Добавление ролей в ASP.NET Identity

Последнее обновление: 31.10.2015

Сущность ролей в ASP.NET Identity представлена классом IdentityRole, который реализует интерфейс IRole. И мы можем продолжать использовать IdentityRole, но при необходимости также можем переопределить, добавив в него новые свойства. Итак, возьмем проект из прошлой темы и добавим в него функциональность управления ролями.

Для начала добавим в папку Models новый класс ApplicationRole:

```
using Microsoft.AspNet.Identity.EntityFramework;
public class ApplicationRole: IdentityRole
{
    public ApplicationRole(){}

    public string Description { get; set; }
}
```

Новый класс наследует весь функционал от IdentityRole плюс добавляет новое свойство `Description`, которое будет содержать описание роли.

Для управления ролями используется менеджер ролей RoleManager. Поэтому добавим в папку Models новый класс ApplicationRoleManager:

```
class ApplicationRoleManager : RoleManager<ApplicationRole>
{
    public ApplicationRoleManager(RoleStore<ApplicationRole> store)
                : base(store)
    {}
    public static ApplicationRoleManager Create(IdentityFactoryOptions<ApplicationRoleManager> options,
                                            IOwinContext context)
    {
        return new ApplicationRoleManager(new
                RoleStore<ApplicationRole>(context.Get<ApplicationContext>()));
    }
}
```

Опять же наследуем функционал от уже имеющегося класса RoleManager. Метод `Create` позволит классу приложения OWIN создавать экземпляры менеджера ролей для обработки каждого запроса, где идет обращение к хранилищу ролей RoleStore.

И теперь нам надо зарегистрировать менеджер ролей в классе приложения OWIN. Поэтому откроем файл *Startup.cs*, который по предыдущим темам у нас находился в папке App_Start, и изменим его содержимое следующим образом:

```
public class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.CreatePerOwinContext<ApplicationContext>(ApplicationContext.Create);
        app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);

        // регистрация менеджера ролей
        app.CreatePerOwinContext<ApplicationRoleManager>(ApplicationRoleManager.Create);

        app.UseCookieAuthentication(new CookieAuthenticationOptions
        {
```

```
            AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
            LoginPath = new PathString("/Account/Login"),
        });
    }
}
```

Благодаря регистрации менеджер ролей будет использовать тот же контекст данных, что и менеджер пользователей.

Если мы создаем систему ролей в приложении после первого запуска приложения, то нам надо будет произвести миграцию, так как контекст данных у нас изменился в силу изменения системы ролей. Поэтому в Visual Studio в окне Package Manager Console введем команду: enable-migrations и нажмем Enter.

Теперь нам надо создать саму миграцию. Там же в консоли Package Manager Console введем команду:

```
PM> Add-Migration "DataMigration"
```

Visual Studio автоматически сгенерирует класс миграции:

```
public partial class DataMigration : DbMigration
{
    public override void Up()
    {
        AddColumn("dbo.AspNetRoles", "Description", c => c.String());
        AddColumn("dbo.AspNetRoles", "Discriminator", c => c.String(nullable: false, maxLength: 128));
    }

    public override void Down()
    {
        DropColumn("dbo.AspNetRoles", "Discriminator");
        DropColumn("dbo.AspNetRoles", "Description");
    }
}
```

И чтобы выполнить миграцию, применим этот класс, набрав в той же консоли команду:

```
PM> Update-Database
```

Теперь создадим контроллер, который будет выполнять стандартные действия с ролями:

```
using AspNetIdentityApp.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;

public class RolesController : Controller
{
    private ApplicationRoleManager RoleManager
    {
        get
        {
            return HttpContext.GetOwinContext().GetUserManager<ApplicationRoleManager>();
        }
    }

    public ActionResult Index()
    {
        return View(RoleManager.Roles);
    }

    public ActionResult Create()
    {
        return View();
    }
    [HttpPost]
    public async Task<ActionResult> Create(CreateRoleModel model)
    {
        if (ModelState.IsValid)
        {
            IdentityResult result = await RoleManager.CreateAsync(new ApplicationRole
            {
                Name = model.Name,
                Description = model.Description
```

```
            });
            if (result.Succeeded)
            {
                return RedirectToAction("Index");
            }
            else
            {
                ModelState.AddModelError("", "Что-то пошло не так");
            }
        }
        return View(model);
    }

    public async Task<ActionResult> Edit(string id)
    {
        ApplicationRole role = await RoleManager.FindByIdAsync(id);
        if (role != null)
        {
            return View(new EditRoleModel { Id = role.Id,  Name = role.Name, Description = role.Description });
        }
        return RedirectToAction("Index");
    }
    [HttpPost]
    public async Task<ActionResult> Edit(EditRoleModel model)
    {
        if (ModelState.IsValid)
        {
            ApplicationRole role = await RoleManager.FindByIdAsync(model.Id);
            if (role != null)
            {
                role.Description = model.Description;
                role.Name = model.Name;
                IdentityResult result = await RoleManager.UpdateAsync(role);
                if (result.Succeeded)
                {
                    return RedirectToAction("Index");
                }
                else
                {
                    ModelState.AddModelError("", "Что-то пошло не так");
                }
            }
        }
        return View(model);
    }

    public async Task<ActionResult> Delete(string id)
    {
        ApplicationRole role = await RoleManager.FindByIdAsync(id);
        if (role != null)
        {
            IdentityResult result = await RoleManager.DeleteAsync(role);
        }
        return RedirectToAction("Index");
    }
}
```

Это стандартный CRUD-контроллер, выполняющий чтение, удаление, редактирование и добавление ролей. Вначале для взаимодействия с менеджером ролей мы получаем его объект из контекста OWIN:

```
HttpContext.GetOwinContext().GetUserManager<ApplicationRoleManager>()
```

Затем методы менеджера ролей используются для управления ролями. Обратите внимание, что из представлений в методы Create и Edit мы получаем не объект ApplicationRole, а специальные модели EditRoleModel и CreateRoleModel, который могут выглядеть так:

```
public class EditRoleModel
{
    public string Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
}
public class CreateRoleModel
{
    public string Name { get; set; }
    public string Description { get; set; }
}
```

Использование моделей позволит избежать различных проблем с контекстом данных и управлением объектами, которые могли возникнуть, если бы мы напрямую использовали бы ApplicationRole.

И представления будут выглядеть стандартно. Представление *Index.cshtml*:

```
@model IEnumerable<AspNetIdentityApp.Models.ApplicationRole>

@{
    ViewBag.Title = "Роли";
}

<h2>Роли</h2>
<table class="table">
    <tr>
        <th>Название</th>
        <th>Описание</th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>@item.Name</td>
        <td>@item.Description</td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}
</table>
```

Представление *Create.cshtml*:

```
@model AspNetIdentityApp.Models.CreateRoleModel

<h2>Добавление роли</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        @Html.ValidationSummary(true, "")
        <p>Название:  @Html.EditorFor(model => model.Name)</p>
        <p>Описание: @Html.EditorFor(model => model.Description)</p>
        <p><input type="submit" value="Добавить" /></p>
    </div>
}
```

И представление для редактирования *Edit.cshtml*:

```
@model AspNetIdentityApp.Models.EditRoleModel

<h2>Редактирование роли</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        @Html.ValidationSummary(true, "")
        @Html.HiddenFor(model => model.Id)
        <p>Название:  @Html.EditorFor(model => model.Name)</p>
        <p>Описание: @Html.EditorFor(model => model.Description)</p>
        <p><input type="submit" value="Изменить" /></p>
    </div>
}
```

Назад **Содержание** **Вперед**

Доброго времени суток. Все делал по образцу, выдает ошибку В чем может быть дело?:
Ссылка на объект не указывает на экземпляр объекта.
Строка 33: if (ModelState.IsValid)
Строка 34: {
Строка 35: IdentityResult result = await RoleManager.CreateAsync(new Role
Строка 36: {
Строка 37: Name = model.Name,
Исходный файл: e:\OneDrive\Programming\C#\MVCProject\MVCProject\Controllers\RolesController.cs Строка: 35
В чем может быть дело?

1 • Ответить • Поделиться ›

Metanit Модератор → Sergey Pogorelov • 2 года назад
наверное RoleManager не установлен, проверьте правильность определения свойства RoleManager.
• Ответить • Поделиться ›

Mihail Kmet • 2 года назад
Представление Index - непонятное. Передается IEnumerable<applicationrole>, а принимается IEnumerable<applicationuser>. Ну и выводимые свойства соответственно наверное должны быть другими
1 • Ответить • Поделиться ›

Metanit Модератор → Mihail Kmet • 2 года назад
подправил
• Ответить • Поделиться ›

Евгений Иванов • 2 месяца назад
Здравствуйте.
При создании роли ошибка.
The model backing the 'ApplicationContext' context has changed since the database was created. Consider using Code First Migrations to update the database (http://go.microsoft.com/fwl....

Строка 35: if (ModelState.IsValid)
Строка 36: {
Строка 37: IdentityResult result = await RoleManager.CreateAsync(new ApplicationRole
Строка 38: {
Строка 39: Name = model.Name,

Исходный файл: C:\Users\genom\Documents\Visual Studio 2017\Projects\ZeroIdentity\ZeroIdentity.Web\Controllers\RolesController.cs Строка: 37
Пишет, что контекст данных был изменен с момента создания бд.
Я прописал еще раз update-database, но это не помогло.
Заранее спасибо.
• Ответить • Поделиться ›

Metanit Модератор → Евгений Иванов • 2 месяца назад
надо создать миграцию и выполнить ее. Перед
update-database
надо выполнить команду

Add-Migration "DataMigration"
• Ответить • Поделиться ›

Юра • 7 месяцев назад
Добрый вечер.
Я восхищен вашей работой и изложением!
Могли бы Вы исправить небольшую ошибку - " Опять де наследуем функционал от уже имеющегося класса RoleManager ".

Огромнейшее Вам спасибо за ваш труд.
• Ответить • Поделиться ›

Metanit Модератор ➚ Юра • 7 месяцев назад

поправил.

• Ответить • Поделиться ›

Анастасия Васильева • год назад

Огромное спасибо за полезные статьи!!! Подскажите, пожалуйста, в проекте, использующем аналогичный код, как сгенерировать PasswordHash? В миграциях в Configuration.cs в метод Seed добавляю следующий код:

```
protected override void Seed(tfomsSite.Models.AppContext context)
context.Roles.Add(new AppRole
{
Name = "admin",
Description = "Администратор сайта"
});
context.Users.Add(new AppUser
{
Email = "admin@admin.com",
UserName = "admin@admin.com",
PasswordHash = ???
});
}
```

• Ответить • Поделиться ›

Metanit Модератор ➚ Анастасия Васильева • год назад

точно уже не могу сказать, посмотрите в исходных кодах identity , как это делается - https://aspnetidentity.code...

• Ответить • Поделиться ›

Анастасия Васильева ➚ Metanit • год назад

Самый простой способ нашла такой:

```
AppUser user = new AppUser { UserName = "admin2@admin.com", Email = "admin2@admin.com",
PasswordHash = Crypto.HashPassword("123456"), SecurityStamp = "484878787" };
context.Users.Add(user);
AppRole role = new AppRole
{
Name = "admin",
Description = "Администратор сайта"
};
context.Roles.Add(role);
context.SaveChanges();
context.Database.ExecuteSqlCommand("INSERT INTO AspNetUserRoles (UserId, RoleId) VALUES
('"+user.Id+"', '"+role.Id+"')");
```

Мне просто нужно, чтобы администратор был "вшит" в базу, а остальные роли и пользователей он добавлял через административную панель.

А есть какие-то требования к генерации SecurityStamp? Я поискала в интернете, ничего не нашла и просто добавила произвольную строку.

Кириллица по какой-то причине в БД добавляется абракадаброй через метод Seed в Configuration.cs.

• Ответить • Поделиться ›

Metanit Модератор ➚ Анастасия Васильева • год назад

посмотрите в начале статьи http://metanit.com/sharp/mv... как добавляется администратор в базу данных по умолчанию

1 • Ответить • Поделиться ›

Анастасия Васильева ➚ Metanit • 9 месяцев назад

Наконец-то нашла я ответ на... "Кириллица по какой-то причине в БД добавляется

Наконец-то нашла я ответ на - "Кириллица по какой-то причине в БД добавляется абракадаброй через метод Seed в Configuration.cs."
Оказывается, файл Configuration.cs был сконфигурирован в VS 2015 с кодировкой ANSI, а у сайта-то везде кодировка UTF-8.
Теперь осталось найти, где настроить, чтобы файлы создавались в кодировке UTF-8...
• Ответить • Поделиться ›

Igor Lysenko • 2 года назад
Здравствуйте. Я создал две роли Admin и User. Как теперь сделать так чтобы админ мог зайти в админку а юзер нет?
• Ответить • Поделиться ›

Metanit Модератор ➚ Igor Lysenko • 2 года назад
[Authorize(Roles="admin")]
public class AdminController
• Ответить • Поделиться ›

Виктор Алексеев • 2 года назад
Добрый день!

Все замечательно с примером, только Я не понял одного момента - каким образом в представление для редактирования **Edit.cshtml** передается **model.Id** ?

Т.е. в методе **Edit** контроллера при создании переменной класса **EditRoleModel** Мы его никаким образом не задаем - но этот **Id** откуда-то (как черт из табакерки) появляется в представлении.

Даже более того:
1. Если Мы сначала создадим переменную класса **EditRoleModel**,
2. Зададим для **Id** любую белиберду,
3. Передадим эту переменную в представление для редактирования **Edit.cshtml** в качестве модели,

То:
1. Элементы **@Html.HiddenFor(model => model.Id)** или **@Html.EditorFor(model => model.Id)** - все равно будут содержать корректный **Id**.
2. Хотя, элемент **@Html.DisplayTextFor(model => model.Id)** - выведет именно заданную нами белиберду.

Если есть какое-либо разумное объяснение этому - Я был бы рад это услышать :)
• Ответить • Поделиться ›

Metanit Модератор ➚ Виктор Алексеев • 2 года назад
да там должна быть установка в контроллере id.
• Ответить • Поделиться ›

Виктор Алексеев ➚ Metanit • 2 года назад
Речь немного о другом, а именно - о корректности работы внутренней логики **MVC-конвейера** (если это можно назвать таким словом).

Т.е. получается такая картина:
1. При обращении к методу **get** контроллера, мы передаем в него **id** в качестве параметра.
2. По какой-то внутренней (невидимой для нас) логике этот **id** из входного параметра метода перекочевывает в выходное поле модели, а именно в **model.Id**.
3. Даже если мы захотим изменить **model.Id** (внутри **get**-метода) - все равно далее (в модель для представления) перекочует именно тот **id**, который был в качестве входного параметра **get**-метода.

Вот такая вырисовывается несколько нелогичная внутренняя логика (уж извиняйте за каламбурчик) работы **MVC-конвейера**.
1 • Ответить • Поделиться ›

Ярослав Орлов ➚ Виктор Алексеев • 8 месяцев назад
Все очень просто, изначально в индекс выводятся роли - RoleManager.Roles (все поля

включая id). В этом представлении формируются ссылки на редактирование роли включая тот самый id в route data, после чего происходит перенаправление на GET Edit с нашим id из route data и по нему соответственно мы возвращаем модель роли на представления редактирования. После всех махинаций с ролью мы делаем пост обновление всех полей, включая hidden поле с id роли, по которому мы собственно находим в базе нужную роль и методами RoleManager производим все махинации с ней успешно.

Ответить • Поделиться ›

Сергей • 2 года назад

как делается вот такое: пользователь входит и при удачном входе отображается список предметов из БД принадлежащих ему, для другого пользователя другой список. наведите на мысль, спасибо.

Ответить • Поделиться ›

Metanit Модератор ➚ Сергей • 2 года назад

в методе получаем id пользователя и из связанной таблицы по этому id извлекаем все предметы пользователя и передаем их в представление для отображения

Ответить • Поделиться ›

Елена Конорева • 2 года назад

Здравствуйте. По вашему примеру сделала класс ApplicationRole:

При миграции получается 2 таблицы с ролями:
1. dbo.IdentityRoles (Id, Name)
2. dbo.Role (Id -- внешний ключ к предыдущей таблице,Description)

как настроить, чтоб все данные класса ApplicationRole попадали в одну таблицу? Даже если закомментить поле Description в классе, все равно создаются 2 таблицы, в dbo.Role только Id - ссылка

С ApplicationUser все аналогично и нет такой проблемы.
Я переопределила имена таблиц, это как-то может влиять?

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
modelBuilder.Entity<applicationuser>()
.ToTable("User");

modelBuilder.Entity<applicationrole>()
.ToTable("Role");

modelBuilder.Entity<identityuserrole>()
.HasKey(t => new { t.RoleId, t.UserId })
.ToTable("User_Role");

//// .....
}
```

Ответить • Поделиться ›

Елена Конорева ➚ Елена Конорева • 2 года назад

может, кому-то пригодится решение проблемы:

в метод OnModelCreating в самое начало добавить строчку:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
base.OnModelCreating(modelBuilder);

//// ...

//вместо переопределенного класса настроить имя таблицы для родительского:
modelBuilder.Entity<identityrole>()
.ToTable("Role");

// а это убираем
//modelBuilder.Entity<applicationrole>()
```

```
//modelBuilder.Entity<ApplicationRole>()
// .ToTable("Role");

}
```

но все равно мне не понятно, почему логика работы переопределенных классов разная и для Ролей нужны какие-то доп. манипуляции, а для юзеров - нет? Или я что-то упускаю?

• Ответить • Поделиться ›

Роман • 2 года назад

Здравствуйте! Подскажите, пожалуйста, если я создал классы-наследники ApplicationUser и ApplicationRole, нужно ли в них переопределять свойства соответственно Roles и Users, чтобы свойства возвращали коллекции уже новых объектов ApplicationRole и ApplicationUser, а не старых IdentityRole и IdentityUser??

• Ответить • Поделиться ›

Metanit Модератор ➚ Роман • 2 года назад

нет, Roles и Users переопределять не надо

• Ответить • Поделиться ›

Петр Гаврилов • 2 года назад

Объясните кто-то пожалуйста, я вот создал ASP net identity с нуля как тут, все работает отлично, создавал ту которую студия предлагает, все ок. Но собственно, после того как все зарегались и залогинились, мне нужно дальше что-то создавать в базе, кароче работать с данными которые не связанны с инфой о пользователях, что нужно создавать уже совершенно новый контекст для работы с базой или я могу как-то использовать тот что был создан для пользователей?

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

можно использовать тот же самый контекст, только при добавлении новых таблиц надо провести миграции

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

Я понял, спс дошло, но получается что этот контекст(ApplicationContext) из примера, я могу использовать при условии Code First, а если у меня уже есть база, и вообще я не люблю подход Code First. Мне проще в конструкторе модели накидать или в базе таблицы создать, потом нажал Update from или все обновилось, вобщем вопрос в чем, я создал систему авторизаци все окей пользователи создаются, логинятся и т.д. Теперь в базе создаю таблицу SomeTable, есть у меня возможность как-то заставить ApplicationContext увидеть эту таблицу и уметь ней пользоваться?

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

надо сделать миграции http://metanit.com/sharp/mv...
даже если вы вручную создаете базу данных, а потом к ней подключаетесь, это тоже code first

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

До меня дошло что я от вас хочу услышать))) Можно ли как-то всю эту кухню с Identity реализовать по принципу MODEL FIRST?

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

слабо себе представляю, как все это инкорпорировать через model frst. Наверное лучше придерживаться code first, особенно учитывая что model first не несет никаких преимуществ

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

Ну в вот плане model first удобнее для меня, что я с ней только и работал, есть база, из нее сгенерились классы, связи и т.д. А при code first нужно руками создавать связи и все остальное. В любом случае спасибо за ответы, и вообще за то что вы делаете, очень все доступно и понятно описанно.

Ответить • Поделиться ›

Metanit Модератор → Петр Гаврилов • 2 года назад

не, это видимо вы имеете ввиду не model first, а database first, когда по имеющейся уже бд создается полный набор классов

Но в code first тоже можно все это автоматически генерировать без ручного создания классов. Как это делать, описано тут - http://metanit.com/sharp/en...

Ответить • Поделиться ›

Петр Гаврилов → Metanit • 2 года назад

Скажите а зачем у вас в классе ApplicationContext есть метод Create соторый каждый раз создает новый класс, почему не сделать синглтон?

Ответить • Поделиться ›

Metanit Модератор → Петр Гаврилов • 2 года назад

ну можно сделать и через синглтон

Ответить • Поделиться ›

Петр Гаврилов → Metanit • 2 года назад

Я пытаюсь осмыслить жизненый цикл вэб приложения и все никак не пойму, если я со своего компа вызываю страничку, класс Startup и его метод Configuration вызовутся только раз сколько бы я потом окно не создал и сколько браузеров не запустил(хотя конструкторы контроллеров создаются для каждого нового запроса), если в это время кто-то с другой машины тоже обратится к какой-то странице, Startup и его метод Configuration вызовутся у него свои, или эти классы общие для всех кто конектится и они создаются только один раз когда-то там когда сайт стартанул на IIS?

Ответить • Поделиться ›

Metanit Модератор → Петр Гаврилов • 2 года назад

Startup запускается один раз на старте приложения (не для запроса, а для всего приложения)

Ответить • Поделиться ›

Петр Гаврилов → Metanit • 2 года назад

Спасибо за ответы ваши, скажите мы не можем с вами пообщаться где-то на просторах инета, а то не хочеться здесь загаживать, а у меня к вам еще есть вопросы, по MVC.) Или продолжать тут?

Ответить • Поделиться ›

Metanit Модератор → Петр Гаврилов • 2 года назад

можете тут писать

Ответить • Поделиться ›

Петр Гаврилов → Metanit • 2 года назад

Ок тогда попробую все и сразу.
1. Если я правильно понял когда я развернул приложение на IIS, сайт запустился, то сразу отрабатывают методы protected void Application_Start(), класс Startup и его метод Configuration и т.д.?
2. Потом если кто-то обращается к контроллеру то вызывается контсруктор этого контроллера, ну а далльше метод действия, при чем если я буду запускать хоть сто окон, каждый раз будет создаваться новый допустим HomeController и его конструктор, такой вопрос, если я в это контроллере создам статическую переменную, и буду ее инкрементить в конструкторе, я пробовал делать с одного компа, она одна для всех окон, она растет в каждом новом окне, собственно вопрос, если кто-то запрашивает этот же

контроллер с другой машины, у него эта переменная уже наинкременченная или у этого пользователя своя песочница и переменные тоже свои?
3. По-поводу работы с базой, я создал с нуля как у вас систему аутентификации, добавил нужные таблицы в базу все ок все работает, собственно теперь я понял почему не нужно делать через синглтон (Скажите а зачем у вас в классе ApplicationContext есть метод Create соторый каждый раз создает новый класс, почему не сделать синглтон?) он и так этот контекст создается один раз, начал делать через паттерн UnitForWork, создав класс по типу

показать больше

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

1.да
2. если обычная переменная, то она создается вместе с контроллером заново для каждого запроса. Если статиечская переменная, то она общая для всех запросов
3. смотря как вы применяете UoW
4. контекст должен как можно быстрее уничтожаться. В MVC для каждого запроса, как правило, создается свой объект контекста. Поэтому тут все нормально

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

Спс все понятно теперь, только по третьему пункту, UoW точно как вот у вас в последней главе, если смотерть на

```
public HomeController(IOrderService serv)
{
orderService = serv;
}
```

то получается что при каждом запросе по цепочке будет создаваться

```
public EFUnitOfWork(string connectionString)
{
db = new MobileContext(connectionString);
}
```

получается что десять одновременных запросов к HomeController сздадут 10 обектов EFUnitOfWork, так как конструктор HomeController будет вызываться для каждого запроса. Исходя из вашего ответа по п 3. можно сделать так чтоб этого не происходило, если можно то в кратце как это сделать? Ну и нужно ли вообще, потому как я пытаюсь понять не накладно ли это для системы каждый раз для обращения к контоллеру создавать класс.

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

а ну да, так и должно быть - для каждого запроса свой контекст

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

Снова я, появилось пару вопросов,
1.В описании "Использование объектов Cliam" вы описываете как все делается, при том что система создана по шаблону identity который предлагает студия, если я создал все с нуля, то как я понимаю если я добавлю в ApplicationUser метод generateUserIdentityAsync то буду вынужден создавать класс ApplicationSignInmanager как это делает стандартная система?
2. Второй вопрос вытикает из первого, при логине в системе с нуля у вас есть такой код

```
ClaimsIdentity claim = await UserManager.CreateIdentityAsync(user,
DefaultAuthenticationTypes.ApplicationCookie);
AuthenticationManager.SignOut();
AuthenticationManager.SignIn(new AuthenticationProperties
{
IsPersistent = true
}, claim);
```

Получается я просто здесь могу добавить с объект claim нужные мне данные и они будут доступны пока пользователь в системе, есть разница, добавлять здесь при логине или в методе generateUserIdentityAsync который создается в стандартной системе?
3. Опять же в методе Login в системе с нуля вы вызываете SingIn а стандартная система

показать больше

• Ответить • Поделиться ›

Metanit Модератор ➚ Петр Гаврилов • 2 года назад

1. Да большинство классов придется создавать такими же, как и в стандартной системе. Возможно легче сразу взять стандартную систему
2. Claim по сути просто добавляет куки с некоторыми значениями. Фактически можно сами создать классы, которые взаимодействуют с куками и выполняют роль claimов
3. Я просто пытался предельно упростить пример по сравнению со стандартной схемой. Хотя по факту все то же самое делается - проверяется логин и пароль пользователя. Стандартная схема просто создает набор дополнительных классов.
SignOut вызывается перед SignIn чтобы удалить все куки, если пользователь ранее был залогинен
4. Это посто простейшая система авторизации с использованием Identity. При желании вы можете добавить в нее дополнительные классы, в том числе и SignInManager
private IAuthenticationManager AuthenticationManager используется, чтобы получить менеджера аутентификации для текущего контекста запроса, чтобы потом этот менеджер использовать для авторизации пользователя.
Просто не надо зацикливаться на стандартной реализации, которая есть в Identity. Она только за два последних года сто раз менялась и еще сто раз поменяется, главное уловить суть взаимодействий

• Ответить • Поделиться ›

Петр Гаврилов ➚ Metanit • 2 года назад

Ок, спс, за ответы и терпение.)

• Ответить • Поделиться ›

Alexander Rezvanov • 2 года назад

Как можно в представлении получить роль пользователя?

делаю так:

//В контроллере RolesController добавил метод

```
public IEnumerable<aspnetidentityapp.models.applicationrole> getRoleUser()
{
return RoleManager.Roles;
}
```

В представлении пишу так:

```
@{
var r = new AspNetIdentityApp.Controllers.RolesController();
var s = r.getRoleUser();
}
```

В результате получаю ошибку в этом месте:

```
private ApplicationRoleManager RoleManager
```

показать больше

• Ответить • Поделиться ›

Metanit Модератор ➚ Alexander Rezvanov • 2 года назад

смотрите пример в конце статьи http://metanit.com/sharp/mv...

• Ответить • Поделиться ›

Alexander Rezvanov ➚ Metanit • 2 года назад

Та же самая ошибка

в контроллере:

```
public IList<string> getUSer()
{
IList<string> roles = new List<string> { "Роль не определена" };
ApplicationUserManager userManager = HttpContext.GetOwinContext()
.GetUserManager<applicationusermanager>();
ApplicationUser user = userManager.FindByEmail(User.Identity.Name);
if (user != null)
roles = userManager.GetRoles(user.Id);
```