**Explain what is Model-View-Controller?**

MVC is a software architecture pattern for developing web application. It is handled by three objects Model-View-Controller.

**Mention what does Model-View-Controller represent in an MVC application?**

* **Model**– It represents the application data domain. In other words applications business logic is contained within the model and is responsible for maintaining data
* **View**– It represents the user interface, with which the end users communicates. In short all the user interface logic is contained within the VIEW
* **Controller**– It is the controller that answers to user actions. Based on the user actions, the respective controller responds within the model and choose a view to render that display the user interface.  The user input logic is contained with-in the controller

**Mention what is the advantages of MVC?**

* MVC segregates your project into a different segment, and it becomes easy for developers to work on
* It is easy to edit or change some part of your project that makes project less development and maintenance cost
* MVC makes your project more systematic

**Explain the role of “ActionFilters” in MVC?**

* In MVC “ ActionFilters” help you to execute logic while MVC action is executed or its executing.

**Explain what are the steps for the execution of an MVC project?**

The steps for the execution of an MVC project includes

* Receive first request for the application
* Performs routing
* Creates MVC request handler
* Create Controller
* Execute Controller
* Invoke action
* Execute Result

**Explain what is routing? What are the three segments for routing is important?**

Routing helps you to decide a URL structure and map the URL with the Controller.

The three segments those are important for routing is

* ControllerName
* ActionMethodName
* Parameter

**14) Explain how routing is done in MVC pattern?**

There is a group of routes called the RouteCollection, which consists of registered routes in the application.  The RegisterRoutes method records the routes in this collection.  A route defines a URL pattern and a handler to use if the request matches the pattern. The first parameter to the MapRoute method is the name of the route. The second parameter will be the pattern to which the URL matches.  The third parameter might be the default values for the placeholders if they are not determined.

**List out the types of result in MVC?**

In MVC, there are twelve types of results in MVC where “ActionResult” class is the main class while the 11 are their sub-types

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

**Mention the order of the filters that get executed, if the multiple filters are implemented?**

The filter order would be like

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

#### What are Action Methods?

Action methods are defined by the controllers; urls are mapped to the action methods.  
The request which is received by our mvc application is ultimately handled by an action method. Action method generates the response in the  form of ActionResult. The action method to execute is determined according to the routing rules defined by our application.

What is ActionResult?

ActionResult is a class which represents the result of an action method. Action methods returns an object of a type which derives from this ActionResult class.Since ActionResult is an abstract class so it provides few derived classes whose object the action method can create to return the response.Also there are few methods available to the controller class to create ActionResult subclass,so we don’t need to explicitly create an object of the ActionResult  and can just call the method.  
Some of the classes deriving from the ActionResult are:

|  |  |  |
| --- | --- | --- |
| Action Result | Helper Method | Description |
| ViewResult | View | Renders a view |
| PartialViewResult | PartialView | Renders a partial view, which is a view which can be used inside another view. |
| RedirectResult | Redirect | Redirects to different action method as specified in the URL. |
| RedirectToRouteResult | RedirectToRoute | Redirects to another action method. |
| ContentResult | Content | Returns the user supplied content type. |
| JsonResult | Json | Returns a JSON object. |

**What are HTML helpers?**

HTML helpers are methods which returns HTML strings.There are few inbuilt HTML helpers which we can use.If the inbuilt helpers are not meeting our needs ,we can also create our custom HTML helpers.They are similar to the webforms controls as both the webforms controls and the MVC HTML helpers returns HTML.  
But HTML helpers are lightweight compared to the webforms controls.  
Following are some of the commonly used HTML helper methods for rendering the HTML form elements

* BeginForm()
* EndForm()
* TextArea()
* TextBox()
* CheckBox()
* RadioButton()
* DropDownList()
* Hidden()
* Password()

**What are the main differences between ASP.NET WebForms  and ASP.NET MVC?**

|  |  |
| --- | --- |
| WebForms | MVC |
| There is no proper separation of concerns,the application logic resides in the code behind of the webform ,so the .aspx page and its code behind are tightly coupled.This makes it difficult to make the changes in one without effecting the other. | In the case of ASP.NET MVC there is a separation of concerns ,so the Model,View and Controller are loosely coupled.This means that we can easily make changes in one component without effecting the other components. |
| There is viewstate | No Viewstate |
| Limited control over the generated HTML | provides complete control over the generated HTML. |

#### Do we have ViewState in MVC?

This is a very commonly asked question.It lets the interviewer judge your understanding of MVC.One big difference between WebForms and MVC is MVC does not have viewstate.The reason that MVC does not have viewstate is because viewstate is stored in a hidden field on the page.So this increases the size of the page significantly and impacts the page load time.

#### What are Action Filters?

Sometimes we want to execute some logic either before the execution of the action method or after the execution of the action method.We can use Action Filter for such kind of scenario.Action Filters defines logic which is executed before or after the execution of the action method.Action Filters are attributes which we can apply to the action methods.Following are the MVC Filters:

* Authorization filter
* Action filter
* Result filter
* Exception filter

#### Which Filter executes last?

Exception filter executes last,after all other filters have executed.

#### What is a View Engine?

View Engines are responsible for generating the HTML from the views.Views contains HTML and source code in some programming language such as C#. View Engine generates HTML from the view which is returned to the browser and rendered.Two main View Engines are WebForms and Razor ,each has its own syntax.

#### What are strongly typed Helpers?

In the case of normal helper methods we need to provide the string values to the helper methods.As these are string literals so there is no compile time checking and also intellisense support is available.

In contrast strongly typed helper methods takes lambda expressions so they provide  
intellisense support and also are type checked at compile time.

Following are the normal and strongly typed helper methods which generates the same HTML

|  |  |
| --- | --- |
| 1  2 | Html.TextBox("Name")  Html.TextBoxFor(model => model.Name) |

#### What is \_ViewStart.cshtml?

\_ViewStart.cshtml defines code which is executed before any code in any of the views is executed.It is applied to all the views in a subdirectory.For example following is commonly included in \_ViewStart.cshtml file:

**What is Bundling and Minification?**

Bundling and Minification is used for improving the performance of the application.Bundling reduces the number of HTTP requests made to the server by combining several files into a single bundle.Minification reduces the size of the individual files by removing unnecessary characters.

#### How can we prevent Cross Site Request Forgery(CSRF)?

To prevent CSRF we apply the ValidateAntiForgeryToken attribute to an action method:

|  |  |
| --- | --- |
| 1 | [ValidateAntiForgeryToken()]  **Partial or RenderPartial:** No need to create action method. use when data to be display on the partial view is already present in model of current page.  **Action or RenderAction:** Requires child action method. use when data to display on the view has independent model. |

**DATABASE**

* Truncate is used to delete table content and the action can **not** be rolled back, whereas Delete is used to delete one or more rows in the table and **can** be rolled back.
* Identity column will start from 1.

**ACID (Atomicity, Consistency, Isolation, Durability)** is a set of properties that guarantee that database transactions are processed reliably. They are defined as follows:

* **Atomicity.** Atomicity requires that each transaction be “all or nothing”: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes.
* **Consistency.** The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
* **Isolation.** The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method (i.e. if it uses strict - as opposed to relaxed - serializability), the effects of an incomplete transaction might not even be visible to another transaction.
* **Durability.** Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

What are the NVL and the NVL2 functions in SQL? How do they differ?

Both the NVL(exp1, exp2) and NVL2(exp1, exp2, exp3) functions check the value exp1 to see if it is null.

With the NVL(exp1, exp2) function, if exp1 is *not* null, then the value of exp1 is returned; otherwise, the value of exp2 is returned, but case to the same data type as that of exp1.

With the NVL2(exp1, exp2, exp3) function, if exp1 is *not* null, then exp2 is returned; otherwise, the value of exp3 is returned.

difference between the RANK() and DENSE\_RANK() functions

 consider the set {25, 25, 50, 75, 75, 100}. For such a set, RANK() will return {1, 1, 3, 4, 4, 6} (note that the values 2 and 5 are skipped), whereas DENSE\_RANK() will return {1,1,2,3,3,4}

**What is a constraint?**

Constraint can be used to specify the limit on the data type of table. Constraint can be specified while creating or altering the table statement. Sample of constraint are.

* NOT NULL.
* CHECK.
* DEFAULT.
* UNIQUE.
* PRIMARY KEY.
* FOREIGN KEY.

### First Normal Form (1NF)

Each column is unique in 1NF.

Example:

Sample Employee table, it displays employees are working with multiple departments.

|  |  |  |
| --- | --- | --- |
| Employee | Age | Department |
| Melvin | 32 | Marketing, Sales |
| Edward | 45 | Quality Assurance |
| Alex | 36 | Human Resource |

**Employee table following 1NF:**

|  |  |  |
| --- | --- | --- |
| Employee | Age | Department |
| Melvin | 32 | Marketing |
| Melvin | 32 | Sales |
| Edward | 45 | Quality Assurance |
| Alex | 36 | Human Resource |

### Second Normal Form (2NF)

The entity should be considered already in 1NF and all attributes within the entity should depend solely on the unique identifier of the entity.

Example:

Sample Products table:

|  |  |  |
| --- | --- | --- |
| productID | product | Brand |
| 1 | Monitor | Apple |
| 2 | Monitor | Samsung |
| 3 | Scanner | HP |
| 4 | Head phone | JBL |

**Product table following 2NF:**  
**Products Category table:**

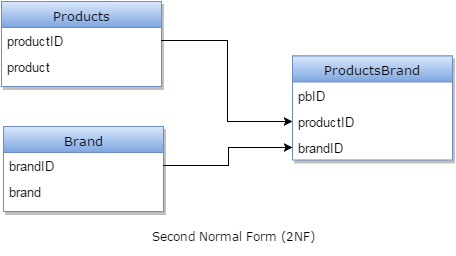
|  |  |
| --- | --- |
| productID | product |
| 1 | Monitor |
| 2 | Scanner |
| 3 | Head phone |

**Brand table:**

|  |  |
| --- | --- |
| brandID | brand |
| 1 | Apple |
| 2 | Samsung |
| 3 | HP |
| 4 | JBL |

**Products Brand table:**

|  |  |  |
| --- | --- | --- |
| pbID | productID | brandID |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 3 |
| 4 | 3 | 4 |



### Third Normal Form (3NF)

The entity should be considered already in 2NF and no column entry should be dependent on any other entry (value) other than the key for the table.

If such an entity exists, move it outside into a new table.

3NF is achieved are considered as the database is normalized.

### Boyce-Codd Normal Form (BCNF)

3NF and all tables in the database should be only one primary key.

## Life-Cycle Events

Within each stage of the life cycle of a page, the page raises events that you can handle to run your own code. For control events, you bind the event handler to the event, either declaratively using attributes such as **onclick**, or in code.

Pages also support automatic event wire-up, meaning that ASP.NET looks for methods with particular names and automatically runs those methods when certain events are raised. If the **AutoEventWireup** attribute of the [@ Page](https://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive is set to **true**, page events are automatically bound to methods that use the naming convention of **Page\_***event*, such as **Page\_Load** and **Page\_Init**. For more information on automatic event wire-up, see [ASP.NET Web Forms Server Control Event Model](https://msdn.microsoft.com/en-us/library/y3bwdsh3.aspx).

The following table lists the page life-cycle events that you will use most frequently. There are more events than those listed; however, they are not used for most page-processing scenarios. Instead, they are primarily used by server controls on the ASP.NET Web page to initialize and render themselves. If you want to write custom ASP.NET server controls, you need to understand more about these events. For information about creating custom controls, see [Developing Custom ASP.NET Server Controls](https://msdn.microsoft.com/en-us/library/zt27tfhy.aspx).

|  |  |
| --- | --- |
| **Page Event** | **Typical Use** |
| [PreInit](https://msdn.microsoft.com/en-us/library/system.web.ui.page.preinit.aspx) | Raised after the start stage is complete and before the initialization stage begins.  Use this event for the following:   * Check the [IsPostBack](https://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property to determine whether this is the first time the page is being processed. The [IsCallback](https://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx)and [IsCrossPagePostBack](https://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) properties have also been set at this time. * Create or re-create dynamic controls. * Set a master page dynamically. * Set the [Theme](https://msdn.microsoft.com/en-us/library/system.web.ui.page.theme.aspx) property dynamically. * Read or set profile property values.  |  | | --- | | **Note Note** | | If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event. | |
| [Init](https://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) | Raised after all controls have been initialized and any skin settings have been applied. The [Init](https://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event of individual controls occurs before the [Init](https://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event of the page.  Use this event to read or initialize control properties. |
| [InitComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.initcomplete.aspx) | Raised at the end of the page's initialization stage. Only one operation takes place between the [Init](https://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) and [InitComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.initcomplete.aspx) events: tracking of view state changes is turned on. View state tracking enables controls to persist any values that are programmatically added to the [ViewState](https://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstate.aspx) collection. Until view state tracking is turned on, any values added to view state are lost across postbacks. Controls typically turn on view state tracking immediately after they raise their [Init](https://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event.  Use this event to make changes to view state that you want to make sure are persisted after the next postback. |
| [PreLoad](https://msdn.microsoft.com/en-us/library/system.web.ui.page.preload.aspx) | Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the [Request](https://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) instance. |
| [Load](https://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) | The [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls the [OnLoad](https://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) method on the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object, and then recursively does the same for each child control until the page and all controls are loaded. The [Load](https://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of individual controls occurs after the [Load](https://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of the page.  Use the [OnLoad](https://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) event method to set properties in controls and to establish database connections. |
| Control events | Use these events to handle specific control events, such as a [Button](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control's [Click](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.click.aspx) event or a [TextBox](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control's [TextChanged](https://msdn.microsoft.com/en-us/library/system.web.ui.mobilecontrols.textbox.textchanged.aspx)event.   |  | | --- | | **Note Note** | | In a postback request, if the page contains validator controls, check the [IsValid](https://msdn.microsoft.com/en-us/library/system.web.ui.page.isvalid.aspx) property of the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) and of individual validation controls before performing any processing. | |
| [LoadComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.loadcomplete.aspx) | Raised at the end of the event-handling stage.  Use this event for tasks that require that all other controls on the page be loaded. |
| [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) | Raised after the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object has created all controls that are required in order to render the page, including child controls of composite controls. (To do this, the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls [EnsureChildControls](https://msdn.microsoft.com/en-us/library/system.web.ui.control.ensurechildcontrols.aspx) for each control and for the page.)  The [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object raises the [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event on the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object, and then recursively does the same for each child control. The [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event of individual controls occurs after the [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event of the page.  Use the event to make final changes to the contents of the page or its controls before the rendering stage begins. |
| [PreRenderComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.prerendercomplete.aspx) | Raised after each data bound control whose [DataSourceID](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.databoundcontrol.datasourceid.aspx) property is set calls its [DataBind](https://msdn.microsoft.com/en-us/library/system.web.ui.control.databind.aspx) method. For more information, see [Data Binding Events for Data-Bound Controls](https://msdn.microsoft.com/en-us/library/ms178472.aspx#data_binding_events_for_databound_controls) later in this topic. |
| [SaveStateComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.savestatecomplete.aspx) | Raised after view state and control state have been saved for the page and for all controls. Any changes to the page or controls at this point affect rendering, but the changes will not be retrieved on the next postback. |
| [Render](https://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) | This is not an event; instead, at this stage of processing, the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls this method on each control. All ASP.NET Web server controls have a [Render](https://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method that writes out the control's markup to send to the browser.  If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the [Render](https://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method. For more information, see [Developing Custom ASP.NET Server Controls](https://msdn.microsoft.com/en-us/library/zt27tfhy.aspx).  A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code. |
| [Unload](https://msdn.microsoft.com/en-us/library/system.web.ui.control.unload.aspx) | Raised for each control and then for the page.  In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.  For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks.   |  | | --- | | **Note Note** | | During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream. If you attempt to call a method such as the **Response.Write** method, the page will throw an exception. | |