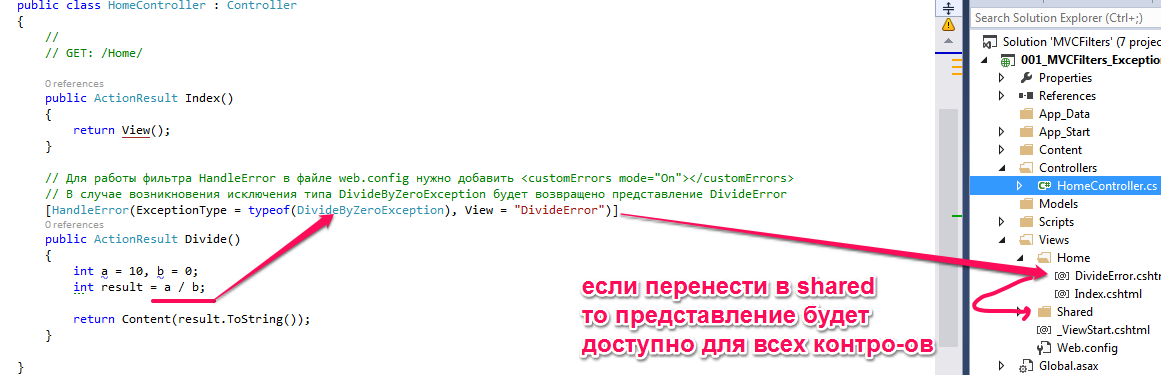


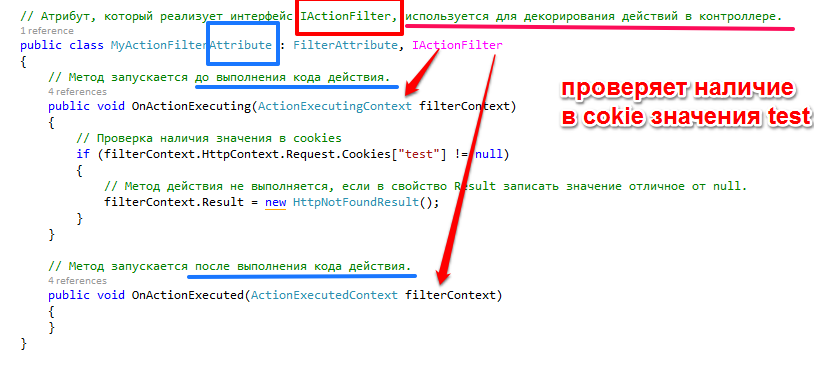
Фильтр ошибок

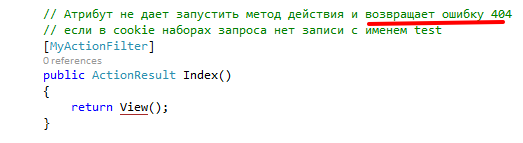
в файле web.config нужно добавить <customErrors mode="On"></customErrors>

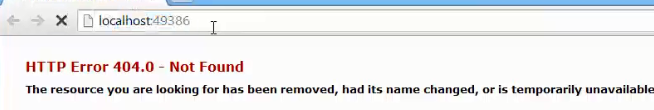




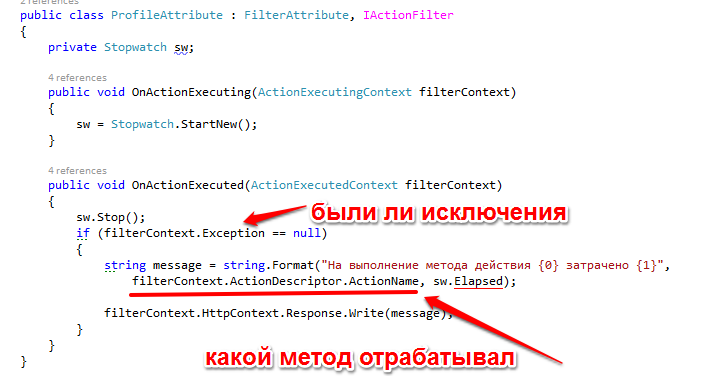
Пользовательский фильтр

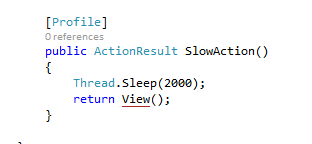


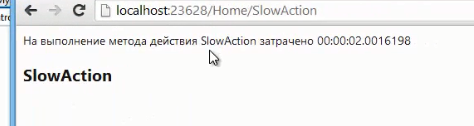




Фильтр считающий время выполнения действия

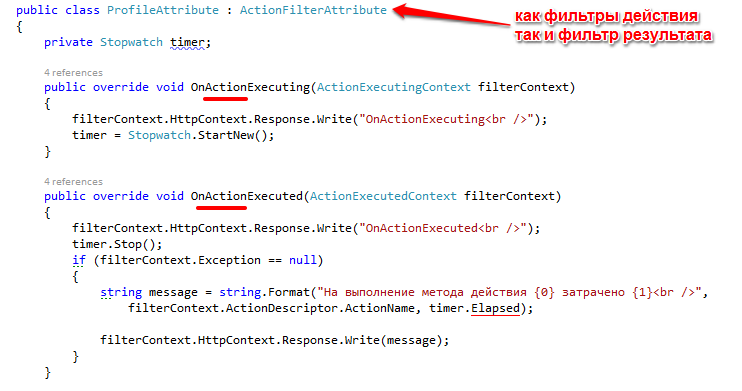




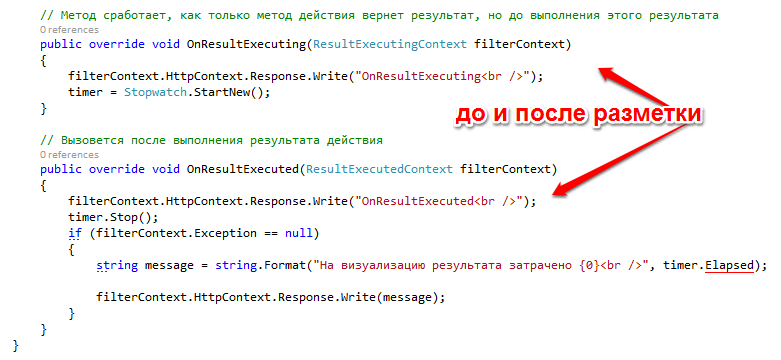


Фильтр действия и фильтр результата

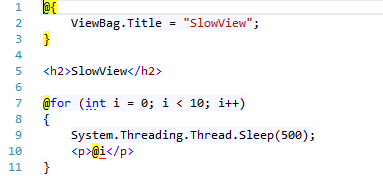
Замеряем время контролеров, методов действия



Замеряем скорость в представлении

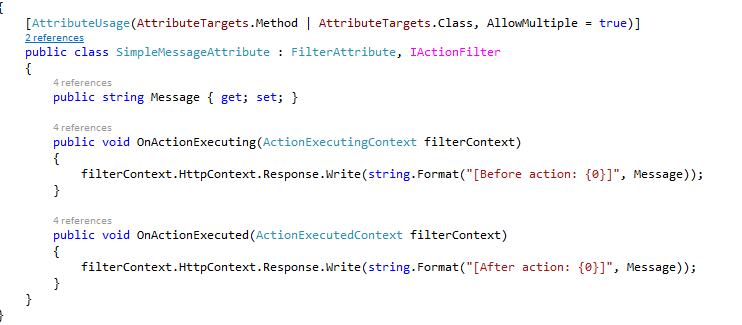


Одно из представлений





Фильтр действия

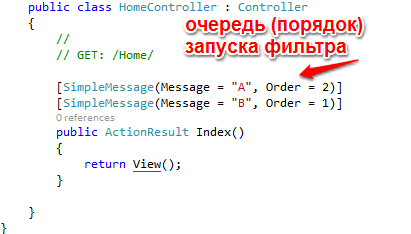


Свойства ActionExecutingContext

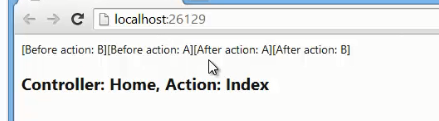
ActionDescriptor ActionDescriptor Предоставляет информацию о методе действия

Result ActionResult Результат для метода действия; фильтр может отменить запрос, установив данному свойству иное значение, кроме

null



Срабатывают в начале и конце в разном порядке



Можно в фильтре добавить какие либо значения в модель

var result = filterContext.Result as ViewResultBase;

var model = result.Model as MyViewModel;

if (model == null)

{

// there's no model or the model was not of the expected type

// => no need to continue any further

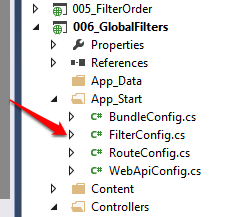
return;

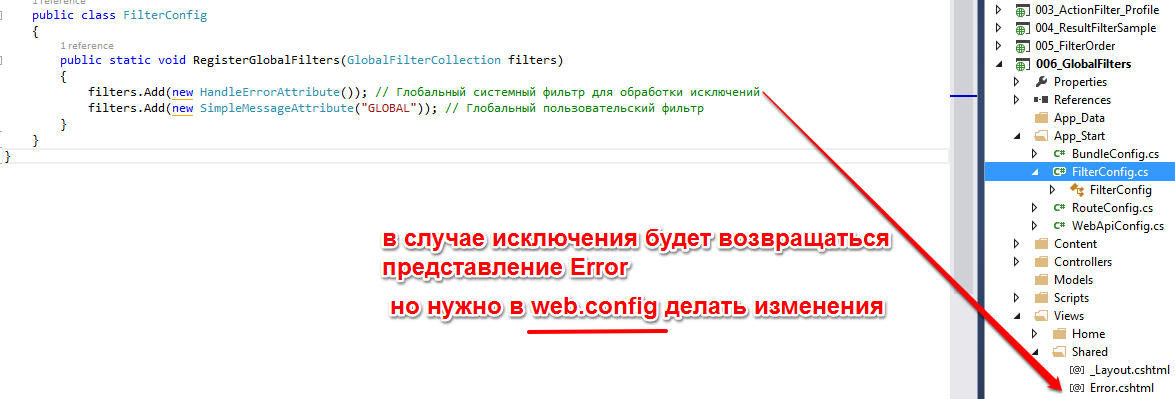
}

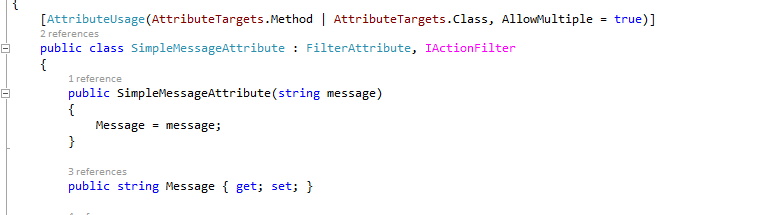
// modify some property value

model.Foo = "bar";

Глобальное использование фильтра



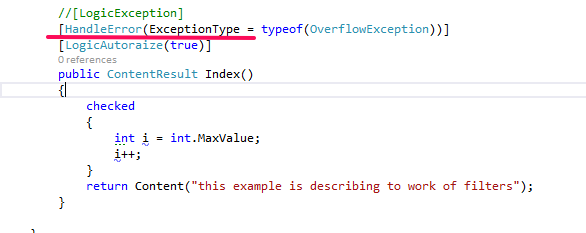




Обработка ошибки для метода

В представление прокидуется @model HandleErrorInfo

<customErrors mode="On" defaultRedirect="~/Views/Shared/Error.cshtml"/>



Например, можно проверять статус авторизированности запроса в каждом методе

Действия

public ViewResult Index()

{

if (!Request.IsAuthenticated)

{

FormsAuthentication.RedirectToLoginPage();

}

// ...rest of action method

}

public class HomeController : Controller

{

[Authorize(Users = "adam, steve, jacqui", Roles = "admin")]

public string Index()

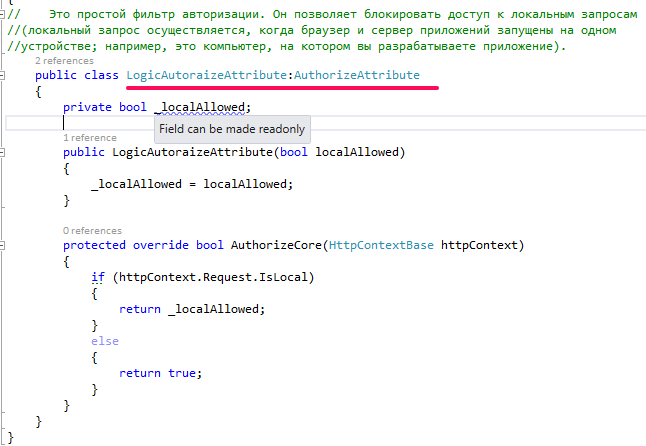
{

return "This is the Index action on the Home controller";

}

}

Своя авторизация



Когда появится необработанное исключение, будет вызван метод OnException. Параметром для этого метода является объект ExceptionContext, который наследует от ControllerContext и имеет ряд полезных свойств, с помощью которых можно получить информацию о запросе. Они приведены

Таблица 16-3: Свойства ControllerContext

Название Тип Описание

Controller ControllerBase Возвращает объект контроллера для данного запроса

HttpContext HttpContextBase Обеспечивает доступ к информации о запросе и доступ к ответу

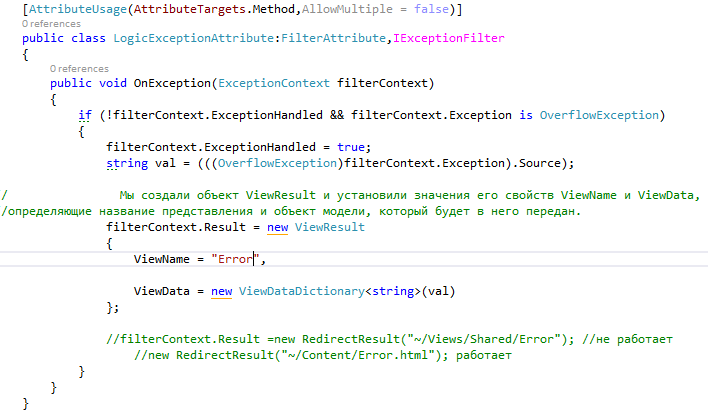
IsChildAction bool Возвращает true, если это дочернее действие (будет кратко

обсуждаться позже в этой главе и подробно в главе 18)

RequestContext RequestContext Предоставляет доступ к объекту HttpContext и данным

маршрутизации, хотя и то, и то доступно через другие свойства

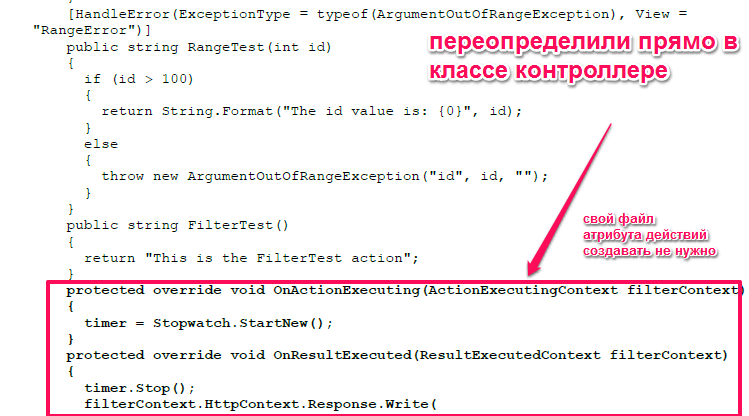
RouteData RouteData Возвращает данные маршрутизации для данного запроса



неудивительно, потому что new RedirectResult("/Views/Home/NotFound.cshtml"); работать не будет. Редирект не будет идти на представления из папки Views. Редирект мы можем сделать на определенные пути - действия контроллера или статичные html страницы не из Views. Либо надо объявить метод в контроллере, который бы обрабатывал исключение   
public ActionResult Error()  
{  
return View("ExceptionFound");  
}  
а фильтр бы делал редирект на этот метод :  
exceptionContext.Result = new RedirectResult("Home/Error");

Либо вместо редиректа в фильтре использовать напрямую генерацию представления:  
exceptionContext.Result = new ViewResult { ViewName = "ExceptionFound" };

Этот подход будет полезен, когда вы создаете базовый класс, от которого наследуют несколько контроллеров вашего проекта. Весь смысл фильтрации заключается в том, чтобы заключить код, необходимый всему приложению, в один модуль, подразумевающий многоразовое использование.



filters.Add(new ProfileAllAttribute());

Обратите внимание, что к фильтру нужно обращаться, используя полное имя класса

(ProfileAllAttribute), а не краткое, которое применяется при обращении к фильтру как к атрибуту

(ProfileAll). Фильтр, зарегистрированный таким образом, будет применен к каждому действию

метода.

Используем фильтр RequireHttps

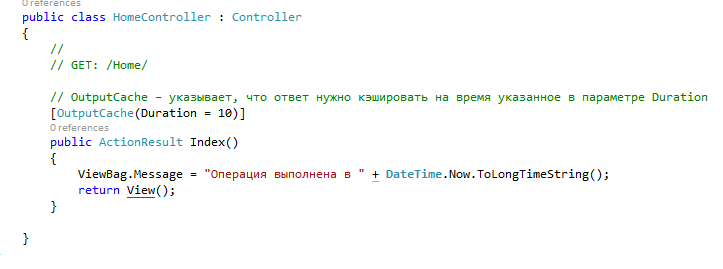
Фильтр RequireHttps разрешает использование протокола HTTPS для действий. Он перенаправляет браузер пользователя к тому же самому действию, но используя префикс протокола https://.

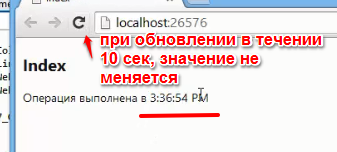
Кеширование

Результат ответа сервера должен кэшироваться на сервере 10 секунд

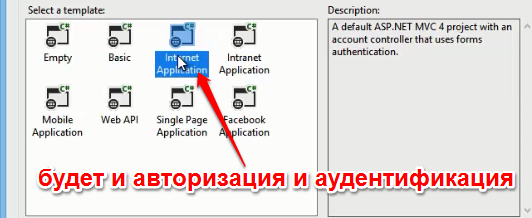
Благодаря этому разметка кэшируется в операционной системе HTTP CIS

Нужно для того чтобы не запускать данный метод при повторном обращении пользователю, а вернется ему эта готовая разметка

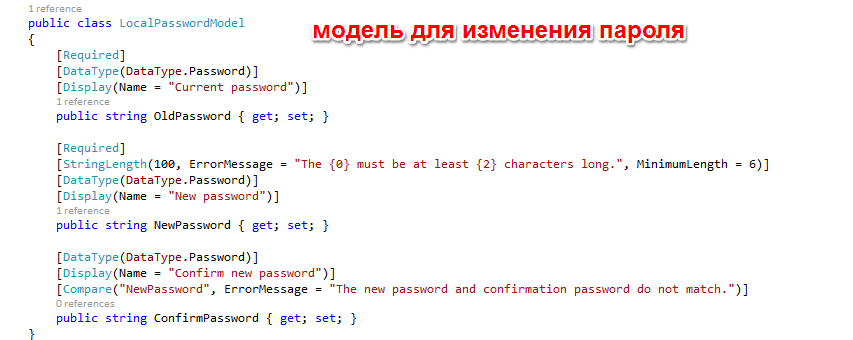


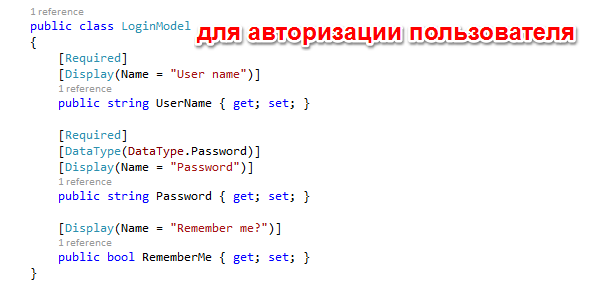


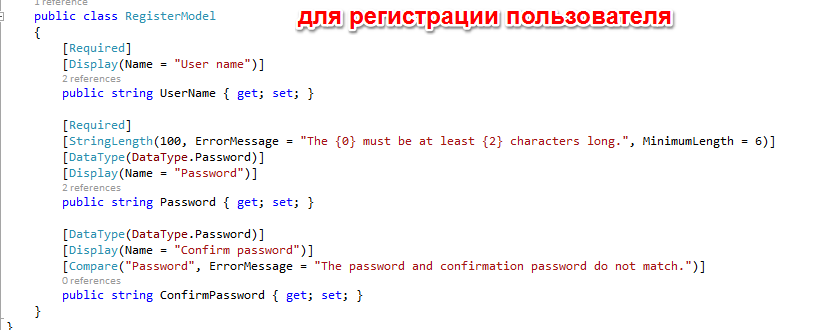




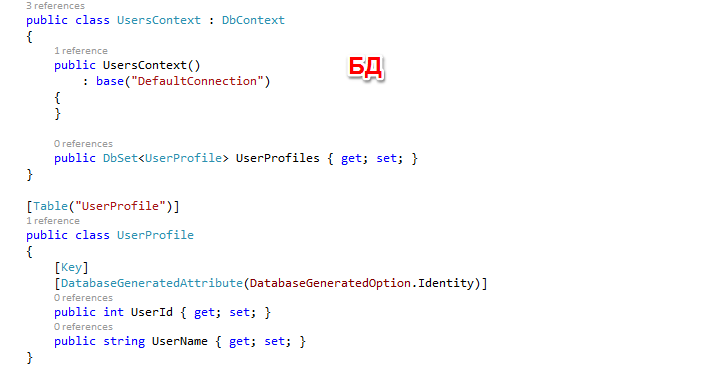
Модели



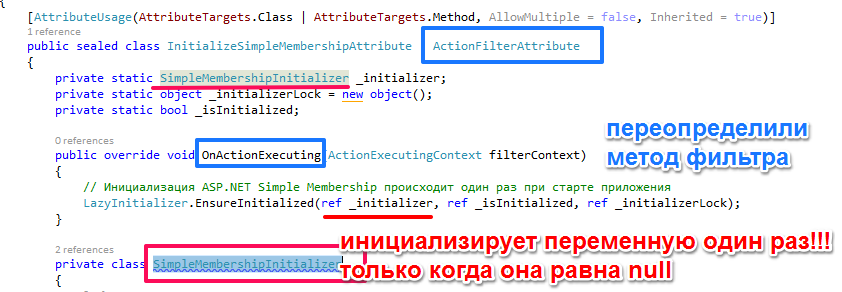


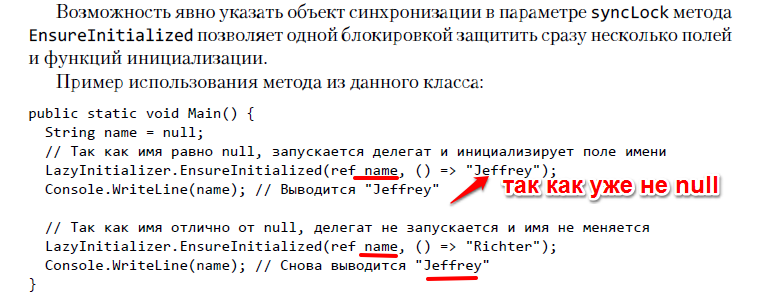


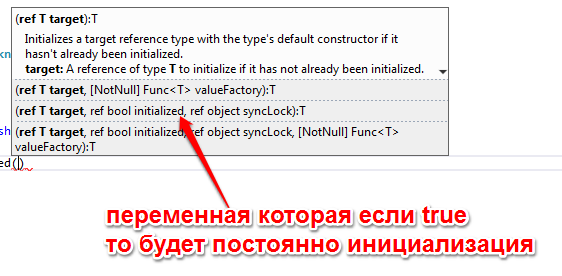
База данных

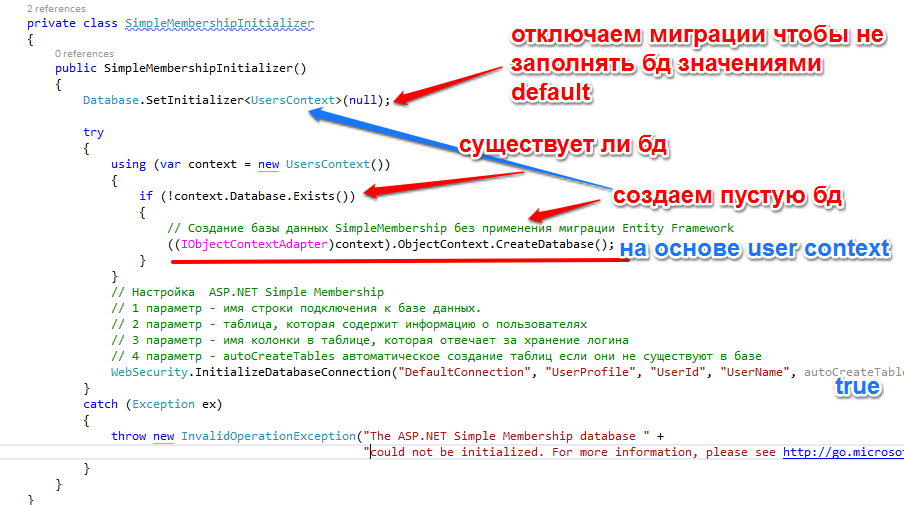


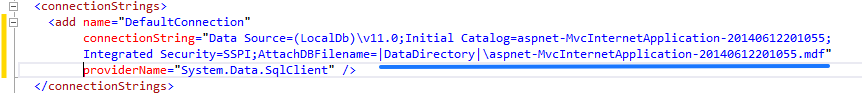
Фильтр действия





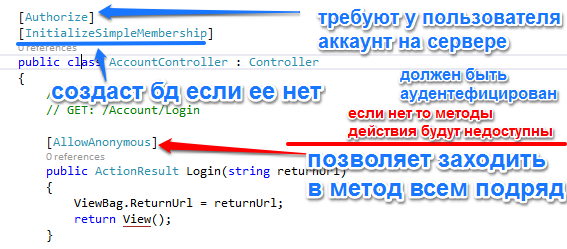




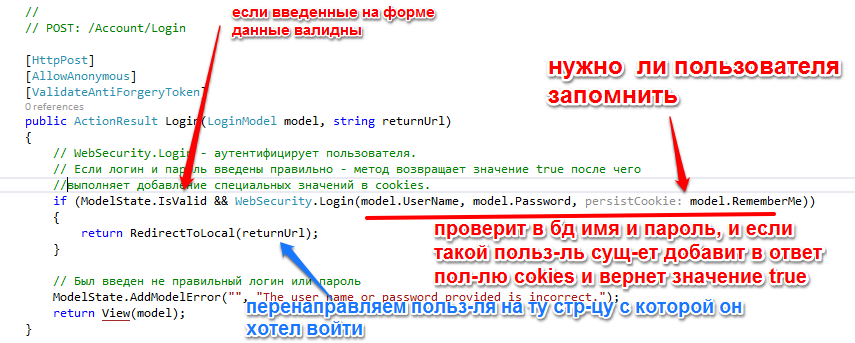


Контролеры

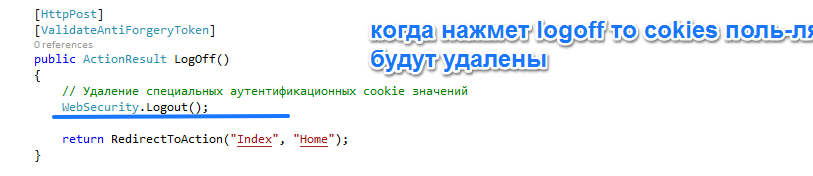




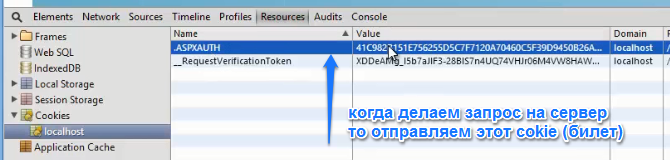
Зайдя в логин и заполнив форму мы отправим данные методу



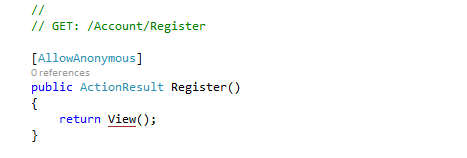
Выход из акаунта

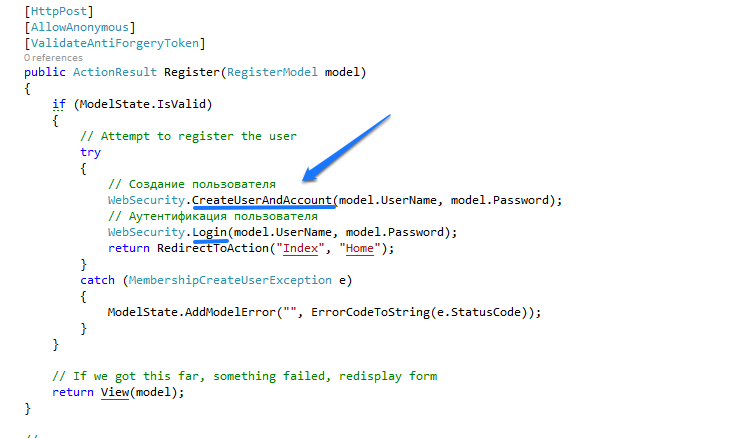


Если авторизовались

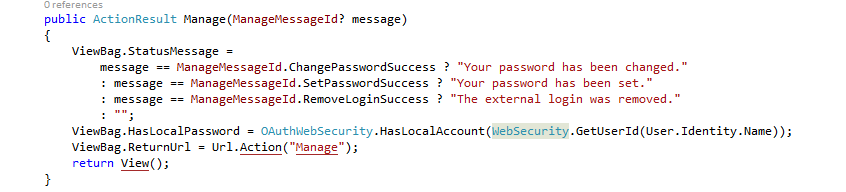


Регистрация





Смена пароля пользователя

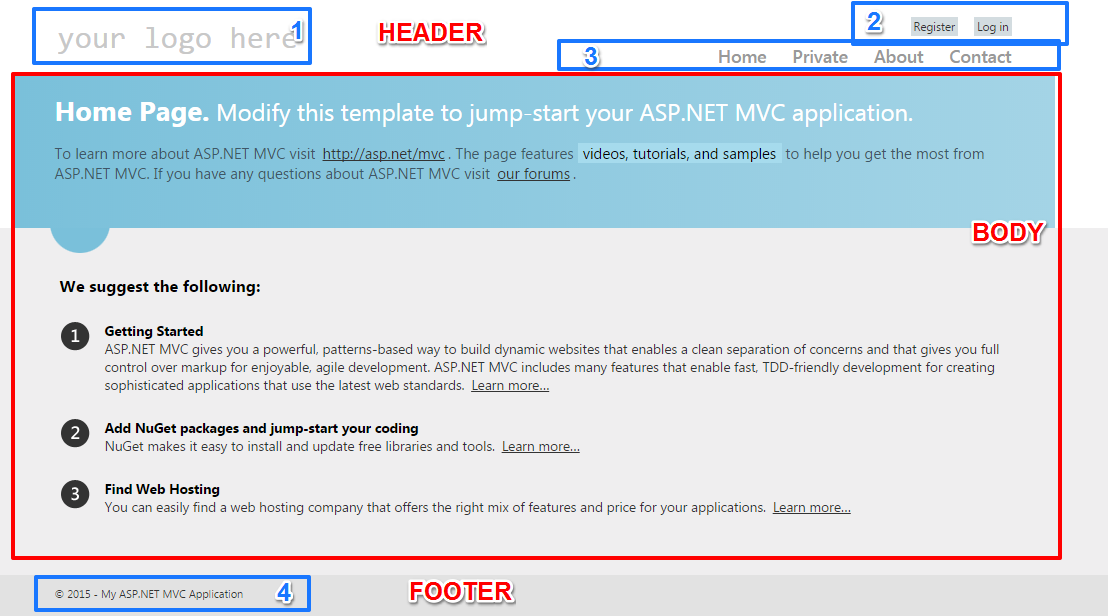


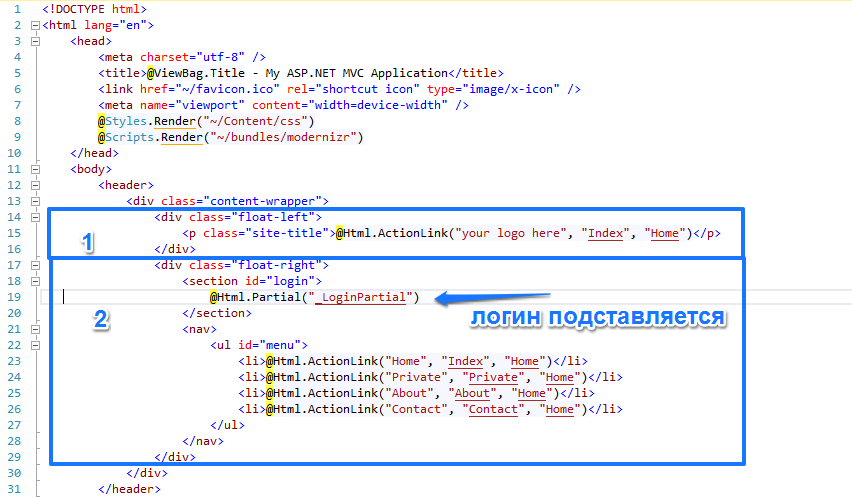
Когда возвращает форму

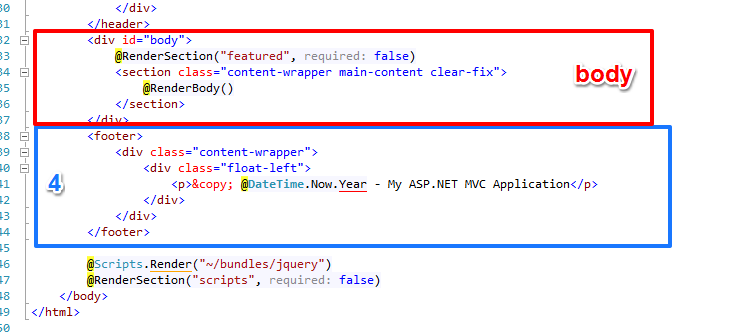


Подробный разбор

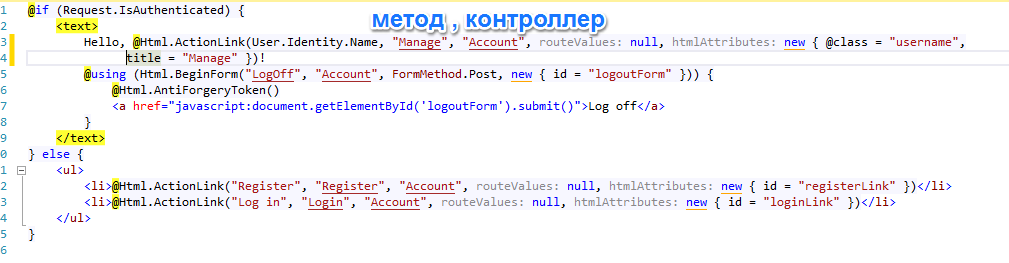
Layout



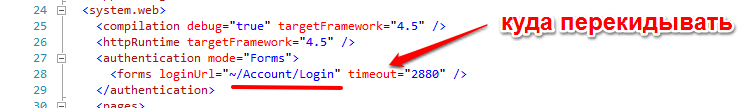




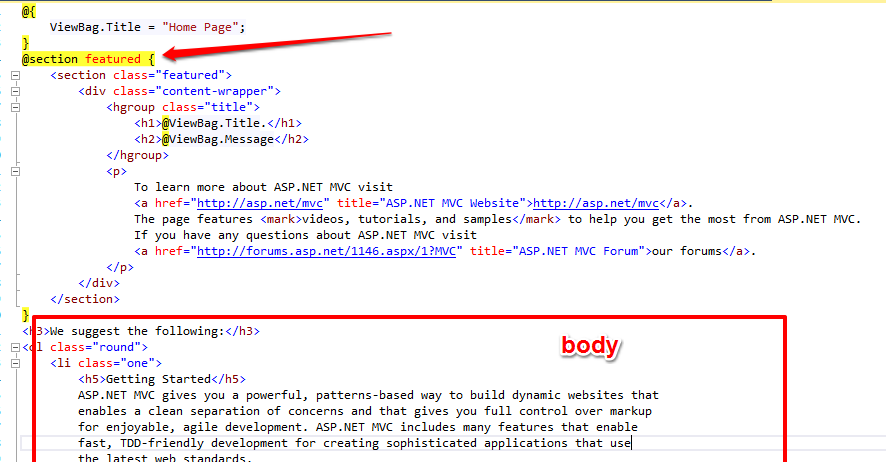
\_LoginPartial



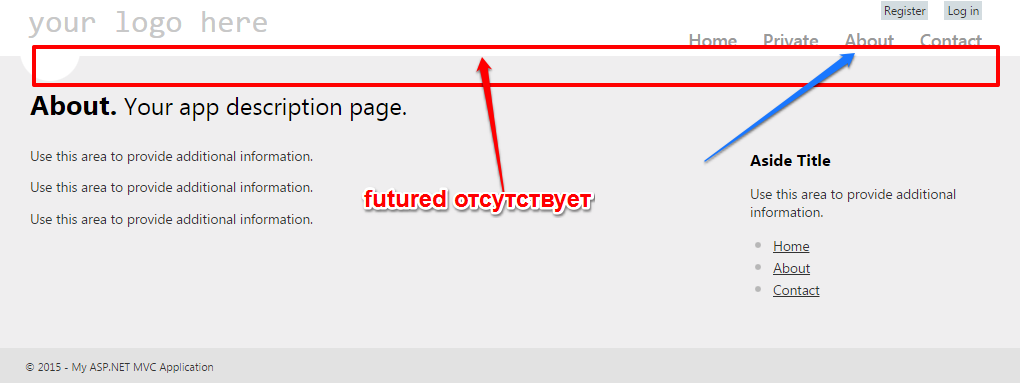
При запрещении доступа куда перекидывать

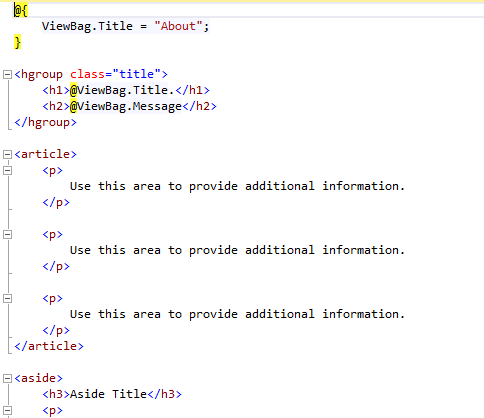


Index

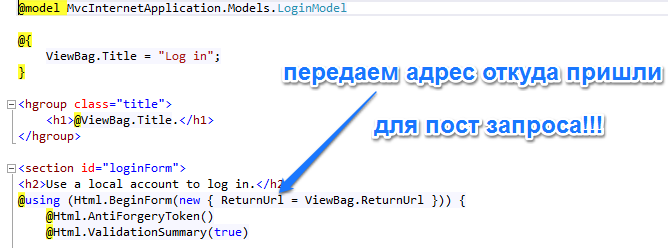


About

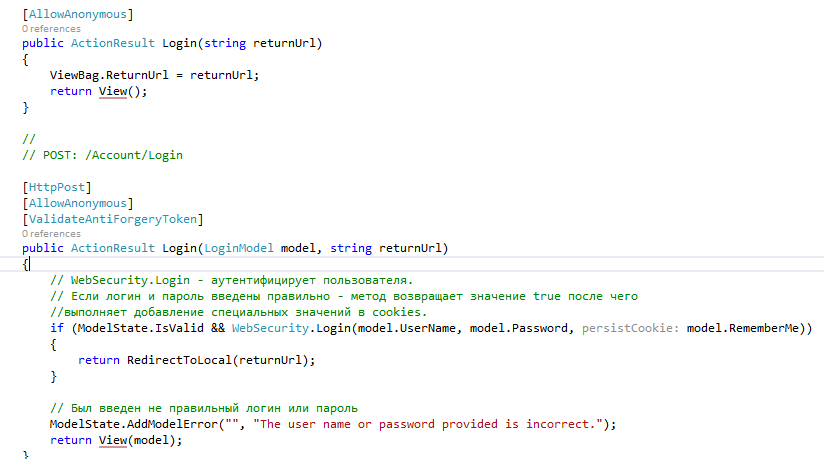


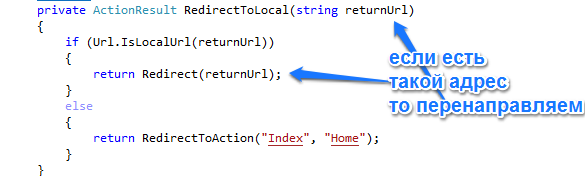


Login



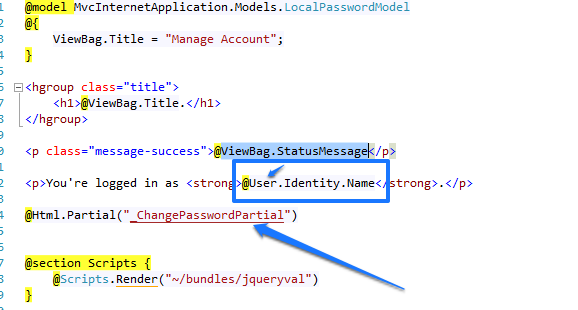






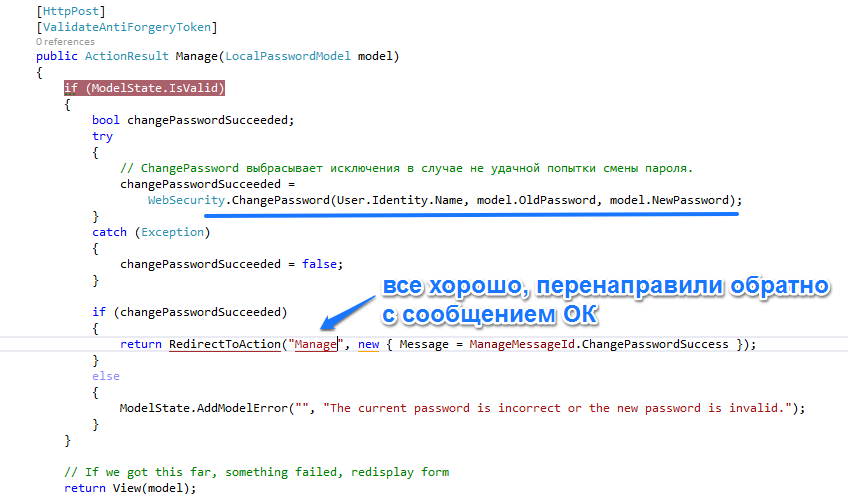
Manage

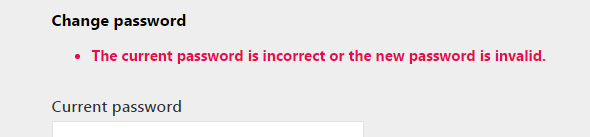


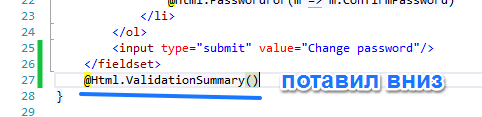


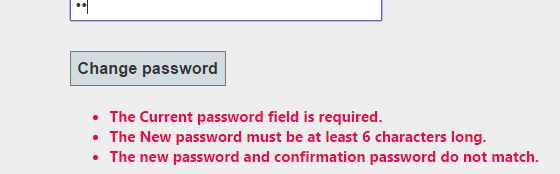
\_ChangePasswordPartial



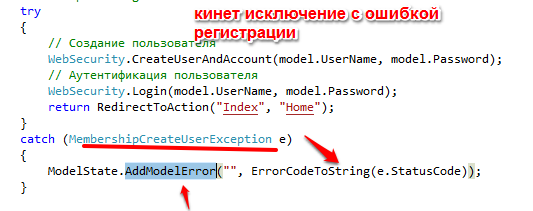








Ригистрация





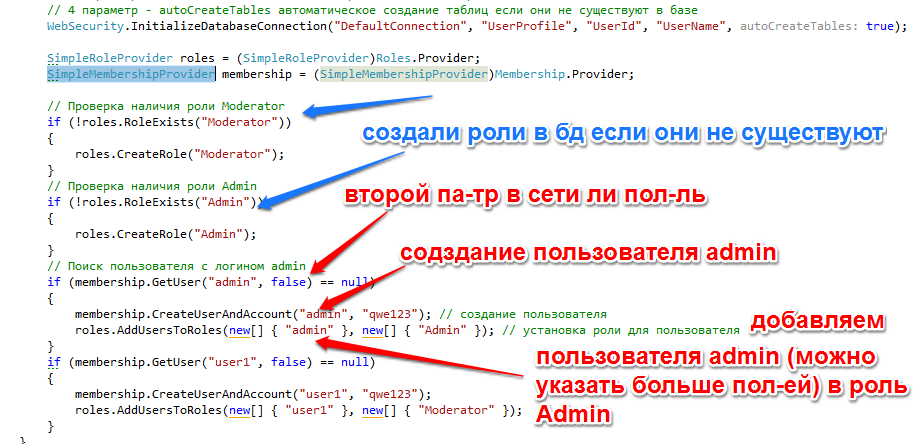
Авторизация

SimpleRoleProvider

SimpleMembershipProvider – можно добавлят, редактировать, удалять существующих пользователей

Добавление в класс атрибута

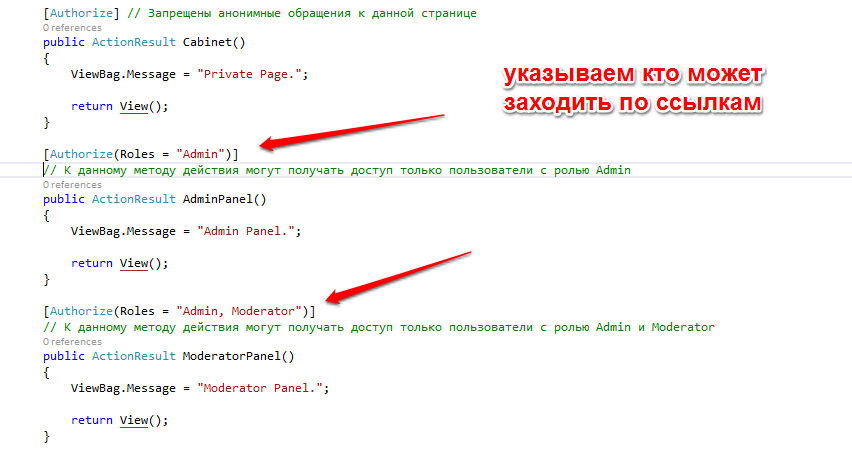
InitializeSimpleMembershipAttribute



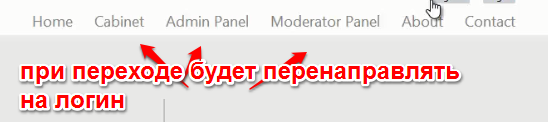
В layout



HomeController



При недостатке прав будет перенаправлять на логин



48 44