**Custom Action Filter in ASP.NET MVC 5**

ASP.NET MVC 5 provides five different kinds of Filters. They are as follows:

1. Authentication [Introduced in MVC5]
2. Authorization
3. Action
4. Result
5. Exception

Filters are used to inject logic at the different levels of request processing. Let us understand where at the various stages of request processing different filters get applied.

* Authentication filter runs before any other filter or action method
* Authorization filter runs after Authentication filter and before any other filter or action method
* Action filter runs before and after any action method
* Result filter runs before and after execution of any action result
* Exception filter runs only if action methods, filters or action results throw an exception

I have tried to show the filter execution timing in context of request processing in the below diagram:  
[](http://blog.falafel.com/wp-content/uploads/2014/10/customfilterimg1.png)  
**Action Filter**   
An action filter consists of codes that run either before or after an action runs. It can be used for tasks like logging, privileged based authorization, authentication, caching etc.  
Creating a custom action filter is very easy. It can be created in four simple steps:

1. Create a class
2. Inherit **ActionFilterAttribute** class
3. Override the **OnActionExecuting** method to run logic before the action method
4. Override the **OnActionExecuted** method to run logic after the action method

Let us see how we can create a custom action filter. The purpose of the action filter is to find whether a logged in user belongs to a particular privileges or not. On the basis of the result, a user will access a particular action or navigate to the login action. To do this, I have created a class and extended the ActionFilterAttribute class.

|  |
| --- |
| **using PrivilegeDemo.Services;**  **using System.Web;**  **using System.Web.Mvc;**  **using Microsoft.AspNet.Identity;**  **using System.Web.Routing;**  **namespace PrivilegeDemo.Filters**  **{**  **public class AuthorizationPrivilegeFilter : ActionFilterAttribute**  **{**  **public override void OnActionExecuting(ActionExecutingContext filterContext)**  **{**  **AuthorizationService \_authorizeService = new AuthorizationService();**  **string userId = HttpContext.Current.User.Identity.GetUserId();**  **if (userId != null)**  **{**  **var result = \_authorizeService.CanManageUser(userId);**  **if (!result)**  **{**  **filterContext.Result = new RedirectToRouteResult(**  **new RouteValueDictionary{{ "controller", "Account" },**  **{ "action", "Login" }**  **});**  **}**  **}**  **else**  **{**  **filterContext.Result = new RedirectToRouteResult(**  **new RouteValueDictionary{{ "controller", "Account" },**  **{ "action", "Login" }**  **});**  **}**  **base.OnActionExecuting(filterContext);**  **}**  **}**  **}** |

As you see in the above code, the OnActionExecuting method is overridden because we want the code to execute before the action method gets executed. Once the action filter is created, it can be used in three ways:

1. As Global filter
2. As Controller
3. As Action

By adding a filter to the global filter in **App\_Start\FilterConfig** it will be available globally to the entire application.

|  |
| --- |
| **public class FilterConfig**  **{**  **public static void RegisterGlobalFilters(GlobalFilterCollection filters)**  **{**  **filters.Add(new HandleErrorAttribute());**  **filters.Add(new AuthorizationPrivilegeFilter());**  **}**  **}** |

By adding a filter to a particular Controller it will also be available to the all actions of that particular controller.

|  |
| --- |
| **[AuthorizationPrivilegeFilter]**  **public class HomeController : Controller**  **{** |

By adding a filter to a particular action it will be available to the particular action.

|  |
| --- |
| **[AuthorizationPrivilegeFilter]**  **public ActionResult About()**  **{** |

**Conclusion**

In this post we learned about custom action filters. We create custom action filters to run logic either before or after the action gets executed.