. The JSON result object that serializes the specified object to JSON format.

public JsonResult JsonResultTest()

{

return Json("Hello My Friend!");

}

1. **What are the Difference between ViewBag&ViewData?**

**ViewData** is a dictionary of objects that is derived from ViewDataDictionary class and accessible using strings as keys.

**ViewBag** is a dynamic property that takes advantage of the new dynamic features in C# 4.0.

**ViewData** requires typecasting for complex data type and check for null values to avoid error.

**ViewBag** doesn't require typecasting for complex data type.

Calling of ViewBag is:

ViewBag.Name = "Vikash";

Calling of ViewData is :

ViewData["Name"] = " Vikash ";

1. **Explain RenderSection in MVC?**

RenderSection() is a method of the WebPageBase class. Scott wrote at one point, The first parameter to the "RenderSection()" helper method specifies the name of the section we want to render at that location in the layout template. The second parameter is optional, and allows us to define whether the section we are rendering is required or not. If a section is "required", then Razor will throw an error at runtime if that section is not implemented within a view template that is based on the layout file (that can make it easier to track down content errors). It returns the HTML content to render.

<div id="body">

@RenderSection("featured", required: false)

<section class="content-wrapper main-content clear-fix">

@RenderBody()

</section>

</div>

1. **What is GET and POST Actions Types?**

**GET**

GET is used to request data from a specified resource. With all the GET request we pass the URL which is compulsory, however it can take the following overloads.

.get(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ] ).done/.fail

**POST**

POST is used to submit data to be processed to a specified resource. With all the POST requests we pass the URL which is compulsory and the data, however it can take the following overloads.

.post(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ] )

1. **What is display mode in MVC?**

Display mode displays views depending on the device the user has logged in with. So we can create different views for different devices and display mode will handle the rest.

For example we can create a view “Home.aspx” which will render for the desktop computers and Home.Mobile.aspx for mobile devices. Now when an end user sends a request to the MVC application, display mode checks the “user agent” headers and renders the appropriate view to the device accordingly.

1. **How can we do exception handling in MVC?**

In the controller you can override the “OnException” event and set the “Result” to the view name which you want to invoke when error occurs. In the below code you can see we have set the “Result” to a view named as “Error”.

We have also set the exception so that it can be displayed inside the view.

public class HomeController : Controller

{

protected override void OnException(ExceptionContext filterContext)

{

Exception ex = filterContext.Exception;

filterContext.ExceptionHandled = true;

var model = new HandleErrorInfo(filterContext.Exception, "Controller","Action");

filterContext.Result = new ViewResult()

{

ViewName = "Error",

ViewData = new ViewDataDictionary(model)

};

}

}

To display the above error in view we can use the below code

@Model.Exception;

1. **What is the use of remote validation in MVC?**

Remote validation is the process where we validate specific data posting data to a server without posting the entire form data to the server. Let's see an actual scenario, in one of my projects I had a requirement to validate an email address, whether it already exists in the database. Remote validation was useful for that; without posting all the data we can validate only the email address supplied by the user.

Let's create a MVC project and name it accordingly, for me its “TestingRemoteValidation”. Once the project is created let's create a model named UserModel that will look like:

public class UserModel

{

[Required]

public string UserName

{

get;

set;

}

[Remote("CheckExistingEmail", "Home", ErrorMessage = "Email already exists!")]

public string UserEmailAddress

{

get;

set;

}

}

Let's get some understanding of the remote attribute used, so the very first parameter “CheckExistingEmail” is the the name of the action. The second parameter “Home” is referred to as controller so to validate the input for the UserEmailAddress the “CheckExistingEmail” action of the “Home” controller is called and the third parameter is the error message. Let's implement the “CheckExistingEmail” action result in our home controller.

public ActionResult CheckExistingEmail(string UserEmailAddress)

{

bool ifEmailExist = false;

try

{

ifEmailExist = UserEmailAddress.Equals("mukeshknayak@gmail.com") ? true : false;

return Json(!ifEmailExist, JsonRequestBehavior.AllowGet);

} catch (Exception ex)

{

return Json(false, JsonRequestBehavior.AllowGet);

}

}

1. **Explain Dependency Resolution?**

Dependency Resolver again has been introduced in MVC3 and it is greatly simplified the use of dependency injection in your applications. This turn to be easier and useful for decoupling the application components and making them easier to test and more configurable.

1. **Explain Bundle.Config in MVC4?**

"BundleConfig.cs" in MVC4 is used to register the bundles by the bundling and minification system. Many bundles are added by default including jQuery libraries like - jquery.validate, Modernizr, and default CSS references.

1. **What is the meaning of Unobtrusive JavaScript?**

This is a general term that conveys a general philosophy, similar to the term REST (Representational State Transfer). Unobtrusive JavaScript doesn't intermix JavaScript code in your page markup.

Eg : Instead of using events like onclick and onsubmit, the unobtrusive JavaScript attaches to elements by their ID or class based on the HTML5 data- attributes.