**Explain ASP.NET MVC Request Life Cycle?**

**Ans.** The detail ASP.NET MVC pipeline is given below:

**1. Routing -** Routing is the first step in ASP.NET MVC pipeline. Typically, it is a pattern matching system that matches the incoming request to the registered URL patterns in the Route Table.

The UrlRoutingModule(System.Web.Routing.UrlRoutingModule) is a class which matches an incoming HTTP request to a registered route pattern in the Route Table(System.Web.Routing.RouteTable).

**2. Controller Initialization -** The MvcHandler initiates the real processing inside ASP.NET MVC pipeline by using ProcessRequest method. This method uses the IControllerFactory instance (default is System.Web.Mvc.DefaultControllerFactory) to create corresponding controller.

**3. Action Execution –** Action execution occurs in the following steps:

 When the controller is initialized, the controller calls its own InvokeAction() method by passing the details of the chosen action method. This is handled by the IActionInvoker.

 After chosen of appropriate action method, model binders(default is System.Web.Mvc.DefaultModelBinder) retrieves the data from incoming HTTP request and do the data type conversion, data validation such as required or date format etc. and also take care of input values mapping to that action method parameters.

 Authentication Filter was introduced with ASP.NET MVC5 that run prior to authorization filter. It is used to authenticate a user. Authentication filter process user credentials in the request and provide a corresponding principal. Prior to ASP.NET MVC5, you use authorization filter for authentication and authorization to a user.

 By default, Authenticate attribute is used to perform Authentication. You can easily create your own custom authentication filter by implementing IAuthenticationFilter.

 Authorization filter allow you to perform authorization process for an authenticated user. For example, Role based authorization for users to access resources.

 By default, Authorize attribute is used to perform authorization. You can also make your own custom authorization filter by implementing IAuthorizationFilter.

 Action filters are executed before (OnActionExecuting) and after (OnActionExecuted) an action is executed. IActionFilter interface provides you two methods OnActionExecuting and OnActionExecuted methods which will be executed before and after an action gets executed respectively. You can also make your own custom ActionFilters filter by implementing IActionFilter. For more about filters refer this article Understanding ASP.NET MVC Filters and Attributes

 When action is executed, it process the user inputs with the help of model (Business Model or Data Model) and prepare Action Result.

**4. Result Execution -** Result execution occurs in the following steps:

 Result filters are executed before (OnResultExecuting) and after (OnResultExecuted) the ActionResult is executed. IResultFilter interface provides you two methods OnResultExecuting and OnResultExecuted methods which will be executed before and after an ActionResult gets executed respectively. You can also make your own custom ResultFilters filter by implementing IResultFilter.

 Action Result is prepared by performing operations on user inputs with the help of BAL or DAL. The Action Result type can be ViewResult, PartialViewResult, RedirectToRouteResult, RedirectResult, ContentResult, JsonResult, FileResult and EmptyResult.

 Various Result type provided by the ASP.NET MVC can be categorized into two category- ViewResult type and NonViewResult type. The Result type which renders and returns an HTML page to the browser, falls into ViewResult category and other result type which returns only data either in text format, binary format or a JSON format, falls into NonViewResult category.

**4.1 View Initialization and Rendering -** View Initialization and Rendering execution occurs in the following steps:

 ViewResult type i.e. view and partial view are represented by IView (System.Web.Mvc.IView) interface and rendered by the appropriate View Engine.

 This process is handled by IViewEngine (System.Web.Mvc.IViewEngine) interface of the view engine. By default ASP.NET MVC provides WebForm and Razor view engines. You can also create your custom engine by using IViewEngine interface and can registered your custom view engine in to your ASP.NET MVC application as shown below:

 Html Helpers are used to write input fields, create links based on the routes, AJAX-enabled forms, links and much more. Html Helpers are extension methods of the HtmlHelper class and can be further extended very easily. In more complex scenario, it might render a form with client side validation with the help of JavaScript or jQuery.

