

ASP.NET Core Identity

2018年12月8日 14:20

■ 认证 (Authentication) 与授权 (Authorization)

你是谁? 能干什么?

■ 使用 ASP.NET Core Identity 身份认证框架

认证与授权 - 基于声明的认证

来自 <<https://www.jianshu.com/p/cda95dff698c>>

- 1、创建项目，选择 Identity 框架（数据库）
- 2、替换现有的界面
- 3、认证后访问： AuthorizeAttribute

如果一个项目已经创建，也可后期通过基架创建身份认证。

创建完整的标识 UI 源：

来自 <[微软官方文档](#)>

通过 PersonalDataAttribute 自定义用户数据

实体类型：

User	用户
Role	角色
UserClaim	用户的声明
UserToken	身份验证令牌
UserLogin	用户的登录名
RoleClaim	角色的声明
UserRole	用户与角色关联

实体类型之间的关系：

每个 User 具有许多 UserClaims
每个 User 具有许多 UserLogins
每个 User 具有多个 UserTokens
每个 Role 具有多个 RoleClaims
每个 User 都可以具有许多 Roles, 由 UserRole 关联。

具体表之间的映射关系在 IdentityDbContext 中实现

默认模型配置

来自 <[微软官方文档](#)>

```
dbo.AspNetRoleClaims  
dbo.AspNetRoles  
dbo.AspNetUserClaims  
dbo.AspNetUserLogins  
dbo.AspNetUserRoles  
dbo.AspNetUsers  
dbo.AspNetUserTokens
```

```
IdentityUser  
IdentityRole  
IdentityUserClaim  
IdentityUserToken  
IdentityUserLogin  
IdentityRoleClaim  
IdentityUserRole
```

自定义模型

1、提供自定义实体和主键类型作为泛型参数

定义主键类型: IdentityDbContext<IdentityUser, IdentityRole, string>

无需角色: IdentityUserContext<TUser, string>

演示: 自定义一个用户数据

```
public class ApplicationUser : IdentityUser  
{  
    public string CustomTag { get; set; }  
}
```

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
```

```
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }
}
```

```
services.AddDefaultIdentity<ApplicationUser>().AddEntityFrameworkStores<ApplicationDbContext>
()
```

演示：更改主键类型

```
public class ApplicationDbContext : IdentityDbContext<IdentityUser<Guid>, IdentityRole<Guid>,
Guid>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }
}
```

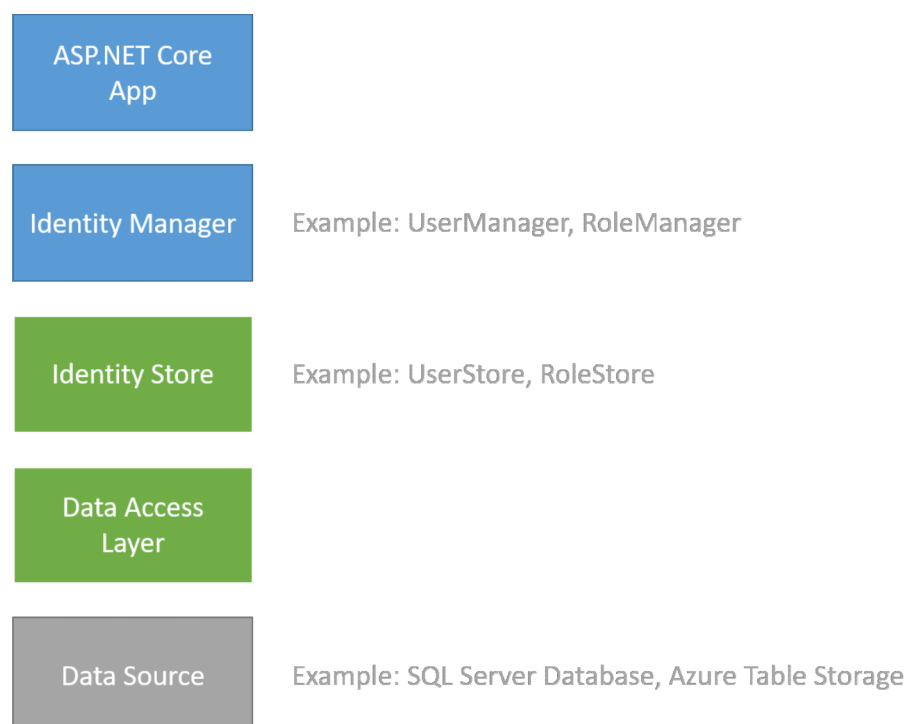
```
services.AddDefaultIdentity<IdentityUser<Guid>>
().AddEntityFrameworkStores<ApplicationDbContext>();
```

2、重写 OnModelCreating 修改映射关系

■ 自定义存储程序

支持不同的持久化，表结构，访问方法如 Dapper

体系结构，主要包含：管理和存储，这样的架构可以随意替换持久层。



```
public void ConfigureServices(IServiceCollection services)
{
    // Add identity types
    services.AddIdentity<ApplicationUser, ApplicationRole>().AddDefaultTokenProviders();

    // Identity Services
    services.AddTransient<IUserStore<ApplicationUser>, CustomUserStore>();
    services.AddTransient<IRoleStore<ApplicationRole>, CustomRoleStore>();
    string connectionString = Configuration.GetConnectionString("DefaultConnection");
    services.AddTransient<SqlConnection>(e => new SqlConnection(connectionString));
    services.AddTransient<DapperUsersTable>();

    // additional configuration
}
```

- [IUserRoleStore](#)
- [IUserClaimStore](#)
- [IUserPasswordStore](#)
- [IUserSecurityStampStore](#)
- [IUserEmailStore](#)
- [IPhoneNumberStore](#)
- [IQueryableUserStore](#)
- [IUserLoginStore](#)
- [IUserTwoFactorStore](#)
- [IUserLockoutStore](#)