## Laboratorul 9 – Guster Andreea

Pentru rezolvarea acestui laborator ne vom folosi de proiectul din laboratorul numarul 8.

## Types of filters in ASP .NET

1. **Authorization filters** – Implementeaza atributul **IAuthorizationFilter**.
2. **Action filters** – Implementeaza atributul **IActionFilter**.
3. **Result filters** – Implementeaza atributul **IResultFilter**.
4. **Exception filters** – Implementeaza atributul **IExceptionFilter**.

Filtrele sunt executate in ordinea enumerata mai sus. De exemplu, **authorization filters** sunt intotdeauna executate inainte de **action filters**, iar **exception filters** sunt executate dupa orice alt tip de filtru.

**Authorization filters** sunt folosite pentru a implementa autentificarea si autorizarea pentru actiunile controller-elor. De exemplu `[Authorize(Roles="User")].`

**Action filters** contin logica executata inainte si dupa ce este apelata o actiune a unui contoller. De exempu, se pot folosi pentru a modifica datele unui view returnate de o actiune dintr-un controller.

**Result filters** contin logica executata inainte si dupa ce rezultatul unui view este executat. De exemplu, se poate folosi pentru a modifica un view result chiar inainte ca acesta sa ii fie afisat de catre browser.

**Exception filters** sunt exceptiile care sunt parsate ultimele. Acestea pot fi folosite pentru a trata erorile aruncate de actiunile controller-elor sau de rezultatele actiunilor controller-elor. De asemenea, le putem utiliza si pentru a afisa log-urile erorilor.

Un filter este un atribut. Majoritatea filtrelor pot fi aplicate pe:

- O actiune individuala din cadrul unui controller
- Un controller intreg

## ActionFilter and Output Caching

**PASUL 1** . In controller-ul **Books** de tipul MVC vom scrie inaintea antetului actiunii **Details** un filtru de actiune numit OutputCache.

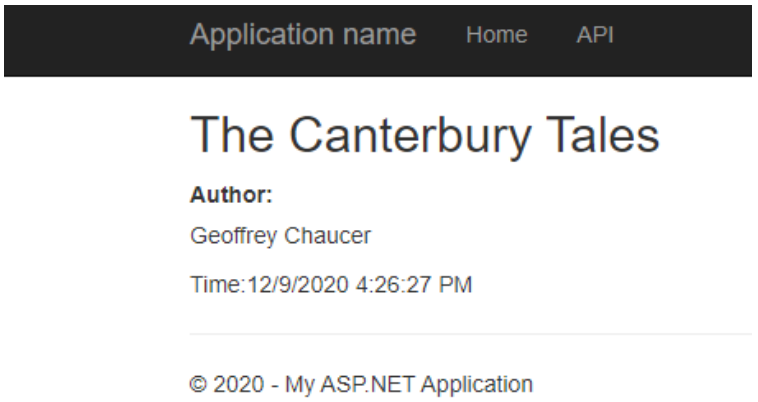
**!! ATENTIE !!** Filtrul de actiune OutputCache stocheaza in memoria cache datele returnate de actiune timp de 10 secunde.

```
[OutputCache(Duration = 10)]
public ActionResult Details(int? id)
{

    if (id.HasValue)
    {
        Book book = db.Books.Find(id);
        if (book != null)
        {
            return View(book);
        }
        return HttpNotFound("Couldn't find the book with id " + id.ToString() + "!");
    }
    return HttpNotFound("Missing book id parameter!");
}
```

**PASUL 2**. In **Views/Details** adaugati urmatoare linie de cod care afiseaza data curenta.

```
<p>Time:<span>@DateTime.Now.ToString()</span></p>
```




**PASUL 3**. Rulati aplicatia si observati ca de fiecare data cand invocam ruta https://localhost:44377/Books/Details/1 din bara de adresa a browser-ului sau cand apasam pe butonul de refresh de mai multe ori, pe ecran o sa ne afiseze aceeasi data timp de 10 secunde. Output-ul metodei Detials este stocat in memoria cache timp de 10 secunde.

**!! ATENTIE !!** Puteti aplica mai multe atribute de tipul ActionFilter pe aceeasi actiune.

## Crearea unui filtru

In **Models** vom crea o clasa noua numita **TheFilter** care mosteneste `ActionFilterAttribute` din namespace-ul `using System.Web.Mvc;`. In acesta clasa vom suprascrie urmatoarele metode ce vor juca rol de evenimente:

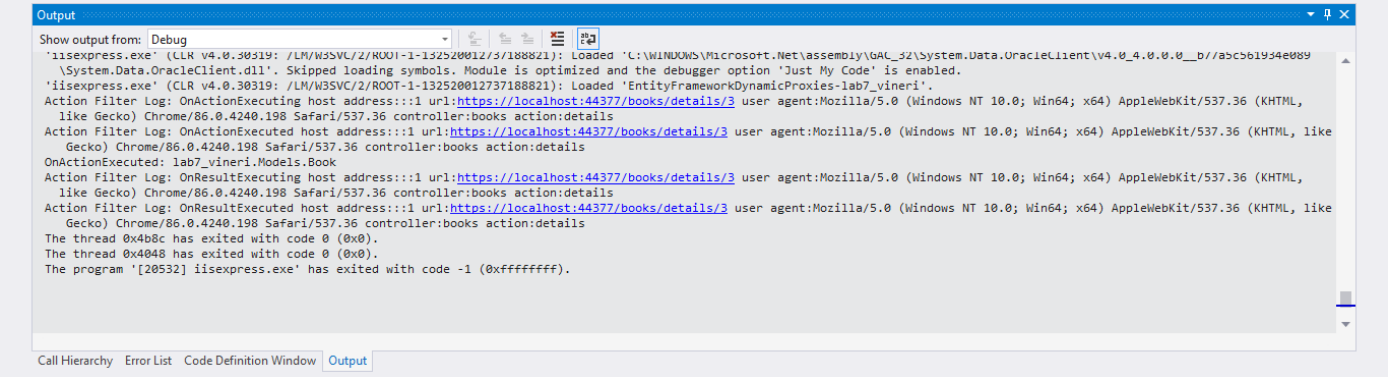
- `OnActionExecuting(ActionExecutingContext filterContext)` - Inainte de rularea actiunii.
- `OnActionExecuted(ActionExecutedContext filterContext)` - Imediat dupa rularea actiunii careia ii este aplicat filtrul. Aici putem vedea si modelul transmis view-ului:

```
((ViewResultBase)filterContext.Result).Model
```

- `OnResultExecuting(ResultExecutingContext filterContext)` - Inainte de rularea ActionResult-ului corespunzator.
- `OnResultExecuted(ResultExecutedContext filterContext)` - Imediat dupa rularea ActionResult-ului corespunzator.

Pentru fiecare din cele 4 metode de mai sus vrem sa afisam in consola log-uri ce contin informatii legate de apelarea actiunilor. În namespace-ul `System.Diagnostics` exista obiectul Debug, cu care putem sa tiparim mesaje în consola de debug, în modul urmator (doar când modul Debug este pornit, adica atunci când rulam cu F5):

```
Debug.WriteLine("mesaj");
```




In **Models/TheFilter.cs**:

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

namespace lab7_vineri.Models
{
    public class TheFilter : ActionFilterAttribute
    {
        public override void OnActionExecuting(ActionExecutingContext filterContext) {
            Log("OnActionExecuting", filterContext.RouteData, filterContext.HttpContext.Request);
        }

        public override void OnActionExecuted(ActionExecutedContext filterContext) {
            var model = ((ViewResultBase)filterContext.Result).Model;

            Log("OnActionExecuted", filterContext.RouteData, filterContext.HttpContext.Request);
            Debug.WriteLine(model.ToString(), "Model Type");
        }

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

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

        // vom crea metoda Log care ne afiseaza in consola de debug un mesaj
        private void Log(string methodName, RouteData routeData, HttpRequestBase request)
        {
            var controllerName = routeData.Values["controller"];
            var actionName = routeData.Values["action"];

            var url = request.Url;
            var hostAddress = request.UserHostAddress;
            var agent = request.UserAgent;

            var message = String.Format("{0} host address:{1} url:{2} user agent:{3} controller:{4}
action:{5} ",
                methodName, hostAddress, url, agent,controllerName, actionName);

            Debug.WriteLine(message, "Action Filter Log");
        }
    }
}
```

In **Controllers/BooksController** de tipul MVC:

```
using lab7_vineri.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Runtime.Caching;

namespace lab7_vineri.Controllers.subfolder
{
    public class BooksController : Controller
    {
        private DbCtx db = new DbCtx();

        [HttpGet]
        public ActionResult Index()
        {
          /*  List<Book> books = db.Books.ToList();
            ViewBag.Books = books;*/
            return View();
        }



        [OutputCache(Duration = 10)]
        [TheFilter]
        public ActionResult Details(int? id)
        {

            if (id.HasValue)
            {
                Book book = db.Books.Find(id);
                if (book != null)
                {
                    return View(book);
                }
                return HttpNotFound("Couldn't find the book with id " + id.ToString() + "!");
            }
            return HttpNotFound("Missing book id parameter!");
        }

        [HttpGet]
        [TheFilter]
        public ActionResult New()
        {
            Book book = new Book();
            return View(book);
        }

       /* [HttpPost]
        public ActionResult New(Book bookRequest)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    db.Books.Add(bookRequest);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                return View(bookRequest);
            }
            catch (Exception e)
            {
                return View(bookRequest);
            }
```

```
        }*/

        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id.HasValue)
            {
                Book book = db.Books.Find(id);

                if (book == null)
                {
                    return HttpNotFound("Coludn't find the book with id " + id.ToString() + "!");
                }
                return View(book);
            }
            return HttpNotFound("Missing book id parameter!");
        }

       /* [HttpPut]
        public ActionResult Edit(int id, Book bookRequest)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    Book book = db.Books
                        .SingleOrDefault(b => b.BookId.Equals(id));

                    if (TryUpdateModel(book))
                    {
                        book.Title = bookRequest.Title;
                        book.Author = bookRequest.Author;
                        db.SaveChanges();
                    }
                    return RedirectToAction("Index");
                }
                return View(bookRequest);
            }
            catch (Exception)
            {
                return View(bookRequest);
            }
        }*/

       /* [HttpDelete]
        public ActionResult Delete(int id)
        {
            Book book = db.Books.Find(id);
            if (book != null)
            {
                db.Books.Remove(book);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return HttpNotFound("Couldn't find the book with id " + id.ToString() + "!");
        }*/
    }
}
```