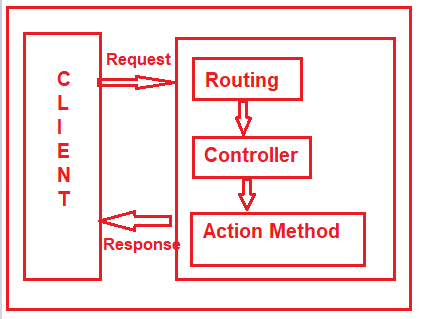
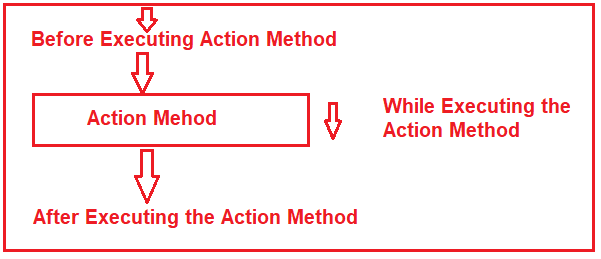
**Filters**



* As of now, we discussed when a client makes a request, then that request comes to the Routing Engine and then the Routing Engine navigates that Request to the Controller.
* The controller then selects the appropriate action method to execute.
* So, it is the Controller action method which is going to handle the incoming request and send the response back to the client who initially made the request



* However, what will you do, if you want to execute some code or logic either before or after the action method executed
* If that is what you want to achieve then you need to use Filters in ASP.NET MVC application.
* The Filters in ASP.NET MVC are the attribute that allows us to inject some logic or code which is going to be executed before or after an action method is invoked
* ASP.NET MVC Filters are used to perform the following common functionalities in your application.

1. Caching
2. Logging
3. Error Handling
4. Authentication and Authorization, etc

* The ASP.NET MVC5 framework provides five different types of Filters. They are as follows

1. Authentication Filter (Introduced in MVC 5)
2. Authorization Filter
3. Action Filter
4. Result Filter
5. Exception Filter

* This is also the order of the execution of Filters if more than one filter is applied. But the point that you need to remember is the Exception Filter can be executed at any point in time when there is an unhandled exception occurred in your application
* **Can we Create Custom Filters in MVC?**

Yes, you can create custom filters in MVC. If the built-in filters do not serve your purpose then you can create your own custom filter as per your business requirements. You can create the Custom Filter for all the above five different categories of Filters

* You can configure the filters at three different levels of our application. They are as follows

1. Global Level
2. Controller Level
3. Action Level

###### **Global Level:**

* Here you need to register the Filter within the **Application\_Start()** method of **Global.asax.cs** file as shown below. As we know this is the first method of our application which is going to be executed when the application starts

Protected void Application\_Start()

{

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

}

* **Note:** When you register a filter at the Global level, then it is applicable to all the Action Methods of all the Controllers of your MVC application

###### **Controller Level:**

Here you need to apply the filter at the top of the controller name as shown below.

[Authorize(Roles = "Admin")]

public class AdminController : Controller

{

//Code

}

* **Note**: When you apply the filter at the Controller level, then it is applicable to all the action methods of that controller only

###### **Action Level:**

Here you need to apply the filter on the top of the action method name as shown below.

public class UserController : Controller

{

[Authorize(Users = "User1,User2")]

public ActionResult LinkToLogin(string provider)

{

// Code

return View();

}

}

**Note:** When you apply the filter to a particular action method, then it is only applicable to that particular action method

**Authorization Filters**

* This filter is used to check whether the user has the rights to access the particular resource or page.
* The built-in AuthorizeAttribute and RequireHttpsAttribute are examples of Authorization Filters.
* The Authorization Filters in MVC implements the IAuthorizationFilter interface
* Two features will be there 1. Allow Anonymous 2. Authorize
* You can assign users into roles and according to that restriction you can apply different access roles for different user to controller actions or controller itself.

**[Authorize(Users = "Betty, Johnny")]**

**public ActionResult SpecificUserOnly()**

**{**

**return View();**

**}**

or you can restrict according to group

**[Authorize(Roles = "Admin, Super User")]**

**public ActionResult AdministratorsOnly()**

**{**

**return View();**

**}**

##### ****Action Filters****

* The Action Filter will be executed before the action method starts executing or after the action has executed.
* if you want to execute some custom logic that should be get executed before and after an action method executes, then you need to use the Action Filters in MVC applications.
* The Action filters implement the IActionFilter interface that has two methods OnActionExecuting and OnActionExecuted

##### Result Filter

* The Result filters in MVC application are executed before or after generating the result for an action.
* Action Result type can be ViewResult, PartialViewResult, RedirectToRouteResult, RedirectResult, ContentResult, JsonResult, FileResult and EmptyResult which derives from the ActionResult abstract class.
* Result filters are called after the Action filters

##### Exception Filter

* The Exception filters are executed when there is an unhandled exception occurs during either the execution of actions or filters. The in-built [HandleErrorAttribute](https://dotnettutorials.net/lesson/exception-filter-mvc/)class is an example of Exception Filters.
* The IExceptionFilter interface is used to create [Custom Exception Filter](https://dotnettutorials.net/lesson/custom-exception-filter-mvc/) which provides the OnException method, which will be executed when there is an unhandled exception occurs during the actions or filters execution

##### Custom filter creation

* We cannot do the following things using the built-in HandleErrorr attribute
* We cannot log the Exception using the HandleError attribute.
* It is not possible to handle the exceptions raised outside the controllers.
* Exception Handling is not possible Based on Scenario

##### How to do it in custom filter

* Create table for exception logging
* Add web config in following code : customErrors mode="On"></customErrors>
* Registering the Custom Exception Filter in **FilterConfig** class in App\_start
* The **Application\_Start()** method of **Global.asax** file is the first method that is going to be executed when the application starts. So, from here we need to call the **RegisterGlobalFilters()**
* Create custom view in views
* Create customfilter class in model and write coding for insert log into DB
* Call the created customclass in Controller like controller wise or action method wise