Action Filters: Lab 1 – Getting Started With Action Filters

Lab Setup: Open the MVCDemo solution file in the Lab-3-Exercise-1-Complete folder and review the contents

In this lab we're going to create a simple ActionFilter to demonstrate how they work and how to implement them.

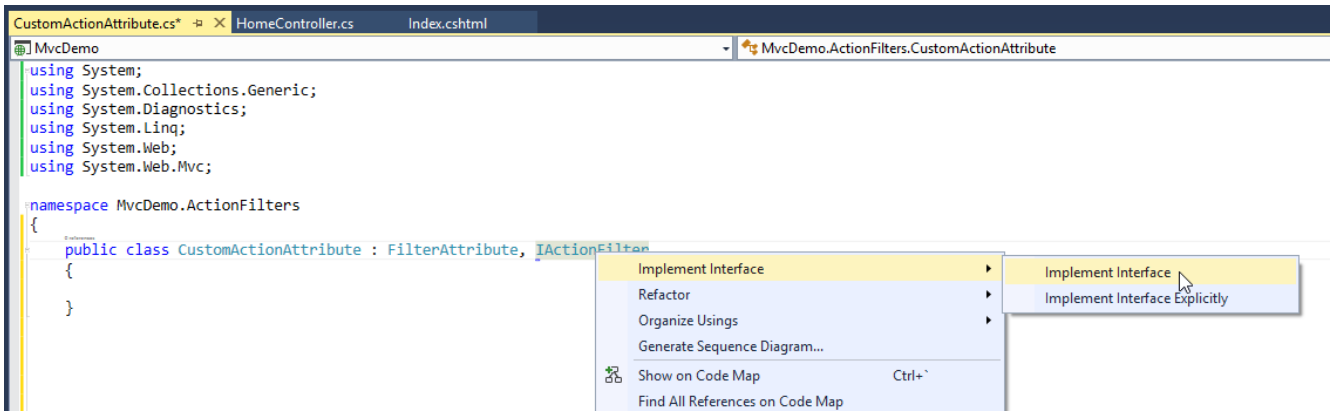
1. Add a new folder named Filters.
2. Add a new CustomActionAttribute class in the newly created Filters folder.
3. Specify that the CustomActionAttribute class inherits from the FilterAttribute base class and implements the IActionFilter interface (You will need to add a using reference to System.Web.Mvc). At this point your code should look like this:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcDemo.ActionFilters
{
    public class CustomActionAttribute : FilterAttribute, IActionFilter
    {

    }
}
```

4. Right click in IActionFilter and select Implement Interface from the shortcut menus.



5. Review you code. It should look like this:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcDemo.ActionFilters
{
   public class CustomActionAttribute : FilterAttribute, IActionFilter
   {
     public void OnActionExecuted(ActionExecutedContext filterContext)
     {
          throw new NotImplementedException();
     }

     public void OnActionExecuting(ActionExecutingContext filterContext)
     {
          throw new NotImplementedException();
     }
   }
}
```

6. The first ActionFilter we're going to create will be very simple and won't do anything particularly useful but you will gain an understanding of the pieces involved and how they interact. Replace the throw command in the OnActionExecuted method with the bolded code below and remove the throw command from the OnActionExecuting method. This ActionFilter will result in a 404 error if the action is run on your local machine. Your code should look like this now:

```
.    public class CustomActionAttribute : FilterAttribute, IActionFilter

  {
     public void OnActionExecuted(ActionExecutedContext filterContext)
     {
          if (filterContext.HttpContext.Request.IsLocal)
          {
              filterContext.Result = new HttpNotFoundResult();
          }
     }

     public void OnActionExecuting(ActionExecutingContext filterContext)
     {

     }
  }
```

7. Save the file.

8. Now let's test it by adding a new FilterTest method to the HomeController and annotate it with the new CustomAction filter we created (see the bolded code at the bottom of the controller). Your HomeController should look something like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MvcDemo.Models;
using MvcDemo.ActionFilters;

namespace MvcDemo.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            var meetingRepository = new MockMeetingRepository();
            List<Meeting> meetings = meetingRepository.GetAll();

            return View(meetings);
        }

        [CustomAction]
        public string FilterTest()
        {
            return "This is the FilterTest action";
        }
    }
}
```

9. Save the file.
10. That should do it for now. When the FilterTest action is called it should generate a 404 error on your machine. Try it by running the project and changing the URL line to http://localhost:63524/Home/FilterTest. (Obviously your localhost port will be different).

Action Filters: Lab 2 – Timing the Execution of an Action

Lab Setup: Continue working with the current project

In this lab we're going to create and ActionFilter to calculate how long it takes to complete an action.

1. If it's not already open, open the CustomActionAttribute.cs class file.
2. Directly below the CustomActionAttribute class created in the prior lab, add the following code to create the ProfileActionAtribute class.

```
public class ProfileActionAttribute : FilterAttribute, IActionFilter
{

}
```

3. As you did in Step 4 of the prior lab, right click on IActionFilter and select the Implement Interface menu options to add the two methods.
4. Remove the throw commands from the two methods and your code should look like this:

```
public class ProfileActionAttribute : FilterAttribute, IActionFilter
{
   public void OnActionExecuted(ActionExecutedContext filterContext)
   {

   }

   public void OnActionExecuting(ActionExecutingContext filterContext)
   {

   }
}
```

5. Add a timer variable to the class above the OnActionExecuted method (you will need to add `using System.Diagnostics;`).

```
private Stopwatch timer;
```

6. Add the following line to the OnActionExecuting method to start the timer.

```
timer = Stopwatch.StartNew();
```

7. Add the following code to the OnActionExecuted method to stop the timer and display the results.

```
timer.Stop();
if (filterContext.Exception == null)
{
   // This line should be all one line.
      filterContext.HttpContext.Response.Write(
      string.Format("<div>Action method elapsed time: {0:F6}</div>",
      timer.Elapsed.TotalSeconds));
}
```

8. Review your code for the ProfileActionAttribute. It should look like this:

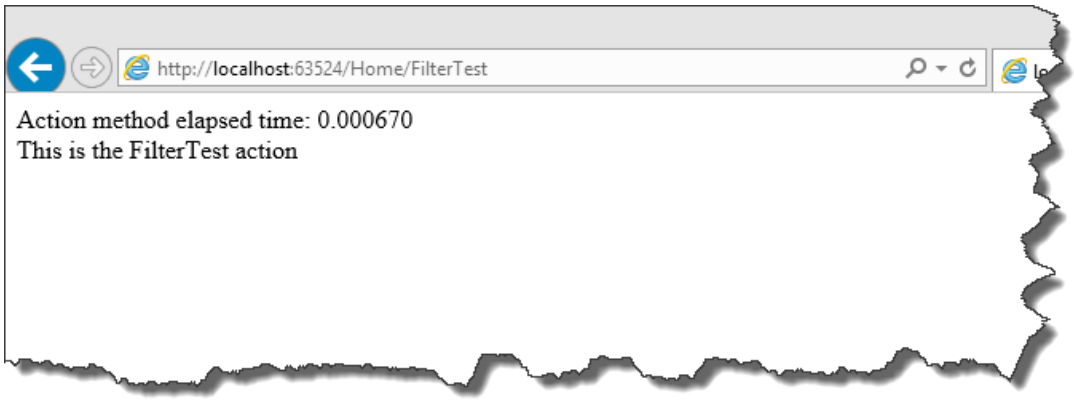
```
public class ProfileActionAttribute : FilterAttribute, IActionFilter
{
      private Stopwatch timer;
      public void OnActionExecuted(ActionExecutedContext filterContext)
      {
          timer.Stop();
          if (filterContext.Exception == null)
          {
              filterContext.HttpContext.Response.Write(
   string.Format("<div>Action method elapsed time: {0:F6}</div>",
   timer.Elapsed.TotalSeconds));
          }
      }

      public void OnActionExecuting(ActionExecutingContext filterContext)
      {
          timer = Stopwatch.StartNew();
      }
}
```

9. Now it's time to test it. Open the HomeController.cs file and replace the [CustomAction] with [ProfileAction]. Your code should look like this:

```
      [ProfileAction]
      public string FilterTest()
      {
          return "This is the FilterTest action";
      }
```

10. Save the file.
11. When the FilterTest action is called it should display the timing results. Try it by running the project and changing the URL line to http://localhost:63524/Home/FilterTest. (Obviously your localhost port will be different).



The End!

That should give you enough to chew on for now. What other uses could you use ActionFilters for?