Заполнение базы данных для ASP.NET Identity

1 Одной из проблем в созданном нами ранее приложении, является открытый доступ к панели администратора, использующей контроллеры Admin и RoleAdmin. Если мы ограничим доступ к этим контроллерам, то получим «классическую проблему яйца и курицы» - Admin и RoleAdmin используются для управления пользователями и ролями, но как нам создать первого администратора, если мы ограничили доступ к этим контроллерам? Эта проблема становится актуальной при первом развертывании приложения на сервере, когда еще ни один пользователь не создан.

Решением этой проблемы является заполнение базы данных первоначальными данными при ее создании средствами Entity Framework. Это позволит автоматически создать пользователей и назначить им роли при первом развертывании приложения. Для этого необходимо использовать *метод* *PerformInitialSetup* класса **IdentityDbInit**, являющегося специфичным для Entity Framework и ASP.NET Identity. В следующем примере показано, какие изменения необходимо внести в файл AppIdentityDbContext.cs, находящийся в папке Infrastructure:

```
// ...
using Microsoft.AspNet.Identity;

namespace Users.Infrastructure
{
    // ...

    public class IdentityDbInit : DropCreateDatabaseIfModelChanges<AppIdentityDbC
    {
        // ...

        public void PerformInitialSetup(AppIdentityDbContext context)
        {
            AppUserManager userMgr = new AppUserManager(new UserStore<AppUser>(co
            AppRoleManager roleMgr = new AppRoleManager(new RoleStore<AppRole>(co

            string roleName = "Administrators";
            string userName = "Admin";
            string password = "mypassword";
            string email = "admin@professorweb.ru";

            if (!roleMgr.RoleExists(roleName))
            {
                roleMgr.Create(new AppRole(roleName));
            }

            AppUser user = userMgr.FindByName(userName);
            if (user == null)
            {
                userMgr.Create(new AppUser { UserName = userName, Email = email },
                    password);
                user = userMgr.FindByName(userName);
            }

            if (!userMgr.IsInRole(user.Id, roleName))
            {
                userMgr.AddToRole(user.Id, roleName);
            }
        }
    }
}
```

Пройди тесты C# тест (легкий) .NET тест (средний)

Обратите внимание, что в этом примере мы использовали синхронные методы управления пользователями и ролями. Ранее я говорил, что использую преимущественно асинхронные методы, но синхронные методы бывают полезными, когда необходимо обеспечить порядок вызова этих методов.

В этом примере нам необходимо создавать экземпляры AppUserManager и AppRoleManager напрямую, поскольку метод PerformInitialSetup() вызывается до запуска конфигурации OWIN. Мы использовали классы RoleManger и AppManager для создания роли администратора и добавления пользователя в эту роль.

После внесения этих изменений, можно использовать атрибут Authorize для защиты контроллеров Admin и RoleAdmin, как показано в примерах ниже:

```
[Authorize(Roles = "Administrators")]
public class AdminController : Controller
{
        // ...
}
```

```
[Authorize(Roles = "Administrators")]
public class RoleAdminController : Controller
{
        // ...
}
```

База данных заполняется данными по умолчанию только при создании схемы, это означает, что нам необходимо сбросить базу данных. Это конечно нельзя использовать в реальном приложении, а создание роли администратора и админа необходимо закладывать в самом начале проектирования приложения. Но в примере нашего приложения я отложил это до данного момента, т. к. хотел сначала продемонстрировать возможности ASP.NET Identity по администрированию пользователей и ролей, а уже потом озадачить

Чтобы сбросить базу данных вы можете щелкнуть по неи правои кнопкои мыши в приложении Microsoft SQL Server Managment Studio и выбрать команду Delete. Для

создания базы данных щелкните правой кнопкой мыши по узлу Data Connections в окне Server Explorer и выберите в контекстном меню команду Create New SQL Server Database. В открывшемся модальном окне задайте название для базы IdentityDb.

После этого запустите приложение и запросите URL-адрес /Admin/Index или /RoleAdmin/Index. Вы заметите задержку в несколько секунд — Entity Framework воссоздает схему базы данных и заполняет ее первоначальными данными, а затем вам будет предложено ввести учетные данные. Используйте данные администратора, которые мы определили в примере выше.

Итак, в последних трех статьях я показал вам, как используется ASP.NET Identity для базовой настройки систем аутентификации и авторизации. Мы рассмотрели, как Identity лаконично вписывается в жизненный цикл приложений ASP.NET, позволяя перехватывать запросы, проверять учетные записи пользователей на предмет аутентификации и ограничивать доступ к методам действий на основе ролей. В последующих статьях мы рассмотрим расширенные настройки ASP.NET Identity.