**View state** :

maintain state in asp .net( cache)

controls across postbacks

It is automatically managed by the framework and is used to preserve the values of the controls and other page settings between requests

**State Persistence:** When an ASP.NET Web Forms page is rendered and sent to the client browser, the values of controls and other page settings are serialized and saved into a hidden field called "\_\_VIEWSTATE" in the page's HTML output.

**Postback Handling:** When the client submits the form (e.g., by clicking a button), the browser sends the form data back to the server. The server then processes the request, updates the page state, and generates a response.

**State Restoration:** During the postback, ASP.NET retrieves the view state data from the "\_\_VIEWSTATE" hidden field and uses it to restore the state of the page and its controls to their previous values.

**Drawbacks :**

Increased page size (due to the serialized state data being sent back and forth between the client and server)

Potential security risks (as view state can be tampered with by malicious users)

Performance overhead (as view state data must be processed on each request).

To overcome these drawbacks, developers can control the size of viewstate by disabling it for certain controls(in the properties section) or using viewstate compression and viewstate encryption

ASP .net core doesnot use the view state and uses the lighter state management techniques.

**Asp .net: event driven**

Asp .net : server side technology, used for developing dynamic web applications.

Latest version of active server pages( server side scripting language used for developing web applications). Helps to create dynamic web pages for programmers

Provides interactive data driven applications in the internet. Also helps in making applications colourful way

Has two stages:

1. Application life cycle

Starts when user make request

Involves stages like

1. Application start

When user requests an application for accessing a web server, first the request is checked and the request for the access is provided.

1. Object creation

Helps in holding all the information about the cookies and other browser information.

Also holds http context, http request and http response by the browser

1. HTTP application

Holds back all the information sent back from the user by the web driver. If we have two different applications(ex: gaming and social media) then two http applications will be created.

1. Dispose

Manages the unmanaged resources when the objects are no longer needed.

1. Application end

Unload the memory by cleaning the unwanted memory.

1. Page Life cycle
2. Request of page

Page is requested from server.

Page is first compiled, ready the response and send the response to user.

If page is requested first time, compiled and then the response

If page is not first time loaded, it first checks in cache for already compiled version and loads it to send the output.

1. Start of page

Two objects are created -> request and response

Request: goes to server from page

Response: goes to page from server

1. Initialisation of page

Initialisation of all page controls( textbox, checkbox,…).

1. Page loading

Page is loaded with default values.

1. Validation controls

All validation controls like required field validator validate the form values.

Checks the form values and if the value is wrong then shows the error message, if right the data is sent.

1. Handling of postback event

Postback event is triggered when a same page is loaded.

1. Rendering of page

Compilation of response in a way that user can understand

1. Unloading of page

Freeing up of memory( unwanted objects are deleted from memory)

**Page life cycle events**

1. PreInit

Raised after start stage is complete

Checks the IsPostBack event property to determine if this is the first time to load the page. Also IsCallBack and IsCrossBack property is also set

Create or recreate dynamic control

Render the master page dynamically

Set the theme property

Read or set the property values

1. Init

Use this event to initialise the control properties(textbox, button…)

1. InitComplete

At the end of page initialisation stage

Viewstate tracking is enabled here

View state tracking enables controls to persist any values that are programmatically added to the ViewState collection.

Until view state tracking is turned on, any values added to view state are lost across postbacks.

Use this event to make changes to view state that you want to make sure are persisted after the next postback.

1. PreLoad

Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.

1. Load

the Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded.

Use the OnLoad event method to set properties in controls and to establish database connections.

1. Control Events

Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.

1. LoadComplete

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.

1. PreRenderComplete

It can be used to perform data binding operations just before the page is rendered to ensure that the latest data is displayed to the user.

Developers can use the event to finalize the content of the page, such as adding additional controls or modifying the appearance of existing controls.

used for tasks that need to be performed after all other events have been processed and just before the page is displayed to the user.

1. SaveStateComplete

Raised after the viewstate and control state is saved for page and all controls.

But these don’t reflect after next postback

1. Render

Page markup has this method that writes out the control markup to the browser.

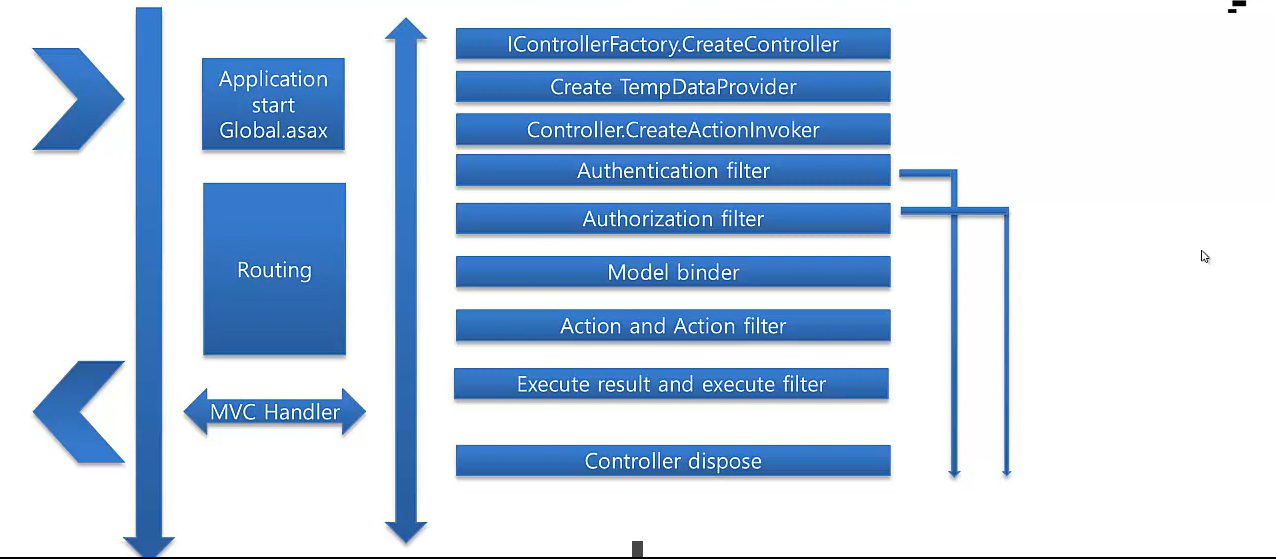
A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code.

1. Unload

Raised for each control and then for the page.

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.

**ASP dotnet MVC life cycle**



**Global.asax** serves as a central location for handling application-level events, defining global configuration settings, and adding custom code to manage the behavior of ASP.NET web applications. It plays a crucial role in the lifecycle of an ASP.NET application and allows developers to customize and extend the application's functionality as needed.

**Routing:**

The ASP.NET MVC framework begins by inspecting the incoming HTTP request to determine which controller and action method should handle it.

Routing is responsible for mapping the URL of the request to a specific controller and action method based on route configuration.

**Controller Initialization:**

Once the appropriate controller and action method have been identified, the MVC framework creates an instance of the corresponding controller class.

The controller's constructor is called, followed by any filters that have been applied to the controller or action method.

**Action Filters**

Action filters are implemented as attributes that can be applied to controller classes or action methods using declarative syntax.

Action filters enable developers to add cross-cutting concerns, such as logging, caching, authentication, authorization, and exception handling, in a modular and reusable way.

**Action Method Execution:**

After the controller has been initialized, the MVC framework invokes the action method associated with the request.

The action method performs any necessary processing, such as retrieving data from a database, manipulating model objects, or invoking other services.

The action method typically returns an ActionResult object, which represents the result of the action (e.g., a view, a JSON response, a redirect).

**Result Execution:**

Once the action method has completed, the MVC framework executes the result returned by the action method.

If the result is a ViewResult, the MVC framework renders the corresponding view, passing any necessary model data to the view.

If the result is a RedirectToRouteResult, the framework performs a redirect to the specified route.

If the result is any other type of ActionResult, the framework executes the result according to its specific behavior (e.g., returning a JSON response, a file download).

**View Rendering:**

If the action result is a ViewResult, the MVC framework renders the associated view using the Razor view engine (or another view engine if configured).

The view combines HTML markup with dynamic content generated from model data to produce the final HTML output.

The rendered HTML is sent back to the client as the HTTP response body.

**Response Finalization:**

Once the view has been rendered and the response has been generated, the MVC framework finalizes the response and sends it back to the client.

Any HTTP headers, cookies, or status codes set during the processing of the request are included in the response.

**Termination:**

After the response has been sent to the client, the controller instance is disposed of, and any resources associated with the request are released.