using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.Extensions.Logging;

using NLog;

using Serilog.Core;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Logger = NLog.Logger;

namespace LoggingNetCoreDemo

{

public class LoggingActionFilter : IActionFilter

{

Microsoft.Extensions.Logging.ILogger \_logger;

public LoggingActionFilter(ILoggerFactory loggerFactory)

{

\_logger = loggerFactory.CreateLogger<LoggingActionFilter>();

loggerFactory.AddFile("LogsDemo/app-{Date}.txt");

}

public void OnActionExecuting(ActionExecutingContext context)

{

// do something before the action executes

\_logger.LogInformation($"Action '{context.ActionDescriptor.DisplayName}' executing");

}

public void OnActionExecuted(ActionExecutedContext context)

{

// do something after the action executes

\_logger.LogInformation($"Action '{context.ActionDescriptor.DisplayName}' executed");

}

}

}

Startip.cs

public void ConfigureServices(IServiceCollection services)

{

services.AddSingleton<LoggingActionFilter>();

services.AddMvc(options =>

{

options.Filters.Add<LoggingActionFilter>();

});

services.AddControllersWithViews();

}

In controller

[ServiceFilter(typeof(LoggingActionFilter))]

public IActionResult Index()

{

try

{

int x = 10;

int y = 0;

int res = x / y;

}

catch (Exception ex)

{

\_logger.LogInformation("Info");

\_logger.LogWarning("Warning");

\_logger.LogCritical(ex.Message);

\_logger.LogDebug("This is a message");

}

return View();

}

<https://dzone.com/articles/customizing-aspnet-core-part-09-actionfilter>