# Logging Application Utilization of an ASP.NET MVC 5 Web Application

John F. Gnazzo

## Introduction

During development of an ASP.NET MVC 5 Web application, the customer needs to understand and determine who is using the application and how they were using it. To assess how users are using the application, one would have to instrument the application in such a way as to collect information every time a controller action method has been called.

This blog will discuss how such a web application can be *easily* instrumented using the open source Log4net logging API and an ASP.NET MVC 5 Action Filter.

## Log4Net

The Apache log4net library is a tool to help a programmer output log statements to a variety of output targets. Common log targets include the system console, text file, or database. The Log4Net library can been added to any ASP.NET MVC 5 Web application through NuGet.

NuGet client tools provide the ability to easily integrate open source libraries such as Log4Net into the application..

## Action Filters

In ASP.NET MVC, controllers define action methods that usually have a one-to-one relationship with possible user interactions, such as clicking a link, or submitting a form. For example, when a user clicks a link, a request is routed to the designated controller, and the corresponding action method is called.

Sometimes you want to perform logic either before an action method is called or after an action method runs. To support this, ASP.NET MVC provides action filters. Action filters are pieces of code that provide a declarative means to add pre-action and post-action behavior to controller action methods.

## The Code

The following code shows an example of an Authentication Action Filter and how it can be instrumented to log application utilization information. This code sample is executed every time a user performs an action and will write a log entry every time a user performs the action.

The piece of code that actually does the logging is highlighted in yellow. Specific Authentication logic has been omitted for brevity.

```
public class AuthenticationFilterAttribute : FilterAttribute, IAuthenticationFilter
{
   public void OnAuthentication(AuthenticationContext filterContext)
   {
      // Authentication Code goes here.

      LogActions(filterContext);
   }
```

```
    private static void LogActions(AuthenticationContext filterContext)
    {
        var descriptor = filterContext.ActionDescriptor;
        var actionName = descriptor.ActionName;
        var controllerName = descriptor.ControllerDescriptor.ControllerName;

        Api.Log.Info(string.Format("{0} - {1} - {2}", controllerName, actionName,
            HttpContext.Current.Session["LoginId"].ToString().ToUpper()));
    }

    public void OnAuthenticationChallenge(AuthenticationChallengeContext filterContext)
    {
        // Do Nothing Here
    }
}
```

## Summary

This article has discussed how an ASP.NET MVC application can be easily instrumented with Log4Net and how the application can be set up to log all user actions through an action filter. This will help the business understand and assess who is using an application and how they are using it.