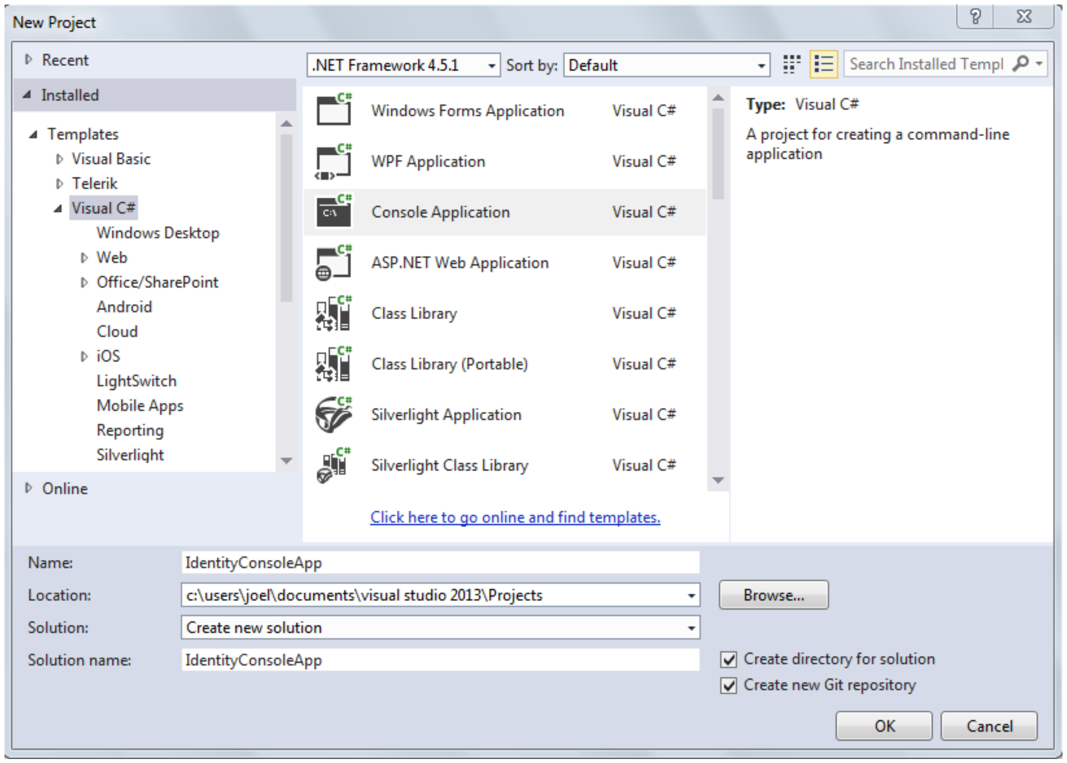
ASP.NET Identity

ASP.NET Identity

Setting up a database using ASP.NET Identity

ASP.NET Identity

We will start by creating a new console application. We will call it "IdentityConsoleApp"


New Project
Recent
Installed
Templates
Visual Basic
Telerik
Visual C#
Windows Desktop
Web
Office/SharePoint
Android
Cloud
iOS
LightSwitch
Mobile Apps
Reporting
Silverlight
Online
.NET Framework 4.5.1
Sort by: Default
Search Installed Templ
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
Click here to go online and find templates.
Browse...
OK
Cancel

Type: Visual C#

A project for creating a command-line application

Name: IdentityConsoleApp

Location: c:\users\joel\documents\visual studio 2013\Projects

Solution: Create new solution

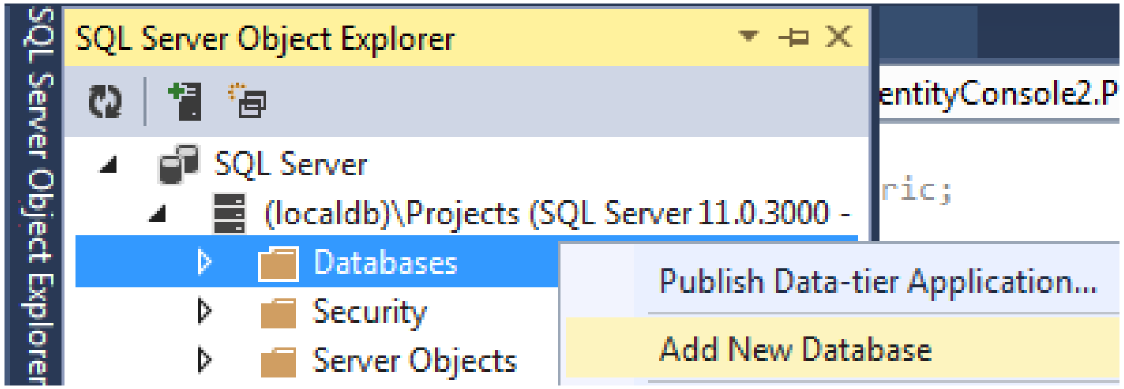
Solution name: IdentityConsoleApp

☑ Create directory for solution

☑ Create new Git repository

ASP.NET Identity

Open the SQL Server Object Explorer and create a new database to your localdb


SQL Server Object Explorer
SQL Server Object Explorer
SQL Server
(localdb)\Projects (SQL Server 11.0.3000 -
Databases
Security
Server Objects
Publish Data-tier Application...
Add New Database
entityConsole2.P
ric;

ASP.NET Identity

Then we add the connectionstring to our app.config file.

```
<connectionStrings>
    <add name="DefaultConnection"
         connectionString=“Data Source=(localdb)\Projects;Initial
         Catalog=IdentityConsole;Integrated Security=True;
         Connect Timeout=30;Encrypt=False;TrustServerCertificate=False"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Please note that the connection string has to be on a single line!

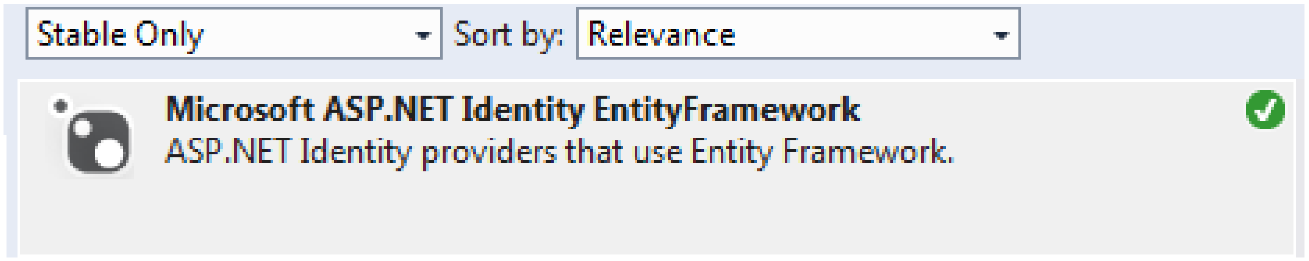
ASP.NET Identity

We will have to add ASP.NET Identity

Open the Package Manager Console and type:

```
PM> Install-Package Microsoft.AspNet.Identity.EntityFramework
```

Or open the NuGet package manager by right clicking your project and choose Manage NuGet Packages


Stable Only | Sort by: Relevance

Microsoft ASP.NET Identity EntityFramework
ASP.NET Identity providers that use Entity Framework.

ASP.NET Identity

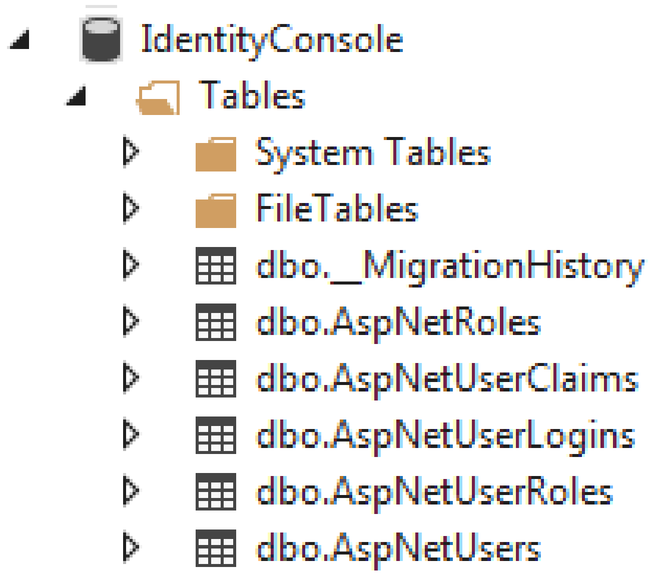
Now we initialize the database by making an instance of **IdentityDbContext**, and calling the Initialize method

Bool value which determines:
If true = run always
If false = run only if context has not been run before

```
static void Main(string[] args)
{
    new IdentityDbContext<IdentityUser>().Database.Initialize(true);
}
```

ASP.NET Identity

When we run the application the **IdentityDbContext** will create tables in our database

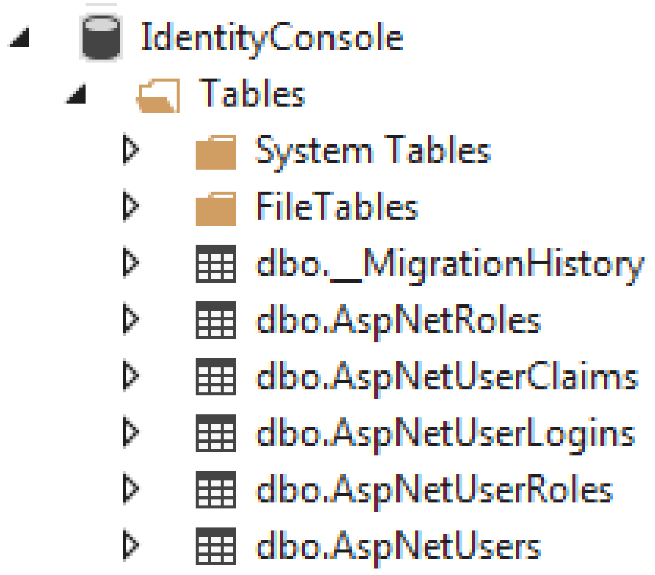


```
IdentityConsole
  Tables
    System Tables
    FileTables
    dbo.__MigrationHistory
    dbo.AspNetRoles
    dbo.AspNetUserClaims
    dbo.AspNetUserLogins
    dbo.AspNetUserRoles
    dbo.AspNetUsers
```

ASP.NET Identity

- dbo.AspNetUsers contains all the users
- dbo.AspNetRoles contains all the roles
- dbo.AspNetUserRoles contains the relations between users and roles
- dbo.AspNetUserClaims contains all the claims which belong to a specific user
- dbo.AspNetUserLogins contains all the users who login using an external login
- dbo.MigrationHistory contains all the performed database migrations

?? Höra med tore

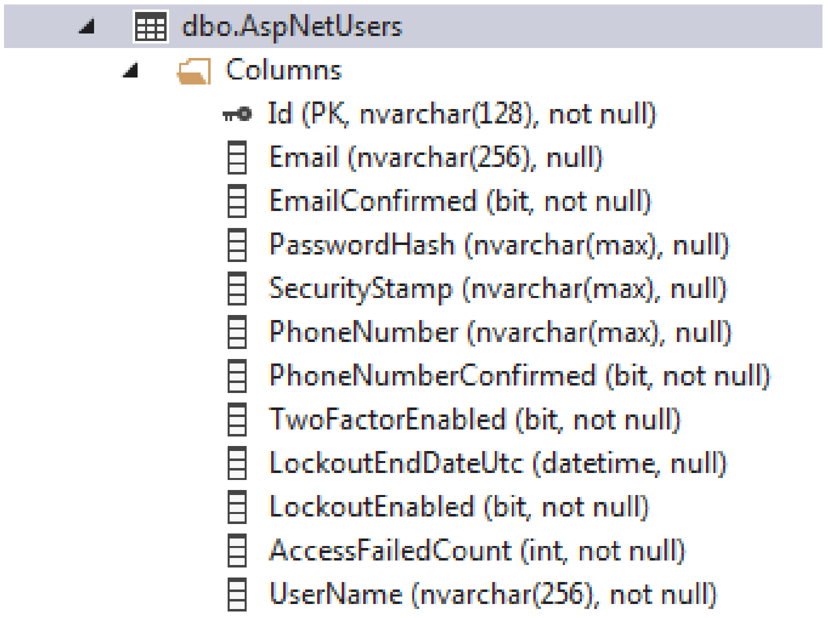


```
IdentityConsole
  Tables
    System Tables
    FileTables
    dbo.__MigrationHistory
    dbo.AspNetRoles
    dbo.AspNetUserClaims
    dbo.AspNetUserLogins
    dbo.AspNetUserRoles
    dbo.AspNetUsers
```

ASP.NET Identity

The table AspNetUsers contains several columns to handle a user

?? Fråga Tore



```
dbo.AspNetUsers
  Columns
    Id (PK, nvarchar(128), 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)
```

ASP.NET Identity

We will now add logic to seed the database, which we do by creating a class that we will call **ApplicationDbInitializer** which

- inherits from the generic **DropCreateDatabaseAlways** class
- will contain the seed method.

```
public class ApplicationDbInitializer :
DropCreateDatabaseAlways<IdentityDbContext<IdentityUser>>
```

ASP.NET Identity

We override the seed method and use the class Usermanager to create 100 new users in the database

```
protected override void Seed(IdentityDbContext<IdentityUser> context)
{
    var usermanager = new UserManager<IdentityUser>(new UserStore<IdentityUser>(new
     IdentityDbContext<IdentityUser>()));

    for (int j = 0; j < 10; j++)
    {
        for (int i = 0; i < 10; i++)
        {
            var email = "user" + ((j * 100) + i) + "@example.com";
            string phone = "12345678" + (j * 100 + i);
            var tempuser = new IdentityUser(){UserName=email,Email=email,PhoneNumber=phone};
            usermanager.Create(tempuser, "ASP+Rocks4U");
        }
        var appstore = new UserStore<IdentityUser>();
        appstore.Context.SaveChanges();
    }
    usermanager.Dispose();
    base.Seed(context);
}
```

ASP.NET Identity

In the main method we set the initializer to initialize the **ApplicationDbInitializer**, which will run the seed method

```
static void Main(string[] args)
{
    Database.SetInitializer(new ApplicationDbInitializer());
    new IdentityDbContext< IdentityUser>().Database.Initialize(true);
}
```

ASP.NET Identity

If we run the application now, the application will populate the database with 100 users

It is easy to customize your user identity, by creating a class which inherits from IdentityUser and adding properties

Summary

In this module we have:

- Created a database that uses ASP.NET Identity structure
- Created a seed method to populate the database every time you run the application