Previous Next

ASP.NET MVC - Action Filters

In the previous section, you learned about filters in MVC. In this section, you will learn about another filter type called **Action Filters** in ASP.NET MVC.

Action filter executes before and after an action method executes. Action filter attributes can be applied to an individual action method or to a controller. When action filter applied to controller then it will be applied to all the action methods in that controller.

OutputCache is a built-in action filter attribute that can be apply to an action method for which we want to cache the output. For example, output of the following action method will be cached for 100 seconds.

Example: ActionFilter

```
[OutputCache(Duration=100)]
public ActionResult Index()
{
    return View();
}
```

You can create custom action filter for your application. Let's see how to create custom action filters.

Custom Action Filter

You can create custom action filter by two ways. First, by implementing IActionFilter interface and FilterAttribute class. Second, by deriving ActionFilterAttribute abstract class.

IActionFilter interface include following methods to implement:

- void OnActionExecuted(ActionExecutedContext filterContext)
- void OnActionExecuting(ActionExecutingContext filterContext)

ActionFilterAttribute abstract class includes the following methods to override:

- void OnActionExecuted(ActionExecutedContext filterContext)
- void OnActionExecuting(ActionExecutingContext filterContext)
- void OnResultExecuted(ResultExecutedContext filterContext)
- void OnResultExecuting(ResultExecutingContext filterContext)

As you can see that ActionFilterAttribute class has four methods to overload. It includes OnResultExecuted and OnResultExecuting methods, which can be used to execute custom logic before or after result executes. Action filters are generally used to apply cross-cutting concerns such as logging, caching, authorization etc.

Consider the following custom Log filter class for logging.

Example: Custom ActionFilter for Logging

```
public class LogAttribute : ActionFilterAttribute
{
```

```
    public override void OnActionExecuted(ActionExecutedContext filterContext)
    {
        Log("OnActionExecuted", filterContext.RouteData);
    }

    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        Log("OnActionExecuting", filterContext.RouteData);
    }

    public override void OnResultExecuted(ResultExecutedContext filterContext)
    {
        Log("OnResultExecuted", filterContext.RouteData);
    }

    public override void OnResultExecuting(ResultExecutingContext filterContext)
    {
        Log("OnResultExecuting ", filterContext.RouteData);
    }

    private void Log(string methodName, RouteData routeData)
    {
        var controllerName = routeData.Values["controller"];
        var actionName = routeData.Values["action"];
        var message = String.Format("{0}- controller:{1} action:{2}", methodName,
                                                                      controllerName,
                                                                      actionName);

        Debug.WriteLine(message);
    }
}
```

As you can see, Log class derived ActionFilterAttribute class. It logs before and after action method or result executes. You can apply Log attribute to any Controller or action methods where you want to log the action. For example, by applying Log attribute to Controller, it will log each action methods of that controller.

Example: Apply Log ActionFilter to Controller

```
[Log]
public class StudentController : Controller
{
    public ActionResult Index()
    {
        return View();
    }

    public ActionResult About()
    {
        return View();
    }

    public ActionResult Contact()
    {
        return View();
    }
}
```

The above example will show following output in the output window of Visual Studio on *http://localhost/student* request.

Output:

```
OnActionExecuting- controller:Home action:Index
OnActionExecuted- controller:Home action:Index
OnResultExecuting - controller:Home action:Index
OnResultExecuted- controller:Home action:Index
```

Points to Remember :

1) Action filters allow pre and post processing logic to be applied to an action method.

2) Action filters are generally used to apply cross-cutting concerns such as logging, caching, authorization etc.

3) Action filter can be registered as other filters at global, controller or action method level.

4) Custom action filter attribute can be created by deriving ActionFilterAttribute class or implementing IActionFilter interface and FilterAttribute abstract class.

5) Every action filter must override OnActionExecuted, OnActionExecuting, OnResultExecuted, OnResultExecuting methods.

Previous

Next

TUTORIALSTEACHER.COM

feedback@tutorialsteacher.com

TUTORIALS

- ASP.NET Core
- ASP.NET MVC
- IoC
- AngularJS 1
- Node.js
- D3.js