Аутентификация OWIN и ClaimsIdentity

Последнее обновление: 16.11.2015

В этой статье рассмотрим, как мы можем аутентифицировать пользователя и разграничить доступ по ролям без всяких провайдеров и ASP.NET Identity средствами инфраструктуры OWIN. И для этого вначале создадим новый проект без аутентификации.

Прежде всего, нам необходимы модели пользователя и роли, а также контекст данных. Поэтому добавим в папку Models следующие классы:

```
public class User
{
    public int Id { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public Role Role { get; set; }
    public int RoleId { get; set; }
}
public class Role
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

И также добавим класс контекста данных и инициализатор:

```
public class UserContext : DbContext
{
    public UserContext():base("DefaultConnection")
    { }
    public DbSet<User> Users { get; set; }
    public DbSet<Role> Roles { get; set; }
}

public class UserDbInitializer : DropCreateDatabaseAlways<UserContext>
{
    protected override void Seed(UserContext db)
    {
        db.Roles.Add(new Role { Id = 1, Name = "admin" });
        db.Roles.Add(new Role { Id = 2, Name = "user" });
        db.Users.Add(new User
        {
            Email = "alice@gmail.com",
            Password = "123456",
            RoleId = 1
        });
        db.Users.Add(new User
        {
            Email = "tom@gmail.com",
            Password = "123456",
            RoleId = 2
        });
        base.Seed(db);
    }
}
```

Пусть в файле web.config у нас будет определена строка подключения DefaultConnection, а в файле Global.asax вызывается вышеопределенный инициализотор БД.

Теперь нам надо подключить OWIN в проект. Для этого добавим через Nuget следующие пакеты:

```
Microsoft.Owin.Host.SystemWeb
Microsoft.Owin.Security.Cookies
```

Они имеют ряд зависимостей, и вместе с ними будут добавлены пакеты OWIN, Microsoft.Owin и Microsoft.Owin.Security.

Затем добавим в проект в папку App_Start класс OWIN, который назовем Startup:

```
using Microsoft.Owin;
using Owin;
using Microsoft.Owin.Security.Cookies;

[assembly: OwinStartup(typeof(IdentityUoWApp.Startup))]
namespace IdentityUoWApp
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = "ApplicationCookie",
                LoginPath = new PathString("/Account/Login"),
            });
        }
    }
}
```

С помощью метода расширения `UseCookieAuthentication()` здесь устанавливается, что при аутентификации пользователей будут использоваться куки: `AuthenticationType = "ApplicationCookie"`.

Теперь сделаем функционал для входа на сайт. Вначале добавим в проект папку ViewModels и в ней определим класс LoginViewModel - модель входа на сайт:

```
using System.ComponentModel.DataAnnotations;

public class LoginModel
{
    [Required]
    public string Email { get; set; }
    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }
}
```

Затем добавим в папку Controllers новый контроллер AccountController:

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;
using System.Security.Claims;
using Microsoft.Owin.Security;
using IdentityUoWApp.ViewModels; // пространство имен LoginViewModel
using IdentityUoWApp.Models; // пространство имен моделей
using System.Threading.Tasks;

namespace IdentityUoWApp.Controllers
{
    public class AccountController : Controller
    {
        UserContext db = new UserContext();
        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        public ActionResult Login()
        {
```

```
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(LoginModel model)
        {
            if (ModelState.IsValid)
            {
                User user = await db.Users.Include(u=>u.Role).FirstOrDefaultAsync(u => u.Email == model.Email &&
u.Password == model.Password);

                if (user == null)
                {
                    ModelState.AddModelError("", "Неверный логин или пароль.");
                }
                else
                {
                    ClaimsIdentity claim = new ClaimsIdentity("ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
                    claim.AddClaim(new Claim(ClaimTypes.NameIdentifier, user.Id.ToString(),
ClaimValueTypes.String));
                    claim.AddClaim(new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email,
ClaimValueTypes.String));
                    claim.AddClaim(new
Claim("http://schemas.microsoft.com/accesscontrolservice/2010/07/claims/identityprovider",
                        "OWIN Provider", ClaimValueTypes.String));
                    if (user.Role!=null)
                        claim.AddClaim(new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.Name,
ClaimValueTypes.String));

                    AuthenticationManager.SignOut();
                    AuthenticationManager.SignIn(new AuthenticationProperties
                    {
                        IsPersistent = true
                    }, claim);
                    return RedirectToAction("Index", "Home");
                }
            }
            return View(model);
        }

        public ActionResult Logout()
        {
            AuthenticationManager.SignOut();
            return RedirectToAction("Index", "Home");
        }
    }
}
```

Первым делом с помощью контекста OWIN мы получаем объект **IAuthenticationManager**. Что это за объект? Исходный код класса AuthenticationManager можно найти здесь. Данный объект управляет аутентификационными куками и выполняет ряд других действий.

В POST-методе Login мы получаем из представления объект LoginModel и по нему определяем пользователя. Если пользователь был найден, то создается объект **ClaimsIdentity**:

```
ClaimsIdentity claim = new ClaimsIdentity("ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
ClaimsIdentity.DefaultRoleClaimType);
```

Первый параметр - "ApplicationCookie" указывате на тип аутентификации, то есть куки. И в конструктор также передаются константы ClaimsIdentity.DefaultNameClaimType и ClaimsIdentity.DefaultRoleClaimType, которые представляют название клейма логина пользователя и название клейма роли.

Далее в объект ClaimsIdentity последовательно добавляются клеймы, которые содержат логин пользователя, его id, роль:

```
claim.AddClaim(new Claim(ClaimTypes.NameIdentifier, user.Id.ToString(), ClaimValueTypes.String));
claim.AddClaim(new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email, ClaimValueTypes.String));
if (user.Role!=null)
    claim.AddClaim(new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.Name, ClaimValueTypes.String));
```

И также нам надо добавить специальный клейм провайдера:

```
claim.AddClaim(new Claim("http://schemas.microsoft.com/accesscontrolservice/2010/07/claims/identityprovider",
    "OWIN Provider", ClaimValueTypes.String));
```

Причем второй параметр представляет в данном случае произвольное название провайдера - "OWIN Provider".

По похожей схеме происходит создание аутентификационного тикета в ASP.NET Identity, правда, там провайдер четко определен - "Asp Net Identity".

Далее с помощью методов `AuthenticationManager.SignOut()` и `AuthenticationManager.SignIn()` куки удаляются и вновь устанавливаются.

Таким образом, аутентификационный тикет будет содержать данные о логине пользователя, его роли и id. Мы уже по умолчанию можем получать логин пользователя: `User.Identity.Name`. Но на данный момент в проекте нет никакого способа определять id пользователя и роль. И для этого добавим в папку Models новый класс IdentityExtensions:

```
using System;
using System.Globalization;
using System.Security.Claims;
using System.Security.Principal;

namespace IdentityUoWApp.Models
{
    public static class IdentityExtensions
    {
        public static T GetUserId<T>(this IIdentity identity) where T : IConvertible
        {
            if (identity == null)
            {
                throw new ArgumentNullException("identity");
            }
            var ci = identity as ClaimsIdentity;
            if (ci != null)
            {
                var id = ci.FindFirst(ClaimTypes.NameIdentifier);
                if (id != null)
                {
                    return (T)Convert.ChangeType(id.Value, typeof(T), CultureInfo.InvariantCulture);
                }
            }
            return default(T);
        }
        public static string GetUserRole(this IIdentity identity)
        {
            if (identity == null)
            {
                throw new ArgumentNullException("identity");
            }
            var ci = identity as ClaimsIdentity;
            string role = "";
            if (ci != null)
            {
                var id = ci.FindFirst(ClaimsIdentity.DefaultRoleClaimType);
                if (id != null)
                    role = id.Value;
            }
            return role;
        }
    }
}
```

Данный класс создает два метода расширения GetUserId и GetUserRole для получения из аутентификационного тикета id и роли пользователя.

И мы можем, например, использовать разграничение входа и эти методы в имеющемся у нас HomeController:

```
using System.Web.Mvc;
using IdentityUoWApp.Models;

namespace IdentityUoWApp.Controllers
{
    [Authorize]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        [Authorize(Roles ="admin")]
        public ActionResult About()
        {
            ViewBag.Message = User.Identity.GetUserRole();
```

```
            return View();
        }
        public ActionResult Contact()
        {
            int id = User.Identity.GetUserId<int>();
            ViewBag.Message = "Ваш id: " + id.ToString();

            return View();
        }
    }
}
```

Назад **Содержание** **Вперед**

AI • месяц назад
Интеллисенс мне подсказывает, что в конструкторе класса Claim третий параметр по умолчанию имеет значение ClaimValueTypes.String, т.е. его можно было не указывать. Или я ошибаюсь?
Ответить • Поделиться ›

Евгений • 2 месяца назад
Здравствуйте.

При данном способе авторизации не работает атрибут [AllowAnonymous], если контроллер имеет атрибут [Authorize], а метод [AllowAnonymous], то при обращении к этому методу неавторизированного пользователя перенаправляет на страницу входа. Подскажите, пожалуйста, как это исправить.
Ответить • Поделиться ›

Аркадий • 4 месяца назад
Здравствуйте!
Спасибо за статью!

Подскажите, пожалуйста, как предотвратить вход пользователя, если он был удалён после того, как залогинился и получил заветные куки с IsPersistent = true?
В Identity это реализовано через SessionStore

^ | v • Ответить • Поделиться ›

Макс Б • 6 месяцев назад

Здравствуйте. А как , используя такой способ аутентификации, использовать Security Stamp?

^ | v • Ответить • Поделиться ›

Айрат • 9 месяцев назад

Добрый вечер! Я довольно новичок в этом. Все выполнил как написано в статье. Единственный момент: "Пусть в файле web.config у нас будет определена строка подключения DefaultConnection, а в файле Global.asax вызывается вышеопределенный инициализотор БД." - я так понял web.config сгенерированный при создании проекта трогать не нужно (не уверен!?), а в Global.asax нужно добавить строчку кода. Можно ли вносимые изменения продемонстрировать здесь в комментариях либо внести небольшие правки в статью?

^ | v • Ответить • Поделиться ›

Metanit Модератор ➚ Айрат • 9 месяцев назад

в web.confige определяете строку подключения. подробнее - http://metanit.com/sharp/mv...
про использование инициализатора - http://metanit.com/sharp/mv...
Я бы вообще посоветовал бы вам, посмотреть хотя бы первые пять глав, если новичок

^ | v • Ответить • Поделиться ›

Edward • 9 месяцев назад

А как атрибут находит все имеющиеся роли? Откуда он знает в какой таблице искать, все же в данном решении таблицы "наши", а не стандартные asp.net Identity.

^ | v • Ответить • Поделиться ›

Metanit Модератор ➚ Edward • 9 месяцев назад

потому что при логине мы получаем данные из бд и сохраняем его роль в claim, который шифруется в куках. Через куки система все определяет

^ | v • Ответить • Поделиться ›

Denis • год назад

Возможна ли такая же система аутентификация и авторизации в Asp.net MVC core?

^ | v • Ответить • Поделиться ›

Alex • год назад

Есть ли возможность на основе этот системе прикрутить owin facebook & google auth?
Возможно у вас есть какие то статейки по этому поводу.Спасибо

^ | v • Ответить • Поделиться ›

Евгений ➚ Alex • год назад

И для меня актуально!

^ | v • Ответить • Поделиться ›

Sulaiman Jumukov • год назад

В папке App_Start есть класс Startup и там есть оператор - LoginPath = new PathString("/Account/Login") (как и у Вас). Но при запуске отображается страница "Index". Что сделать, чтобы при запуске приложения запускалась страница Login?
Заранее благодарен.

^ | v • Ответить • Поделиться ›

Metanit Модератор ➚ Sulaiman Jumukov • год назад

в файле RouteConfig поправьте дефолтный маршрут

^ | v • Ответить • Поделиться ›

Timur • 2 года назад

А не подскажите, как при данном способе авторизации сделать так, чтобы при закрытии браузера авторизация удалялась. Способ с выставлением времени авторизации не подходит, нужно именно как при сессиях - закрыл браузер - кука сессии удалилась.

• Ответить • Поделиться ›

Metanit Модератор ➚ Timur • 2 года назад

в методе
AuthenticationManager.SignIn(new AuthenticationProperties
{
IsPersistent = true
}, claim);
параметр IsPersistent

• Ответить • Поделиться ›

Юрий ➚ Metanit • год назад

параметр IsPersistent нужно изменить на false???

• Ответить • Поделиться ›

Timur ➚ Metanit • 2 года назад

Большое спасибо!

• Ответить • Поделиться ›

Ievgenii Stepanenko • 2 года назад

Здравствуйте. Подскажите какие настройки нужно сделать в web.config перед деплойментом. Написал приложение с вышеприведенной аутентификацией. Локально все работает отлично. После заброски на azure система для входа просто пишет Error. Невозможно залогинится или зарегистрировать нового пользователя

• Ответить • Поделиться ›

Ievgenii Stepanenko ➚ Ievgenii Stepanenko • 2 года назад

Решил проблему добавлением в Web.config

<system.web>
<authentication mode="None"/>
</system.web>
<system.webserver>
<modules>
<remove name="FormsAuthentication"/>
</modules>
</system.webserver>

• Ответить • Поделиться ›

Тарас ➚ Ievgenii Stepanenko • 2 года назад

Ставишь логгер и дебажишь. Узнаешь где ошибка. Гуглишь. Профит!

• Ответить • Поделиться ›

Виктор Алексеев • 2 года назад

Всем приветик!

Думаю, что в целях большей совместимости с примером из предыдущей главы, было бы уместно добавить в класс **LoginModel** переменную **RememberMe**.

В таком случае определение класса **LoginModel** выглядело бы вот так:

```
public class LoginModel
{
[Required]
public string Email { get; set; }

[Required]
[DataType(DataType.Password)]
public string Password { get; set; }

[Required]
```

```
public bool RememberMe { get; set; }
}
```

показать больше

• Ответить • Поделиться ›

Metanit Модератор ➚ Виктор Алексеев • 2 года назад

вот как раз связи с предыдущим проектом быть не должно. Собственно на этом и был акцент. Так как в серии прошлых тем использовалась Identity, а тут, наоборот, как пример, как все упростить и не использовать Identity. Но естественно вы можете по своему сделать, главное, чтобы вы понимали, что к чему

• Ответить • Поделиться ›

Виктор Алексеев ➚ Metanit • 2 года назад

Спасибо за дельный ответ и достаточно подробное изложение (Я бы даже сказал - разжевывание) материала.
По моим ощущениям - вся эта работа - тянет на отличный такой учебник.

• Ответить • Поделиться ›

Андрей • 2 года назад

Для меня осталось неясным, как обрабатываются запросы - через конвейер owin или "по-старинке"? В ASP.NET MVC 5 приложении есть и регистрация webapi не через owin и аутентификация owin. Чую, что знаний где моих -то не хватает... Получается некое смешение, никак не пойму как оно работает.

• Ответить • Поделиться ›

Metanit Модератор ➚ Андрей • 2 года назад

Роль OWIN здесь не такая и большая - просто через контекст OWIN устанавливается менеджер аутентификации, который сохраняет аутентификационные куки в браузере. В web api, правда, несколько сложнее - там еще инфраструктура для авторизации через токены используется

• Ответить • Поделиться ›

Андрей ➚ Metanit • 2 года назад

Нашел интересный цикл статей + Скачал исходники и разобрался. Спасибо.

• Ответить • Поделиться ›

Андрей ➚ Metanit • 2 года назад

Вот я собственно и не понимаю механизм обработки. Если OWIN это полноценный конвейер обработки запросов, то кто кого инициализирует. OWIN содержит в конвейере модуль WebAPI (тогда где он добавляется?) или же OWIN в обычном MVC5 приложении "сбоку", тогда не ясно в какой момент задействуется его конвейер и как он взаимодействует с сервером.

• Ответить • Поделиться ›

develop • 2 года назад

User user = await db.Users.Include(u=>u.Role).FirstOrDefaultAsync(u =< u.Email == model.Email && u.Password == model.Password); В этом коду ошибка идет. А именно await db.Users.Include(u=>u.Role) и (u =< u.Email. Заранее спасибо. Может я ошибаюсь.

• Ответить • Поделиться ›

Metanit Модератор ➚ develop • 2 года назад

какая именно ошибка?

• Ответить • Поделиться ›

develop ➚ Metanit • 2 года назад

Error 1 Cannot convert lambda expression to type 'string' because it is not a delegate type

• Ответить • Поделиться ›

Metanit Модератор ➚ develop • 2 года назад

using System.Data.Entity;

Ответить • Поделиться ›

develop ➚ Metanit • 2 года назад
Спасибо большое. Самый лучший сайт по ASP.NET MVC5. Удачи. Здоровья.
Ответить • Поделиться ›

Timur • 2 года назад
А как быть, если у пользователя несколько ролей? Имеется ввиду как их добавить в куки, а потом проверять?
Ответить • Поделиться ›

Metanit Модератор ➚ Timur • 2 года назад
```
IList<role> roles =user.Roles.ToList();
foreach (string role in roles)
{

claim.AddClaim(new Claim(ClaimsIdentity.DefaultRoleClaimType, role.Name, ClaimValueTypes.String));
}
```
Ответить • Поделиться ›

Vladimir • 2 года назад
А как выставить срок хранения куков с паролем?
Ответить • Поделиться ›

Артем • 2 года назад
Это просто чудесно. То, что нужно. Замучился связывать MySQL и Identity, а тут такая прелесть. Теперь можно хранить пользователей в MySQL, забирать их оттуда Dapper'ом и авторизовывать при помощи чистых Claims. Ничего лишнего и тяжеловесного, типа EntityFramework или SQL Server. Блеск, спасибо!
Ответить • Поделиться ›

Сергей • 2 года назад
Хроший пример использования голых Claims. Спасибо
Ответить • Поделиться ›

Kostya Vyrodov • 2 года назад
Добавьте пожалуйста статью с обзором по всем способам аутентикации в ASP.NET фреймворке. Очень полезно будет.
Ответить • Поделиться ›

Vladimir • 2 года назад
А по безопасности метод OWIN уступает Identity?
Ответить • Поделиться ›

Metanit Модератор ➚ Vladimir • 2 года назад
Identity также использует OWIN. Тут дело не в безопасности, а в большей простоте и контроле за классами. Когда создаются и используются только те классы, которые непосредственно нужны без лишних зависимостей
Ответить • Поделиться ›

Vladimir ➚ Metanit • 2 года назад
А будет обзор asp net 5?
Ответить • Поделиться ›

Metanit Модератор ➚ Vladimir • 2 года назад
пока ничего не могу сказать
Ответить • Поделиться ›

Артем • 2 года назад
Спасибо за статьи и вообще за ваш сайт! Можно вас попросить также написать статью об авторизации с

помощью OAuth? Или уже есть такая, а я не заметил?

Ответить • Поделиться ›

Metanit Модератор ➚ Артем • 2 года назад

пока по OAuth ничего нет

Ответить • Поделиться ›

Misc • 2 года назад

В чем преимущество перед Identity? По-моему, больше кода - легче запутаться. А смысл тот же.

Ответить • Поделиться ›

Metanit Модератор ➚ Misc • 2 года назад

тут больше кода чем в Identity? В Identity только в пространстве Microsoft.Idenitity.Core порядка 40 классов и интерфейсов. В данном случае просто продемонстрирован подход, как можно воспользоваться преимуществами OWIN, не исполтзуя Identity

Ответить • Поделиться ›