
epam
ASP.NET MVC BASICS:
FILTERS
2017

FILTERS

Фильтры описывают блоки функциональности, вызываемой во **время/до/после** наступления определенного события.

Базовые фильтры, предоставляемые в **`System.Web.Mvc`**, реализованы как **атрибуты**, что позволяет уменьшить объем кода, необходимых для их использования и выполнения.

Данные атрибуты могут применяться как ко всему классу, так и к отдельным его методам.

```
[Authorize]
public class AccountController : Controller
{
    ...
}
```

FILTER TYPES

Тип	Интерфейс	Реализация	Описание
Фильтры **авторизации**	`IAuthorizationFilter`	`[AuthorizeAttribute]` `[AllowAnonymousAttribute]`	Фильтр, определяющий, аутентифицирован ли клиент и имеет ли он доступ к данному ресурсу. Данный фильтр запускается до выполнения любого другого фильтра или action.
Action фильтры	`IActionFilter`	`[ActionFilterAttribute]`	Фильтр, применяемый к action. Может запускаться как до, так и после выполнения.
ActionResult фильтры	`IResultFilter`	`[ResultFilterAttribute]`	Фильтр, применяемый к action result. Может запускаться как до, так и после выполнения.
Фильтры **исключений**	`IExceptionFilter`	`[HandleErrorAttribute]`	Атрибут для обработки исключений, выбрасываемых методом действий и результатом действий

AUTHORIZATION FILTERS

Фильтры авторизации отрабатывают до запуска остальных фильтров и вызова методов действий.
Они реализуют интерфейс `IAuthorizationFilter`

Если атрибут `[Authorize]` применен к классу, то для методов, для которых необходим анонимный доступ, необходимо использовать атрибут `[AllowAnonymous]`:

```
[Authorize(Roles="admin, moderator")]
public class AdminController : Controller {
    [AllowAnonymous]
    public ActionResult Login() { ... }

    [HttpPost]
    [AllowAnonymous]
    public ActionResult Login(LoginViewModel viewModel) { ... }
}
```

EXCEPTION FILTERS

Все фильтры исключений должны реализовывать интерфейс **IExceptionFilter**:

```
public interface IExceptionFilter
{
    void OnException(ExceptionContext filterContext);
}
```

Фильтры исключений позволяют отследить не только исключения, возникающие в самом методе, но исключения, генерируемые результатами действий.

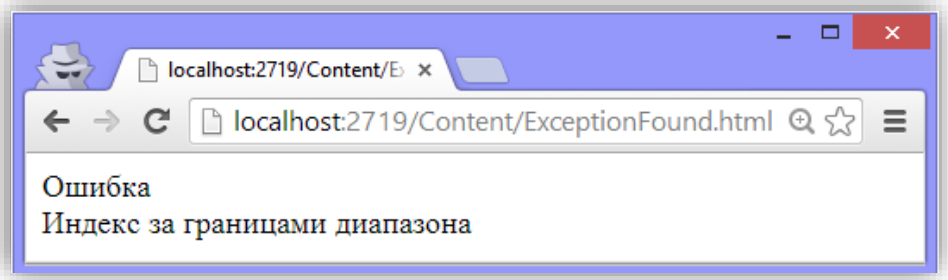
Если вдруг приложение выбрасывает необрабатываемое исключение, то фильтр вызывает **метод OnException()**

С помощью **свойства Exception** мы можем получить доступ к выбрасываемому исключению.

Установив **свойство ExceptionHandled** в **true**, фильтр тем самым помечает исключение как обработанное.

EXCEPTION FILTERS

```
[MyIndexException]
public ActionResult TestException() {
     var array = new int[2];
     array[6] = 4;
     return View();
}
```




```
using System.Web.Mvc;
public class MyIndexException : FilterAttribute, IExceptionFilter {
     public void OnException(ExceptionContext exceptionContext)
     {
          if (!exceptionContext.ExceptionHandled &&
               exceptionContext.Exception is IndexOutOfRangeException)
          {
               exceptionContext.Result =
                    new RedirectResult("/Content/ExceptionFound.html");

               exceptionContext.ExceptionHandled = true;
          }
     }
}
```

ACTION FILTERS

Action фильтры позволяют нам проконтролировать входной контекст запроса при доступе к **action**, а также выполнить определенные действия по завершению работы **action**:

```
public interface IActionFilter
{
    void OnActionExecuting(ActionExecutingContext filterContext);
    void OnActionExecuted(ActionExecutedContext filterContext);
}
```

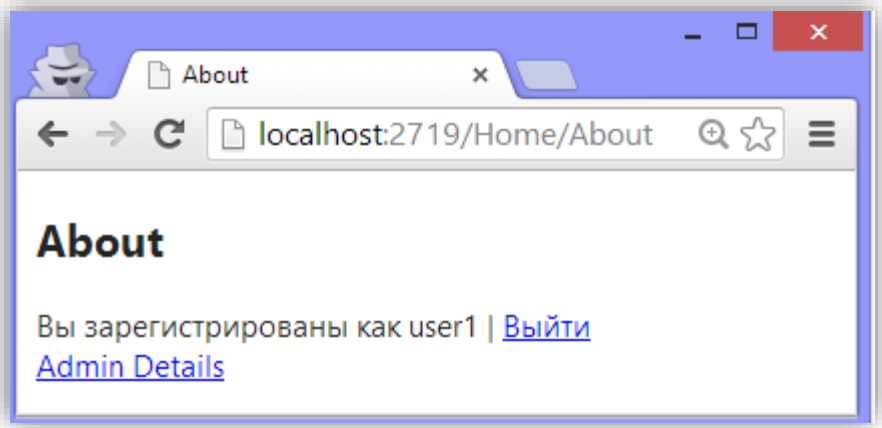
Метод **OnActionExecuting()** вызывается **перед** вызовом **action**.
А метод **OnActionExecuted()** - **после**.

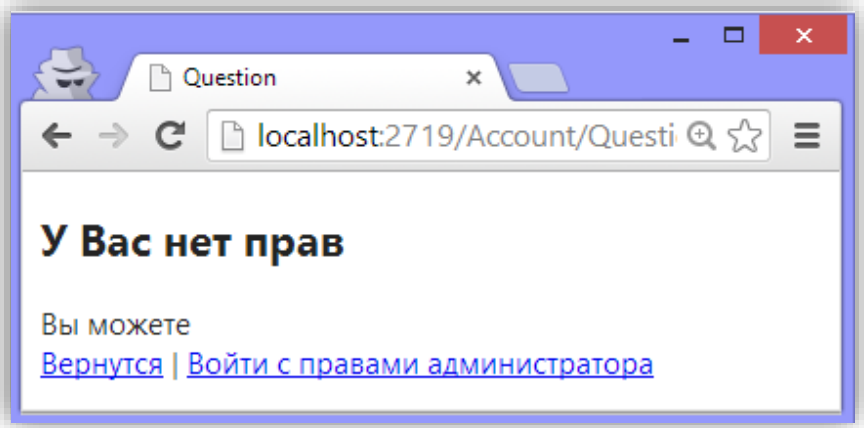
ACTION FILTERS

Например, если пользователь, аутентифицирован, но не имеет прав доступа к контроллеру, то фильтр `[Authorize(Roles = "admin")]` перенаправит пользователя на страницу с аутентификацией.

Это неудобно, т.к. никакой информации о такой причине перенаправления по умолчанию пользователю не сообщается.

Создадим фильтр действий, который бы отправлял пользователя на страницу с предложением войти как администратор или вернуться на исходную страницу:





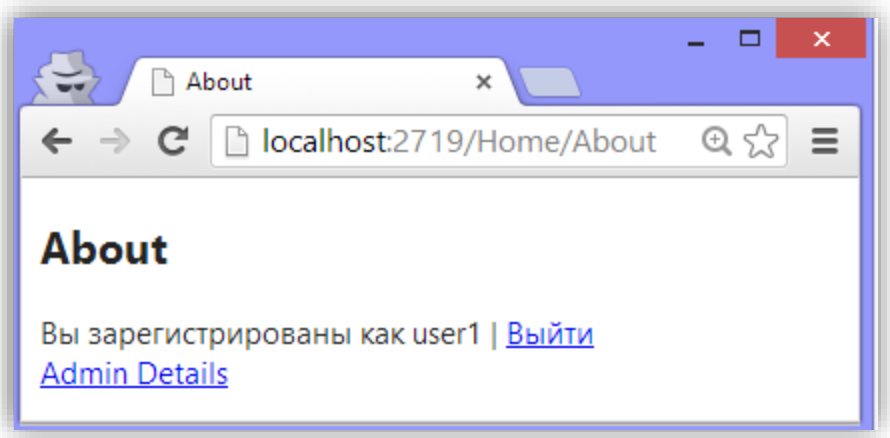
ACTION FILTERS

```
[Authorize] // вход только аутентифицированным пользователям
public ActionResult About()
{
     // запомнить и передавать дальше адрес откуда пришел пользователь
     ViewBag.Url = Request.Path;
     return View();
}

[Authorize]
[AdminAction] // Фильтр, позволяющий обращаться к action только администраторам
public ActionResult Details()
{
     ViewBag.Msg = "Эта страница должна быть доступна только администратору";
     return View();
}
```

ACTION FILTERS

```
<!- About.cshtml -->
<h2>About</h2>
@if (Request.IsAuthenticated)
{
     <span>Вы зарегистрированы как @User.Identity.Name</span><span>|</span>
     @Html.ActionLink("Выйти", "LogOff", "Account")
}
<div>
    @Html.ActionLink("Admin Details","Details",
          new { returnUrl = ViewBag.Url }, null)
</div>
```

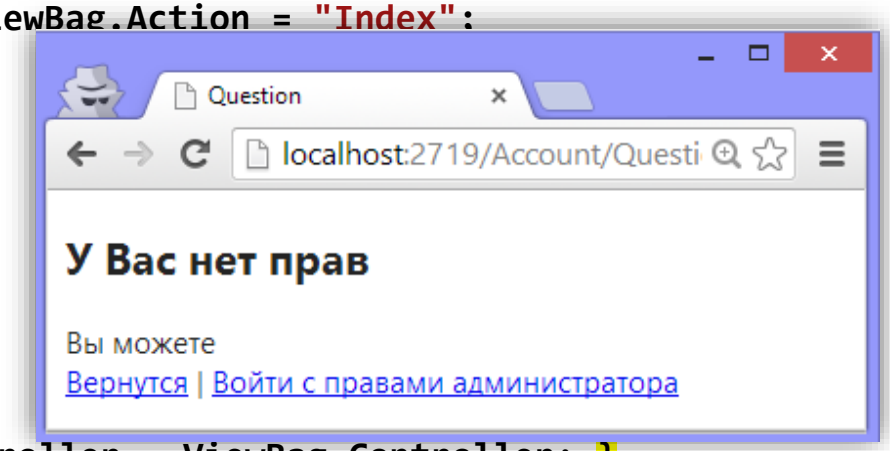


ACTION FILTERS

```
public class AdminActionAttribute : FilterAttribute, IActionFilter {
     public void OnActionExecuting(ActionExecutingContext filterContext) {
          if (filterContext.HttpContext.Request.IsAuthenticated)
          {
             var returnUrl = filterContext.HttpContext.Request["returnUrl"];
             if (!Roles.IsUserInRole("admin"))
             {
                 filterContext.Result =
                    new RedirectToRouteResult(new RouteValueDictionary {
                         { "action", "Question" },
                         { "controller", "Account" },
                         { "returnUrl", returnUrl }
                    });
               }
          }
          else { // перенаправить на страницу с авторизацией }
      }
}
```

ACTION FILTERS

```
public ActionResult Question(string returnUrl){
     if (returnUrl != null) {
          var segments = returnUrl.Split('/');
          ViewBag.Controller = segments[1]; ViewBag.Action = segments[2];
     }  else {
          ViewBag.Controller = "Home"; ViewBag.Action = "Index";
     }
     ViewBag.returnUrl = returnUrl;
     return View();
}
```




```
<h2>У Вас нет прав</h2>
<div> Вы можете </div>
@{ var action = ViewBag.Action; var controller = ViewBag.Controller; }
@Html.ActionLink("Вернутся", action, controller) |
@Html.ActionLink("Войти с правами администратора", "Login", "Account",
              new { returnUrl = ViewBag.returnUrl }, null)
```

RESULT FILTERS

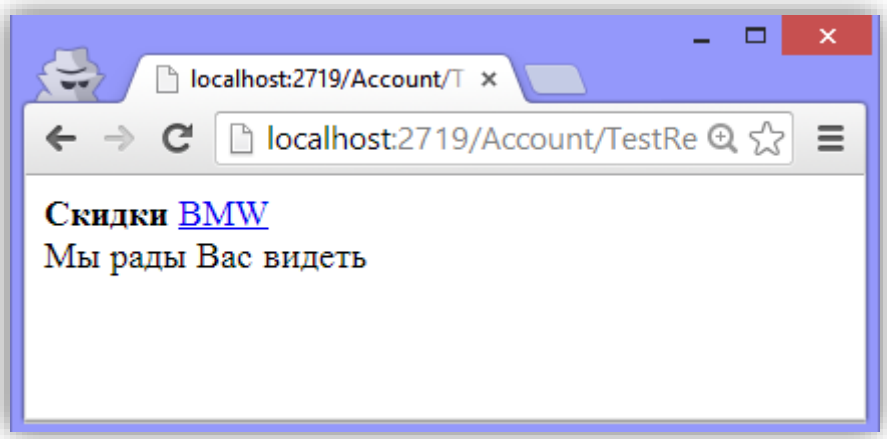
```
public interface IResultFilter
{
    void OnResultExecuting(ResultExecutingContext filterContext);
    void OnResultExecuted(ResultExecutedContext filterContext);
}
```

Предназначены для воздействия на результат, возвращаемый при помощи **action**.
Например, если возвращается объект определенного класса,
то в фильтре результатов можно его сериализовать для дальнейшей записи в поток.

Или **action** возвращает HTML-код, то в фильтре можно результат преобразовать,
например добавить нужные теги и др.

RESULT FILTERS

```
[DiscountFilter]
public ContentResult TestResult()
{
    var result = @"<a href='#'>BMW</a>";
    return Content(result);
}
```



RESULT FILTERS

```
public class MyResultAttribute : FilterAttribute, IResultFilter
{
    public void OnResultExecuting(ResultExecutingContext filterContext)
    {
        var result = filterContext.Result as ContentResult;
        result.Content = "<strong>Скидки</strong> " + result.Content;
        filterContext.Result = result as ActionResult;
    }

    public void OnResultExecuted(ResultExecutedContext filterContext)
    {
        filterContext.HttpContext.Response.Write("<h3>Мы рады Вас видеть</h3>");
    }
}
```

ADDITIONAL FILTERS

Фильтр [OutputCache] кеширует ответ на указанный период времени:

```
[OutputCache (Duration=10)]
public ActionResult Index()
{
    Response.Write(DateTime.Now.ToString());
    return View();
}
```

Фильтр [ValidateAntiforgeryToken]:
предназначен для противодействия подделке межсайтовых запросов, производя верификацию токенов при обращении к action.

Наиболее частым случаем является применение данного фильтра к методам, отвечающим за авторизацию

GLOBAL FILTERS

Предназначены для применения фильтра ко всем контроллерам.

Добавляется глобальный фильтр в файле ***FilterConfig.cs***, который находится в папке ***App_Start***:

```
public class FilterConfig
{
     public static void RegisterGlobalFilters(GlobalFilterCollection filters)
     {
          filters.Add(new HandleErrorAttribute());
          filters.Add(new MyIndexException()); // регистрация собственного
                                               // глобального фильтра
     }
}
```

ANY QUESTIONS?

THANKS FOR YOUR ATTENTION!