1. **WHAT IS ACTION METHOD IN MVC?**

**Ans:** All the public methods which are written inside a Controller are known as Action Method. When creating Action Method you must follow these rules.

**a.** Action method must be public.  
**b.** It cannot be overloaded.  
**c.** It cannot be a static method.  
**d.** Every controller has at least one default Action method Index() that returns the view page.  
**e.** ActionResult is a base class of all the result type action methods.

# ActionResult is the base class of all the result type action method. There Are different return types of a controller action method in MVC:

**ViewResult** - Represents HTML and markup.  
**EmptyResult** - Represents no result.  
**RedirectResult** - Represents a redirection to a new URL.  
**JsonResult** - Represents a JavaScript Object Notation result that can be used in an AJAX application.  
**JavaScriptResult** - Represents a JavaScript script.  
**ContentResult** - Represents a text result.  
**FileContentResult** - Represents a downloadable file (with the binary content).  
**FilePathResult** - Represents a downloadable file (with a path).

# **Why MVC**

ASP.NET MVC is a web framework based on Mode-View-Controller (MVC) architecture. Developers can build dynamic web applications using ASP.NET MVC framework that enables a clean separation of concerns, fast development.

**Loose coupling**

ASP.NET MVC is loosely coupled. The logic is separated into 3 parts:

1. Model – Class which contains properties
2. View - Displays the application UI based on the model data
3. Controller – It is a class contains the (logic) methods called as action method.

The three parts can be changed independently without affecting one another

## **ASP.NET MVC - Life Cycle**

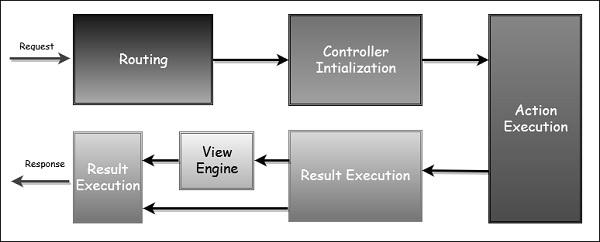
life cycle is simply a series of steps or events used to handle some type of request or to change an application state.  
MVC has two life cycles −

* The application life cycle
* The request life cycle

### The Application Life Cycle

The application life cycle refers to the time at which the application process actually begins running IIS until the time it stops

**The Request life cycle**



Asp.net **Routing** is the first step in MVC request cycle. Basically it is a pattern matching system that matches the request’s URL against the registered URL patterns in the Route Table.

The MVC framework handles converting the route data into a concrete controller that can handle requests. After the controller has been created, the next major step is **Action Execution**. A component called the **action invoker** finds and selects an appropriate Action method to invoke the controller.

After our action result has been prepared, the next stage triggers, which is **Result Execution**. MVC separates declaring the result from executing the result. If the result is a view type, the View Engine will be called and it's responsible for finding and rending our view.

If the result is not a view, the action result will execute on its own. This Result Execution is what generates an actual response to the original HTTP request.

1. **State Management in MVC**

state management techniques that can help us to maintain the data when redirecting from one page to other page or in the same page after reloading. There are several ways to do this in ASP.NET MVC -

1. Hidden Field
2. Cookies
3. Query String
4. ViewData
5. ViewBag
6. TempData

The above objects help us to persist our data on the same page or when moving from “Controller” to “View” or “Controller” to Controller”.

1. **Compare ViewBag, ViewData and TempData**

ViewBag, ViewData, and TempData all are objects in ASP.NET MVC and these are used to pass the data in various scenarios.

The following are the scenarios where we can use these objects.

1. Pass the data from Controller to View.
2. Pass the data from one action to another action in the same Controller.
3. Pass the data in between Controllers.
4. Pass the data between consecutive requests.

ViewBag and ViewData are used to pass the data from Controller action to View and TempData is used to pass the data from action to another action or one Controller to another Controller.

|  |  |  |
| --- | --- | --- |
| ViewData | ViewBag | TempData |
| It is Key-Value Dictionary collection | It is a type object | It is Key-Value Dictionary collection |
| ViewData is a dictionary object and it is property of ControllerBase class | ViewBag is Dynamic property of ControllerBase class. | TempData is a dictionary object and it is property of controllerBase class. |
| ViewData is Faster than ViewBag | ViewBag is slower than ViewData | - |
| ViewData is introduced in MVC 1.0 and available in MVC 1.0 and above | ViewBag is introduced in MVC 3.0 and available in MVC 3.0 and above | TempData is also introduced in MVC1.0 and available in MVC 1.0 and above. |
| ViewData also works with .net framework 3.5 and above | ViewBag only works with .net framework 4.0 and above | TempData also works with .net framework 3.5 and above |
| Type Conversion code is required while enumerating | In depth, ViewBag is used dynamic, so there is no need to type conversion while enumerating. | Type Conversion code is required while enumerating |
| Its value becomes null if redirection has occurred. | Same as ViewData | TempData is used to pass data between two consecutive requests. |
| It lies only during the current request. | Same as ViewData | TempData only works during the current and subsequent request |