**What is a Filter in MVC?**

When user (browser) sends a request to server an action method of a controller gets invoked; sometimes you may require executing a custom code before or after action method gets invoked, this custom code is called as Filter.

**What are the different types of Filters in MVC?**

a. Authorization filter

b. Action filter

c. Result filter

d. Exception filter

[Do not forget the order mentioned above as filters gets executed as per above mentioned sequence]

**How to create a Controller in MVC**

Create a simple class and extend it from Controller class. The bare minimum requirement for a class to become a controller is to inherit it from ControllerBase is the class that is required to inherit to create the controller but Controller class inherits from ControllerBase.

|  |
| --- |
| **What is the difference between ASP.NET MVC and ASP.NET WebForms?**  -ASP.NET WebForms uses the page controller patterns to render a layout. Whereas, ASP.NET MVC provides a model that doesn’t have connection with the View so it becomes easier to test and maintain the applications.   -ASP.NET WebForms uses the Front controller pattern for all the pages to process the web applications requests and used to facilitate routing architecture. Whereas, ASP.NET MVC has the View that is called before the controller and this controller is used to render the View that is based on the actions as the user interacts with the interface.   -ASP.NET WebForms manage the state of the model by using the view state and server based controls. Whereas, ASP.NET MVC doesn’t manage the state information like WebForms.   -ASP.NET WebForms are event driven. Whereas, ASP.NET MVC are test driven. |

|  |
| --- |
| **How ASP.NET MVC differs from ASP.NET Web forms?** ASP.NET MVC 07-15-2012 06:28 AM |
| -ASP.NET MVC does not use View State to record state information whereas ASP.NET web form uses View State to record state information.  -ASP.NET Web forms are better for Rapid application development in the case if you small development team. ASP.NET MVC is better for large project where you give more priority to testability and maintainability.  -In ASP.NET Web form View is called after controller wherein ASP.NET MVC the rendering of View is managed by controller based on the action.  -ASP.NET Web forms development is easy to learn compared to ASP.NET MVC development. |

**What is the Request flow used for ASP.NET MVC framework?**

Request flow handles the request from the clients and passes it to the server. The Request flow is as follows:  
  
-Request is being taken from User to controller.  
  
-Controller processes the request from the user and creates a data Model of that particular request.   
  
-Data model that is being created is then passed to View that handles the frontend or the design.   
  
-View then transforms the Data Model by using its own functions in an appropriate output format.  
  
-The output format that is being given by the View is then gets rendered to the Browser and the View will be seen by the user.

|  |
| --- |
| **Can you describe ASP.NET MVC Request Life Cycle?** ASP.NET MVC Request Life Cycle 07-15-2012 06:31 AM |
| Following are the steps that are executed in ASP.NET MVC Request Life Cycle.  1. Application first receives the request and looks up Route object in RouteTable collection. Then the RouteData object is created.  2. Then RequestContext instance is created.  3. MvcHandler and pass RequestContext to handler is created.  4. Then IControllerFactory from RequestContext is identified.  5. Then the object of class that implements ControllerBase is created.  6. MyController.Execute method is called.  7. Then ControllerActionInvoker finds the action to invoke on the controller and executes that action on the controller and by calling the model associated view returns. |

**What is the main function of URL routing system in ASP.NET MVC?**

-URL routing system provides flexibility to the system and it also enables to define new URL mapping rules that can be used with web applications.   
  
-URL routing system is used to map the application and its routing information gets passed to right controller and action method.   
  
-URL routing system processes and executes the method to run the application without using many designed rules.   
  
-It is used to construct the outgoing URLs that can be used to handle the actions that have the ability to map both incoming and outgoing URLs that adds more flexibility to the application code.   
  
-It follows the rules to execute the application globally and handle the logic that is required for the application.

To add the anti-forgery tokens to a Razor page, use the **HtmlHelper.AntiForgeryToken** helper method:

@using (Html.BeginForm("Manage", "Account")) {

@Html.AntiForgeryToken()

}

# Anti-CSRF and AJAX

The form token can be a problem for AJAX requests, because an AJAX request might send JSON data, not HTML form data. One solution is to send the tokens in a custom HTTP header. The following code uses Razor syntax to generate the tokens, and then adds the tokens to an AJAX request. The tokens are generated at the server by calling **AntiForgery.GetTokens**.

<script>

@functions{

public string TokenHeaderValue()

{

string cookieToken, formToken;

AntiForgery.GetTokens(null, out cookieToken, out formToken);

return cookieToken + ":" + formToken;

}

}

$.ajax("api/values", {

type: "post",

contentType: "application/json",

data: { }, // JSON data goes here

dataType: "json",

headers: {

'RequestVerificationToken': '@TokenHeaderValue()'

}

});

</script>

When you process the request, extract the tokens from the request header. Then call the **AntiForgery.Validate** method to validate the tokens. The **Validate** method throws an exception if the tokens are not valid.

void ValidateRequestHeader(HttpRequestMessage request)

{

string cookieToken = "";

string formToken = "";

IEnumerable<string> tokenHeaders;

if (request.Headers.TryGetValues("RequestVerificationToken", out tokenHeaders))

{

string[] tokens = tokenHeaders.First().Split(':');

if (tokens.Length == 2)

{

cookieToken = tokens[0].Trim();

formToken = tokens[1].Trim();

}

}

AntiForgery.Validate(cookieToken, formToken);

}

**What are sections?**  
Layout pages, can define sections, which can then be overriden by specific views making use of the layout. Defining and overriding sections is optional.

**When using razor views, do you have to take any special steps to proctect your asp.net mvc application from cross site scripting (XSS) attacks?**  
No, by default content emitted using a @ block is automatically HTML encoded to protect from cross site scripting (XSS) attacks.

**In razor syntax, what is the escape sequence character for @ symbol?**  
The escape sequence character for @ symbol, is another @ symbol