Создание приложения с ASP.NET Identity с нуля

Последнее обновление: 07.12.2015

При создании проекта с типом аутентификации Individual User Accounts в него по умолчанию добавляется все необходимые файлы для работы с AspNet Identity. Однако, как правило, редко востребован весь стандартный функционал. Если нам нужна только регистрация и логин, то остальные файлы и неиспользуемый код будут просто висеть в проекте, либо их придется удалять. Однако мы можем и выбрать любой другой тип проекта и в него уже добавить вручную функционал AspNet Identity, причем только тот, который нам будет нужен. Это даст нам больший контроль над тем кодом, который размещается в проекте.

Например, создадим обычный проект MVC с типом аутентификации `No Authentication`.

Чтобы добавить в проект AspNet Identity, нам надо добавить следующие NuGet-пакеты:

- Microsoft.AspNet.Identity.EntityFramework
- Microsoft.AspNet.Identity.OWIN
- Microsoft.Owin.Host.SystemWeb

После добавления пакетов надо обновить файл *web.config*: добавим строку подключения:

```
<connectionStrings>
  <add name="IdentityDb" providerName="System.Data.SqlClient"
    connectionString="Data Source=(localdb)\v11.0;AttachDbFilename=|DataDirectory|\IdentityDb.mdf;Integrated
Security=True;"/>
</connectionStrings>
```

Теперь добавим классы пользователей и контекста данных в папку Models. Класс контекста будет наследоваться от IdentityDbContext:

```
using System.Data.Entity;
using Microsoft.AspNet.Identity.EntityFramework;

public class ApplicationContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationContext() : base("IdentityDb") { }

    public static ApplicationContext Create()
    {
        return new ApplicationContext();
    }
}
```

Класс пользователей:

```
using Microsoft.AspNet.Identity.EntityFramework;

public class ApplicationUser : IdentityUser
{
```

```
        public int Year { get; set; }
    public ApplicationUser()
    {
    }
}
```

В классе пользователя кроме унаследованных от IdentityUser свойств также определяется и свойство Year для хранения года рождения пользователя.

Поскольку вся работа с пользователями идет не напрямую, а через менеджер пользователей, то также добавим в папку Models соответствующий класс:

```
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;

public class ApplicationUserManager : UserManager<ApplicationUser>
{
    public ApplicationUserManager(IUserStore<ApplicationUser> store)
                        : base(store)
    {
        }
    public static ApplicationUserManager Create (IdentityFactoryOptions<ApplicationUserManager> options,
                                            IOwinContext context)
    {
        ApplicationContext db = context.Get<ApplicationContext>();
        ApplicationUserManager manager = new ApplicationUserManager(new UserStore<ApplicationUser>(db));
        return manager;
    }
}
```

Класс менеджера пользователей наследуется от `UserManager`. В конструкторе он принимает объект хранилища пользователей IUserStore. А статический метод `Create()` создает экземпляр класса ApplicationUserManager с помощью объекта контекста `IOwinContext`

Последний шаг в первоначальной настройке проекта для AspNet Identity состоит в добавлении в проект файла запуска приложения OWIN. Итак, добавим в папку App_Start файл *Startup.cs* со следующим содержанием:

```
using Microsoft.Owin;
using Owin;
using AspNetIdentityApp.Models;
using Microsoft.Owin.Security.Cookies;
using Microsoft.AspNet.Identity;

[assembly: OwinStartup(typeof(AspNetIdentityApp.Startup))]

namespace AspNetIdentityApp
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            // настраиваем контекст и менеджер
            app.CreatePerOwinContext<ApplicationContext>(ApplicationContext.Create);
            app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login"),
            });
        }
    }
}
```

Интерфейс `IAppBuilder` определяет множество методов, в данном случае нам достаточно трех методов. Метод `CreatePerOwinContext` регистрирует в OWIN менеджер пользователей ApplicationUserManager и класс контекста ApplicationContext.

Метод `UseCookieAuthentication` с помощью объекта `CookieAuthenticationOptions` устанавливает аутентификацию на основе куки в качестве типа аутентификации в приложении. А свойство `LoginPath` позволяет установить адрес URL, по которому будут перенаправляться неавторизованные пользователи. Как правило, это адрес */Account/Login*.

Это минимально необходимая настройка проекта для использования AspNet Identity, на основе которой мы уже сможем создавать всю остальную систему авторизации и аутентификации.

Назад **Содержание** **Вперед**