عنوان: **ایجاد سرویس Account Manager با تکنولوژی های Identity 2.1 و Web API 2.2**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۲۲ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Security, ASP.NET Web API, AngularJS, ASP.NET Identity, JSON Web Token

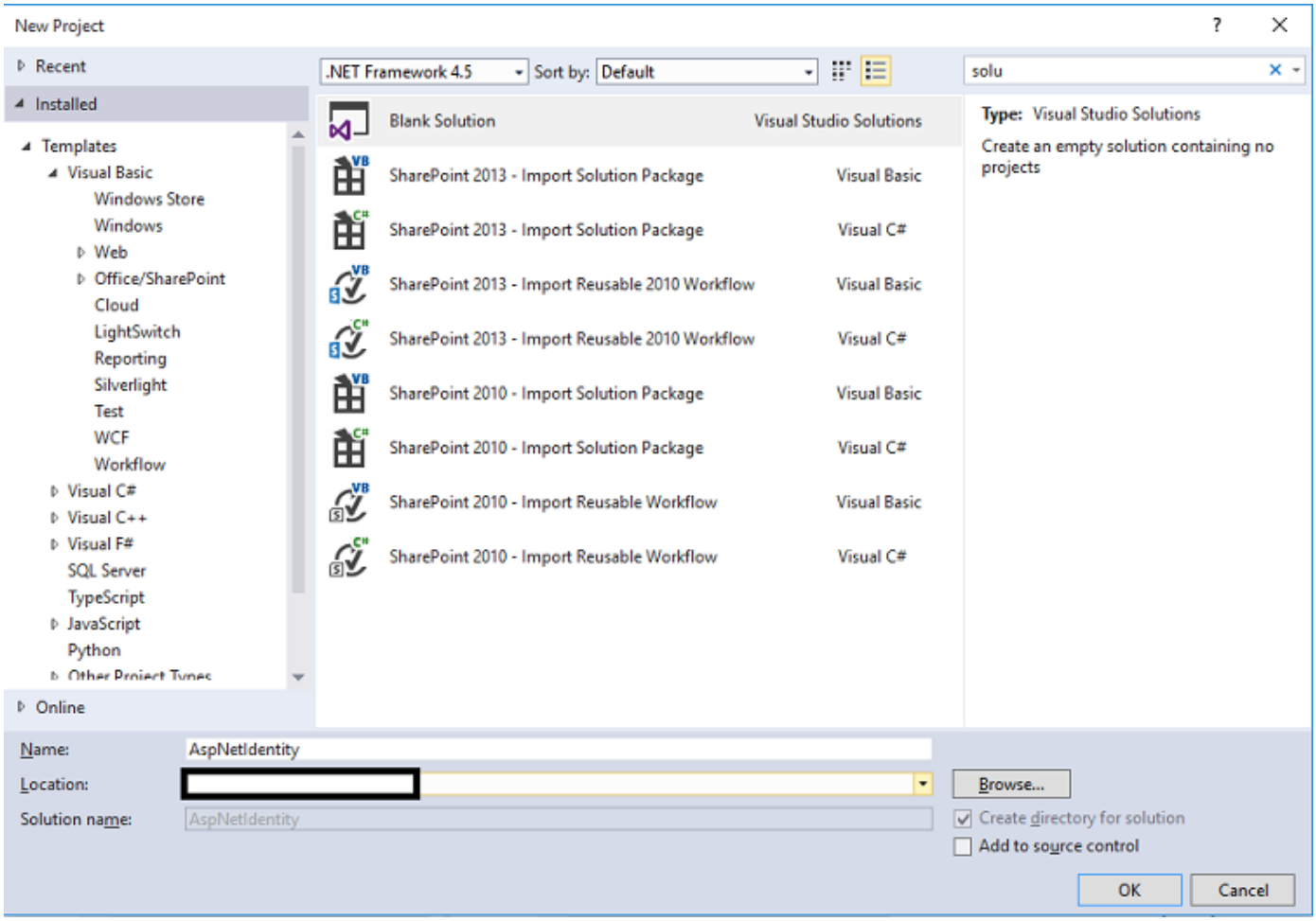
ASP.NET Identity 2.1 جدیدترین فریم ورک عضویت و مدیریت کاربر است که چندی پیش توسط شرکت مایکروسافت منتشر شد. این سیستم عضویت می‌تواند به تمامی فریم‌ورک‌های دات نتی مانند MVC، Web API و ... متصل گردد.
در این دوره چند قسمتی به همراه یک پروژه‌ی نمونه، نحوه‌ی ارتباط Identity و Web API را نمایش خواهیم داد. در قسمت front-end این پروژه‌ی SPA، ما از AngularJs استفاده خواهیم نمود. قسمت front-end که توسط AngularJs توسعه داده می‌شود از bearet token based authentication استفاده می‌کند. این متد از JSON Web Token برای فرایند Authorization بهره می‌گیرد که به اختصار آن را JWT نیز می‌نامند. این روش سیستم اعتبار سنجی role based بوده و تمامی آنچه را که در یک سیستم membership داریم، پوشش می‌دهد. توجه داشته باشید که برای فهم بهتر تمامی مراحل، ما پروژه را بدون هیچ قالب از پیش تعریف شده‌ای در VS2013 آغاز می‌کنیم.
به دلیل اینکه پروژه در طی چند قسمت انجام می‌پذیرد آن را به بخش‌های زیر تقسیم می‌کنیم.
تنظیمات اولیه ASP.NET Identity 2.1 با Web API
ایجاد Account Confirmation به وسیله Identity به همراه تنظیمات policy برای user name و password
توسعه OAuth Json Web Token Authentication به کمک Web API و Identity
ایجاد یک سیستم Role Based و تایید صلاحیت‌های مربوط به آن
توسعه Web API Claims Authorization در Identity 2.1
توسعه بخش front-end با AngularJs

تنظیمات اولیه ASP.NET Identity 2.1 با Web API

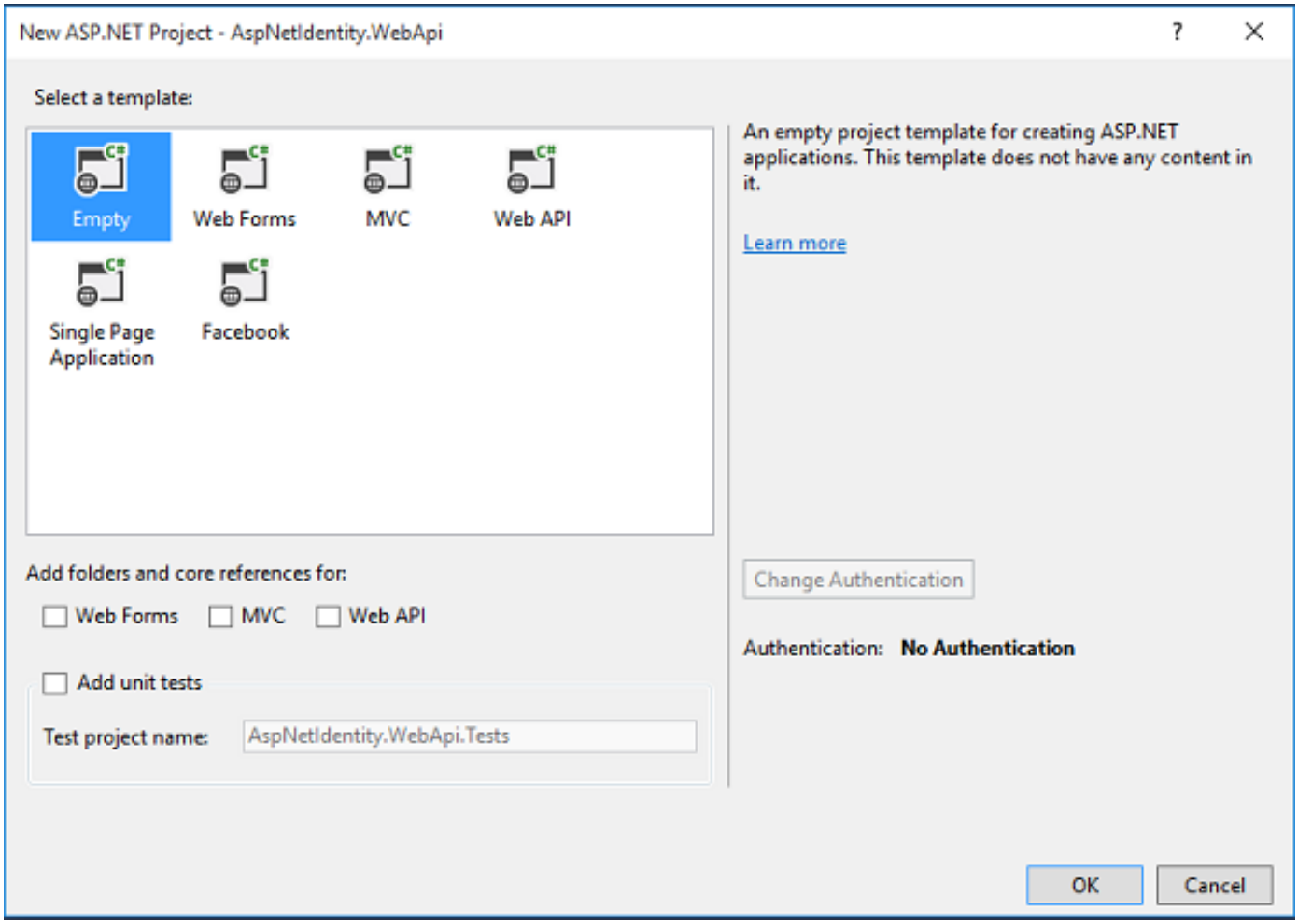
1. تنظیمات ASP.Net Identity 2.1

1-1. ساخت یک پروژه Web API

در ابتدا ما یک empty solution را با نام "AspNetIdentity" همانند شکل مقابل می‌سازیم.



یک ASP.NET Web Application با نام "AspNetIdentity.WebApi" را به این solution اضافه می‌نماییم. در بخش select template ما empty template را انتخاب می‌کنیم. همچنین در قسمت add folders and core references for: نیز هیچیک از گزینه‌ها را انتخاب نمی‌کنیم.



1-2. نصب packageهای مورد نیاز از Nuget

در زیر packageهای مورد نیاز برای ASP.NET Web API و Owin را مشاهده میکنید. همچنین packageهای مربوط به ASP.Net Identity 2.1 نیز در زیر قرار داده شده‌اند.

```
Install-Package Microsoft.AspNet.Identity.Owin -Version 2.1.0
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.1.0
Install-Package Microsoft.Owin.Host.SystemWeb -Version 3.0.0
Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.2.2
Install-Package Microsoft.Owin.Security.OAuth -Version 3.0.0
Install-Package Microsoft.Owin.Cors -Version 3.0.0
```

1-3. اضافه کردن user class و database context

حال که تمامی پکیج‌های مورد نیاز را به پروژه خود اضافه نمودیم، قصد داریم تا اولین کلاس EF را با نام "ApplicationUser" به پروژه اضافه کنیم. این کلاس، کاربری را که قصد ثبت نام در membership system، دارد را نمایش می‌دهد. برای این کار ما یک کلاس جدید را به نام "ApplicationUser" می‌سازیم و کلاس "Microsoft.AspNet.Identity.EntityFramework.IdentityUser" را در آن به ارث می‌بریم.

برای این کار ما یک پوشه‌ی جدید را در برنامه با نام "Infrastructure" می‌سازیم و درون آن کلاس ذکر شده را اضافه می‌کنیم:

```
public class ApplicationUser : IdentityUser
    {
        [Required]
        [MaxLength(100)]
        public string FirstName { get; set; }
```

```
        [Required]
        [MaxLength(100)]
        public string LastName { get; set; }

        [Required]
        public byte Level { get; set; }

        [Required]
        public DateTime JoinDate { get; set; }

    }
```

پس از افزودن کلاس User، نوبت به اضافه نمودن Db Context است. این کلاس وظیفه‌ی ارتباط با پایگاه داده را بر عهده دارد. ما یک کلاس جدید را با نام ApplicationDbContext، به پوشه‌ی Infrastructure اضافه می‌نماییم. کد مربوط به این کلاس به صورت زیر است:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
            Configuration.ProxyCreationEnabled = false;
            Configuration.LazyLoadingEnabled = false;
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }

    }
```

همانطور که ملاحظه می‌کنید این کلاس از IdentityDbContext ارث بری نموده است. این کلاس یک نسخه‌ی جدیدتر از DbContext است که تمامی نگاشت‌های Entity Framework Code First را انجام می‌دهد. در ادامه ما یک Connection String را با نام DefaultConnection در فایل web.config اضافه می‌نماییم.

همچنین متد static ایی را با نام Create که در تکه کد فوق از سوی Owin Startup class فراخوانی می‌گردد که در ادامه به شرح آن نیز خواهیم پرداخت.

ConnectionString قرار داده شده در فایل web.config در قسمت زیر قرار داده شده است:

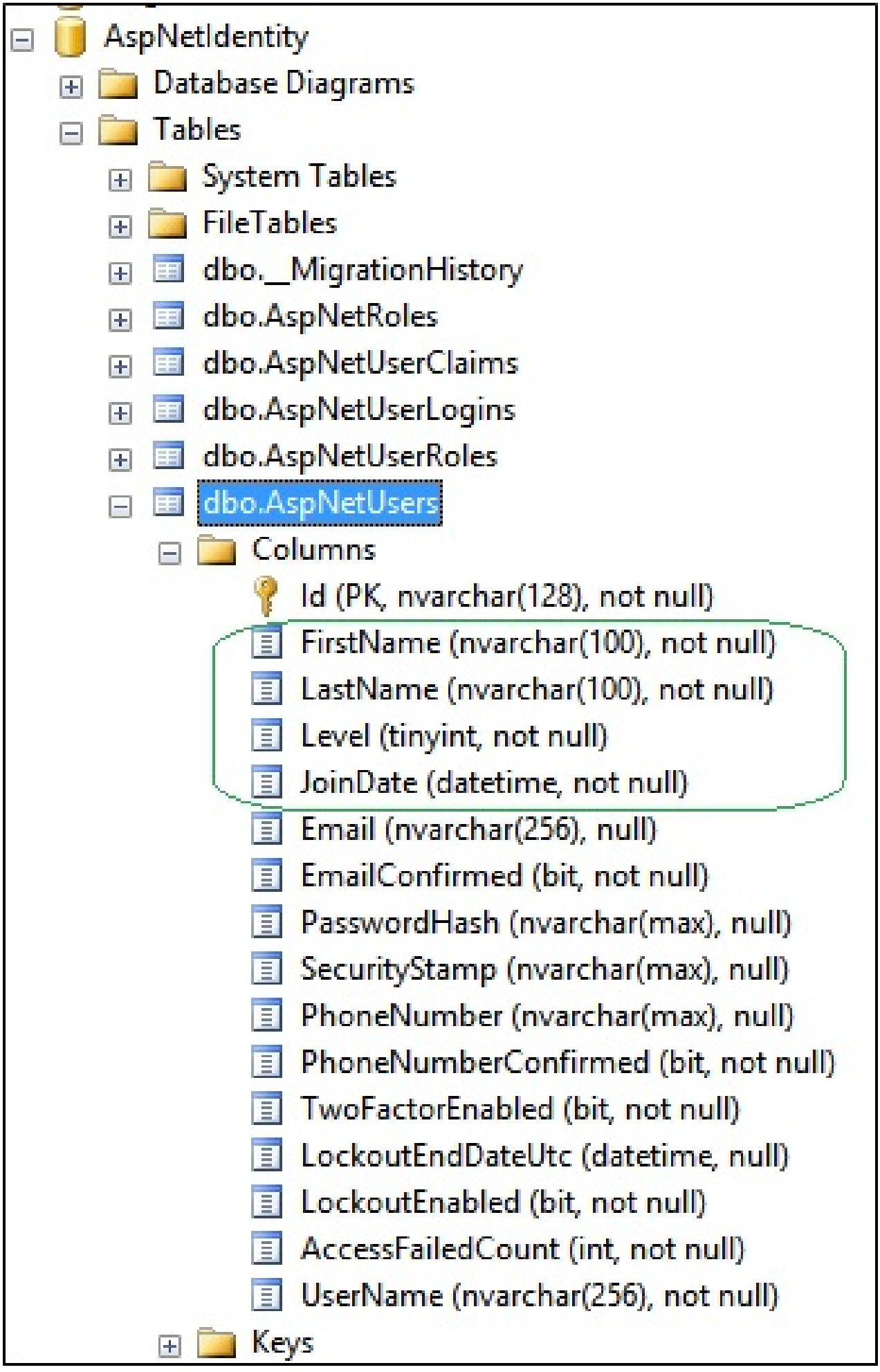
```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=.\sqlexpress;Initial
Catalog=AspNetIdentity;Integrated Security=SSPI;" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

قدم 4: ساخت پایگاه داده و فعال سازی DB Migration

حال ما باید EF CodeFirst Migration را برای آپدیت کردن دیتابیس، بجای دوباره ساختن آن به ازای هر تغییری، فعال نماییم.
برای این کار بایستی در قسمت NuGet Package Manager Console عبارت زیر را وارد نماییم:

```
enable-migrations
add-migration InitialCreate
```

اگر تا به اینجای کار تمامی مراحل به درستی صورت گرفته باشند، پوشه‌ی Migration با موفقیت ساخته می‌شود. همچنین پایگاه داده متشکل از جداول مورد نیاز برای سیستم Identity نیز ایجاد می‌شود. برای اطلاعات بیشتر می‌توانید مقالات مرتبط با Code First را مطالعه نمایید. ساختار جداول ما باید به صورت زیر باشد:


AspNetIdentity
Database Diagrams
Tables
System Tables
FileTables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Columns
Id (PK, nvarchar(128), not null)
FirstName (nvarchar(100), not null)
LastName (nvarchar(100), not null)
Level (tinyint, not null)
JoinDate (datetime, not null)
Email (nvarchar(256), null)
EmailConfirmed (bit, not null)
PasswordHash (nvarchar(max), null)
SecurityStamp (nvarchar(max), null)
PhoneNumber (nvarchar(max), null)
PhoneNumberConfirmed (bit, not null)
TwoFactorEnabled (bit, not null)
LockoutEndDateUtc (datetime, null)
LockoutEnabled (bit, not null)
AccessFailedCount (int, not null)
UserName (nvarchar(256), not null)
Keys

در بخش بعدی، کلاس‌های مربوط به UserManager را ایجاد خواهیم کرد و پس از آن فایل Startup Owin را برای مدیریت کاربران، تشریح می‌کنیم.