ASP.NET MVC - AngularJS, Web API and EntityFramework to build SPA

IntroductionThis article walks you through the steps for create a web application using AngularJS, that uses WebApi to communicate between client and server side.STEP 1 - Create ASP.NET Web Api ApplicationCheck the link below, to see all the steps to create a Web Api wtih Entity

https://code.msdn.microsoft.com/ASPNET-MVC-AngularJS-Web-e071f83d

Introduction

This article walks you through the steps for create a web application using AngularJS, that uses WebApi to communicate between client and server side.

STEP 1 - Create ASP.NET Web Api Application

Check the link below, to see all the steps to create a Web Api wtih Entity Framework code first implementation.

http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx

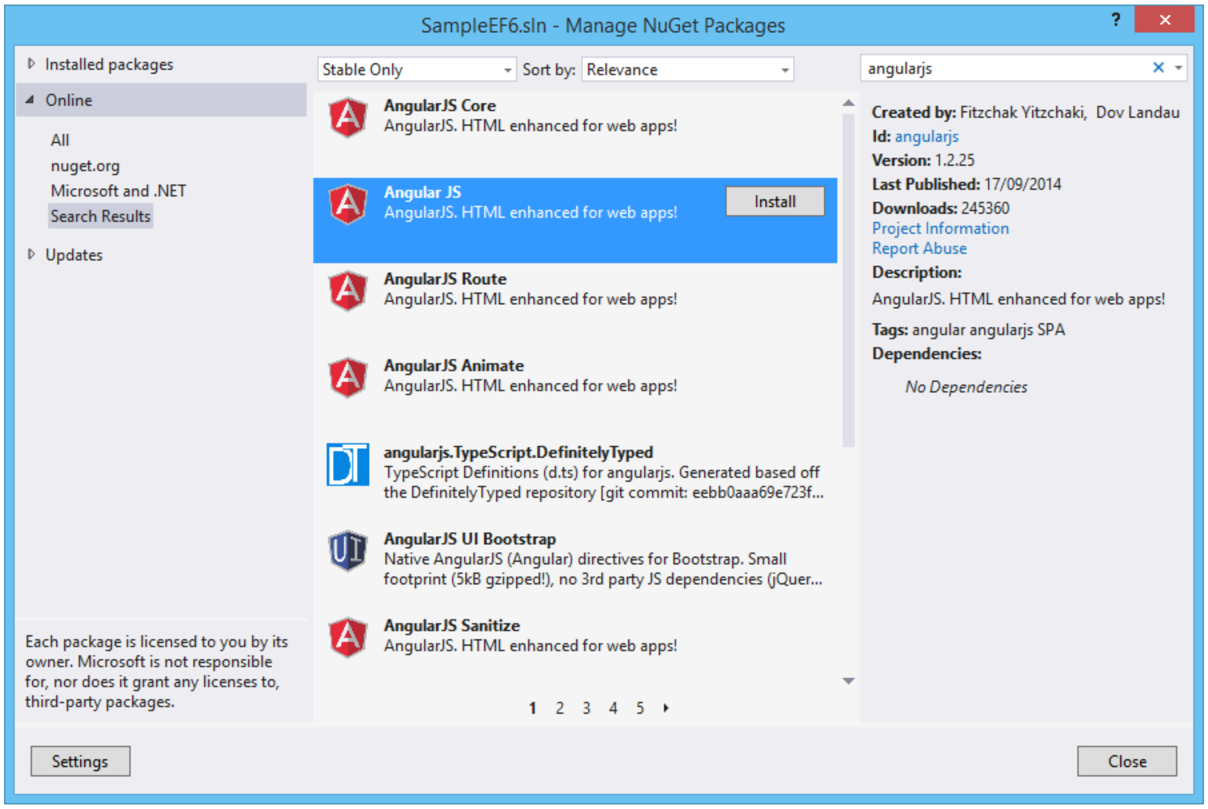
STEP 2 - Install Nuget

Now in order to use Entity Framework we need to install a Nuget package.

So on the Visual Studio 2013, select the follow menu option:

Tools-> Library Package manager -> Manage NuGet Packages for Solution

Search for AngularJs and select the option Install.


SampleEF6.sln - Manage NuGet Packages

Installed packages
Online
All
nuget.org
Microsoft and .NET
Search Results
Updates

Stable Only | Sort by: Relevance | angularjs

AngularJS Core
AngularJS. HTML enhanced for web apps!

Angular JS
AngularJS. HTML enhanced for web apps!

AngularJS Route
AngularJS. HTML enhanced for web apps!

AngularJS Animate
AngularJS. HTML enhanced for web apps!

angularjs.TypeScript.DefinitelyTyped
TypeScript Definitions (d.ts) for angularjs. Generated based off the DefinitelyTyped repository [git commit: eebb0aaa69e723f...

AngularJS UI Bootstrap
Native AngularJS (Angular) directives for Bootstrap. Small footprint (5kB gzipped!), no 3rd party JS dependencies (jQuer...

AngularJS Sanitize
AngularJS. HTML enhanced for web apps!

Created by: Fitzchak Yitzchaki, Dov Landau
Id: angularjs
Version: 1.2.25
Last Published: 17/09/2014
Downloads: 245360
Project Information
Report Abuse
Description:
AngularJS. HTML enhanced for web apps!
Tags: angular angularjs SPA
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

This option, will install automatically the Nuget Package.

STEP 3 - Create Javascript controller

Now add a new javascript file (contactController.js) in scripts directory and add angular functions.

Copy code

JavaScript

```
function contactController($scope, $http) {

    $scope.loading = true;

    $scope.addMode = false;



    //Used to display the data

    $http.get('/api/Contact/').success(function (data) {

        $scope.Contacts = data;
        $scope.loading = false;

    })

    .error(function () {

        $scope.error = "An Error has occured while loading posts!";

        $scope.loading = false;

    });



    $scope.toggleEdit = function () {

        this.Contact.editMode = !this.Contact.editMode;

    };

    $scope.toggleAdd = function () {

        $scope.addMode = !$scope.addMode;

    };



    //Used to save a record after edit

    $scope.save = function () {

        alert("Edit");

        $scope.loading = true;

        var frien = this.Contact;

        alert(emp);

        $http.put('/api/Contact/', frien).success(function (data) {

            alert("Saved Successfully!!");

            emp.editMode = false;

            $scope.loading = false;
        }).error(function (data) {
            $scope.error = "An Error has occured while Saving Contact! " + da
ta;
            $scope.loading = false;

        });
    };

    //Used to add a new record
    $scope.add = function () {
        $scope.loading = true;
        $http.post('/api/Contact/', this.newContact).success(function (data)
{
            alert("Added Successfully!!");
            $scope.addMode = false;
            $scope.Contacts.push(data);
            $scope.loading = false;
        }).error(function (data) {
            $scope.error = "An Error has occured while Adding Contact! " + da
ta;
            $scope.loading = false;

        });
    };

    //Used to edit a record
    $scope.deleteContact = function () {

        $scope.loading = true;

        var Contactid = this.Contact.ContactId;

        $http.delete('/api/Contact/' + Contactid).success(function (data) {

            alert("Deleted Successfully!!");

            $.each($scope.Contacts, function (i) {

                if ($scope.Contacts[i].ContactId === Contactid) {

                    $scope.Contacts.splice(i, 1);

                    return false;

                }

            });

            $scope.loading = false;

        }).error(function (data) {

            $scope.error = "An Error has occured while Saving Contact! " + da
ta;

            $scope.loading = false;

        });

    };

}
```

STEP 4- Create View

On BundlesConfig file add these lines of code.

Copy code

C#

```
 bundles.Add(new ScriptBundle("~/bundles/angularjs").Include(

                     "~/Scripts/angular.js",

                     "~/Scripts/contactController.js"));
```

Open _Layout.cshtml page from Shared folder and add these two lines to render angularJS and contactController in application.

Copy code

JavaScript

```
@Scripts.Render("~/bundles/angularjs")
```

Now let's work on View.

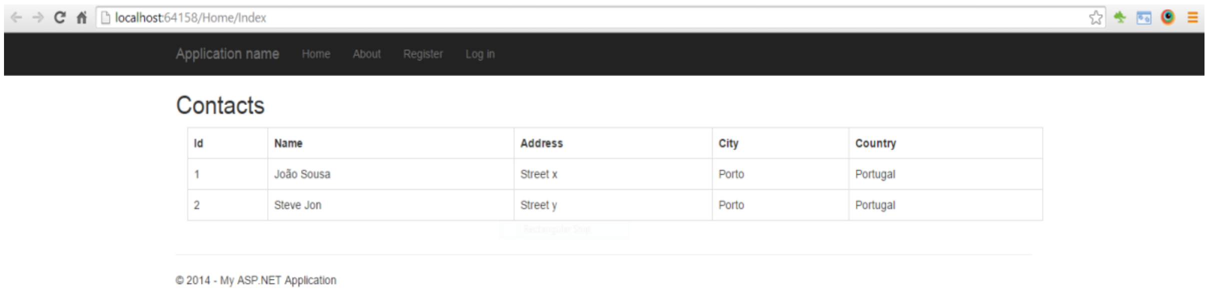
Copy code

HTML

```
@{

    Layout = "~/Views/Shared/_Layout.cshtml";

}



<h2>Contacts</h2>

<div ng-app="" ng-controller="contactController" class="container">

    <strong class="error">{{ error }}</strong>

    <div class="row">

        <div class="logs-table col-xs-12">

            <table class="table table-bordered table-
hover" style="width:100%">

                <tr>

                    <th>Id</th>

                    <th>Name</th>
                    <th>Address</th>

                    <th>City</th>

                    <th>Country</th>

                </tr>

                <tr ng-repeat="contact in contacts">

                    <td>{{ contact.Id }}</td>

                    <td>{{ contact.Name }}</td>

                    <td>{{ contact.Address }}</td>

                    <td>{{ contact.City }}</td>

                    <td>{{ contact.Country }}</td>

                </tr>

            </table>

        </div>

    </div>

</div>
```

STEP 5 - Run application

Run application to see output:


localhost:64158/Home/Index

Application name Home About Register Log in

Contacts

Id	Name	Address	City	Country
1	João Sousa	Street x	Porto	Portugal
2	Steve Jon	Street y	Porto	Portugal

© 2014 - My ASP.NET Application

Resources

Some good resources about Windows Azure could be found here:

- My personal blog: http://joaoeduardosousa.wordpress.com/
- Angular UI: http://angular-ui.github.io/

ASP.NET WebAPI - Entity Framework Code First

Table of Contents

Introduction

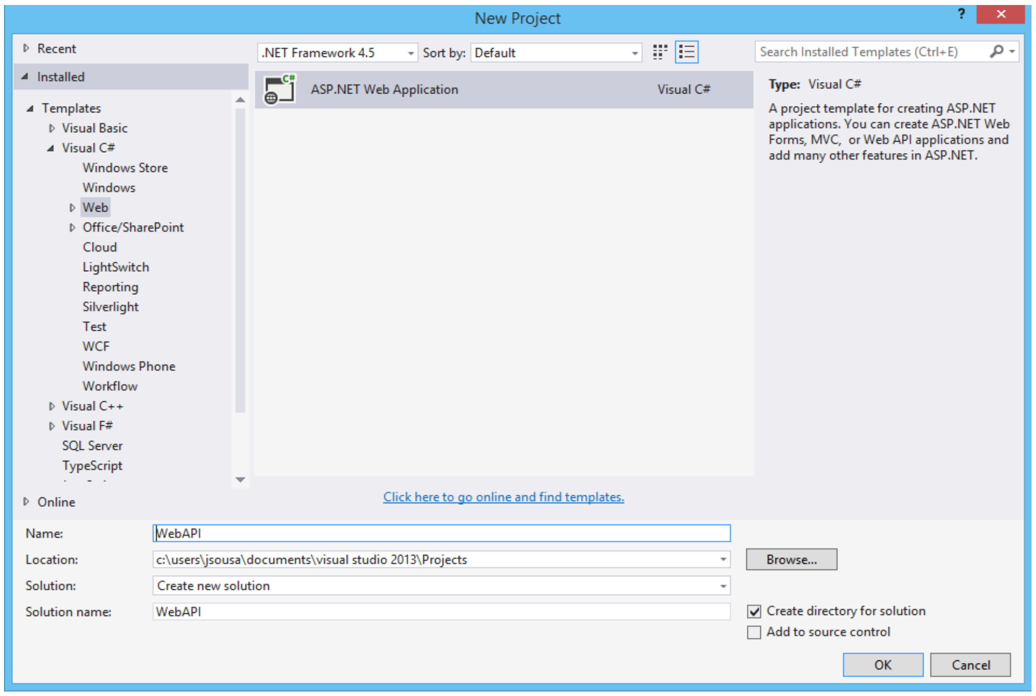
This article walks you through configuration of Entity framework 6.1 on a WebAPi project.

For that we will create a model "Contacts", where we can save data from contact persons.

STEP 1 - Create ASP.NET WebAPI 2 Application

I will be using Visual Studio 2013 as my development environment. Our first step will be to create an ASP.NET Web Application project based on the Web API template:

- Open Visual Studio 2013 and create a new project of type ASP.NET Web Application.
- On this project I create a solution called WebAPI.

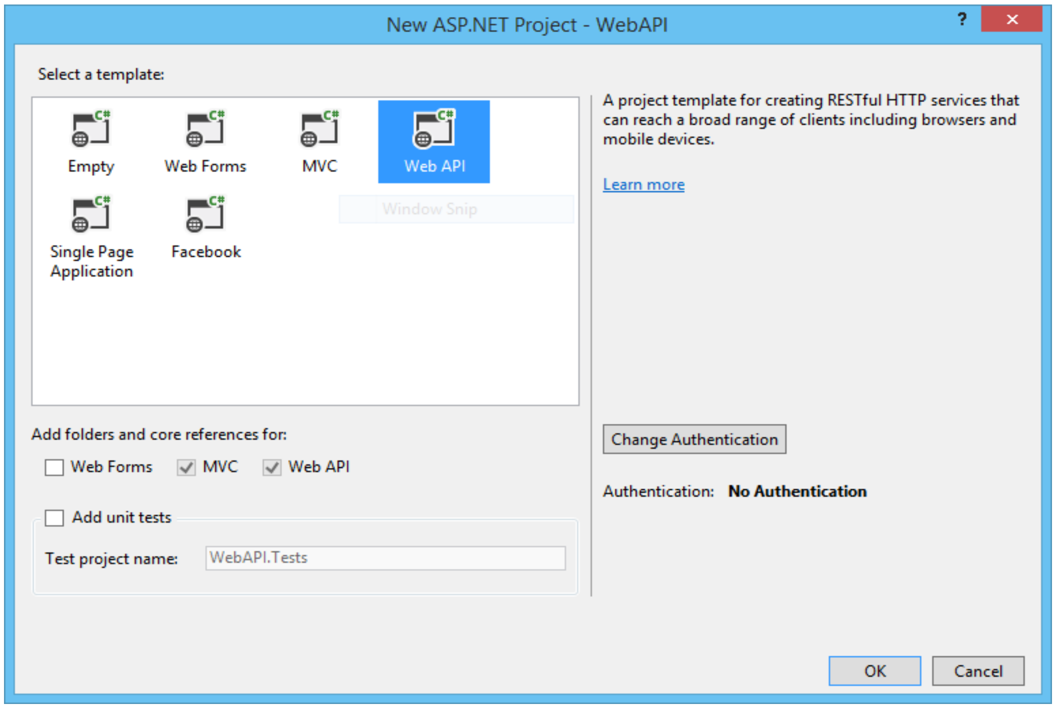


Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Name: WebAPI
Location: c:\users\jsousa\documents\visual studio 2013\Projects
Solution: Create new solution
Solution name: WebAPI
[✓] Create directory for solution
[ ] Add to source control

- Press OK, and a new screen will appear, with several options of template to use on our project.
- Select the option WebAPI.



A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Add folders and core references for:
[ ] Web Forms [✓] MVC [✓] Web API

[ ] Add unit tests

Test project name: WebAPI.Tests

Authentication: **No Authentication**

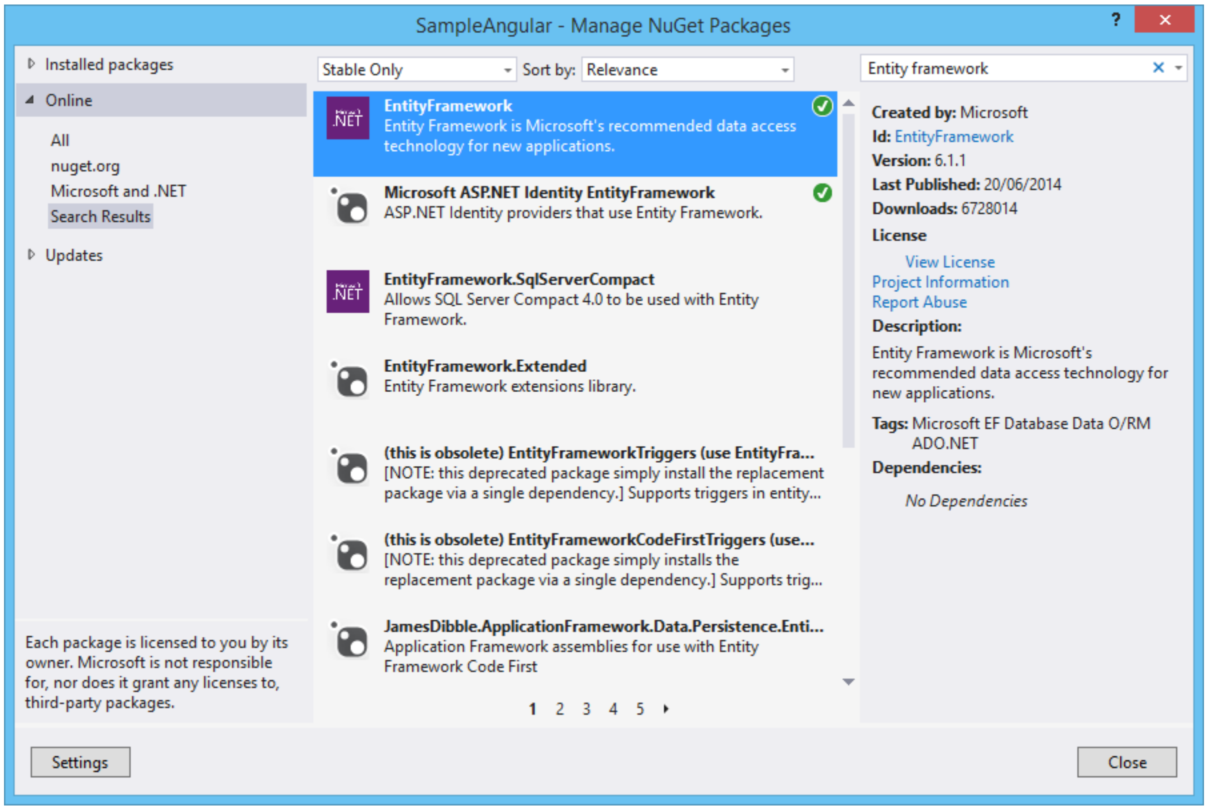
- The solution will be created.

STEP 2 - Install Nuget

Now in order to use Entity Framework we need to install a Nuget package.

So on the Visual Studio 2013, select the follow menu option:

- Tools-> Library Package manager -> Manage NuGet Packages for Solution
- Search for Entity Framework and select the option Install.



EntityFramework
Entity Framework is Microsoft's recommended data access technology for new applications.

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

EntityFramework.SqlServerCompact
Allows SQL Server Compact 4.0 to be used with Entity Framework.

EntityFramework.Extended
Entity Framework extensions library.

(this is obsolete) EntityFrameworkTriggers (use EntityFra...
[NOTE: this deprecated package simply install the replacement package via a single dependency.] Supports triggers in entity...

(this is obsolete) EntityFrameworkCodeFirstTriggers (use...
[NOTE: this deprecated package simply installs the replacement package via a single dependency.] Supports trig...

JamesDibble.ApplicationFramework.Data.Persistence.Enti...
Application Framework assemblies for use with Entity Framework Code First

Created by: Microsoft
Id: EntityFramework
Version: 6.1.1
Last Published: 20/06/2014
Downloads: 6728014
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

This option, will install automatically the Nuget Package.

STEP 3 - Create Data Model

After we have our web application created, we need to create our data model.

For that, we can create a new class Contact with the follow code:

C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace SampleEF6.Models
{
    public class Contact
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string City { get; set; }
        public string Country { get; set; }
    }
}
```

STEP 4 - Configure Data Context

The context must receive on the constructor the name of connection string, so we name it "DefaultConnection", and we will show on the next steps how to configure it.

On the context, we need to define the DBSet, that represent each model created on the below step. On this example we only have one model so we will define it.

C#

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace SampleEF6.Models
{
    public class EFContext : DbContext
    {
        public EFContext()
            : base("name=DefaultConnection")
        {
            base.Configuration.ProxyCreationEnabled = false;
        }

        public DbSet<Contact> Contacts { get; set; }
    }
}
```

Set some initial data. For that we can use the Seed method. This method allows us to insert initial data to every data model created on context.

C#

```
namespace SampleEF6.Migrations
{
    using SampleEF6.Models;
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration : DbMigrationsConfiguration<SampleEF6.Models.E
FContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;
            ContextKey = "SampleEF6.Models.EFContext";
        }

        protected override void Seed(SampleEF6.Models.EFContext context)
        {
            context.Contacts.AddOrUpdate(
              p => p.Name,
              new Contact { Name = "João Sousa", Address= "Street x", City = "Porto",
Country = "Portugal" },
              new Contact { Name = "Steve Jon", Address = "Street y", City = "Porto",
Country = "Portugal" },
              new Contact { Name = "Peter", Address = "Street z", City = "Porto", Coun
try = "Portugal" }
            );
        }
    }
}
```

STEP 5 - Create Contact controller

Contact controller CRUD methods:

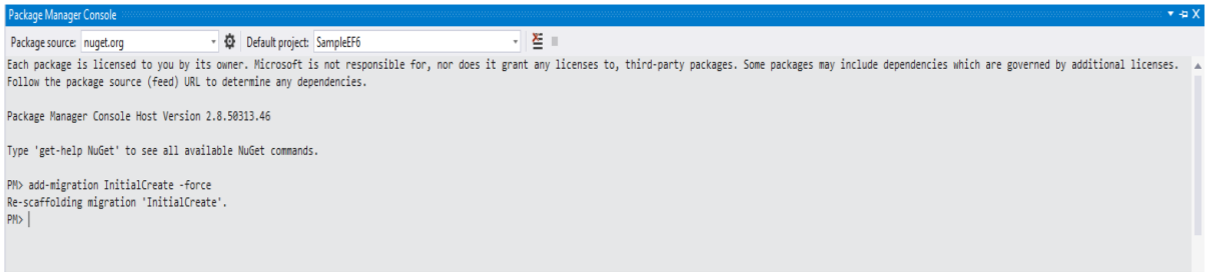
```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using SampleEF6.Models;

namespace SampleEF6.Controllers
{
    public class ContactController : ApiController
    {
        private EFContext db = new EFContext();

        // GET api/<controller>
        [HttpGet]

        public IEnumerable<Contact> Get()

        {

            return db.Contacts.AsEnumerable();

        }


        // GET api/<controller>/5

        public Contact Get(int id)

        {

            Contact Contact = db.Contacts.Find(id);

            if (Contact == null)

            {

                throw new
HttpResponseException(Request.CreateResponse(HttpStatusCode.NotFound));

            }



            return Contact;

        }


        // POST api/<controller>

        public HttpResponseMessage Post(Contact Contact)

        {

            if (ModelState.IsValid)

            {

                db.Contacts.Add(Contact);

                db.SaveChanges();


                HttpResponseMessage response =
Request.CreateResponse(HttpStatusCode.Created, Contact);
```

```
                response.Headers.Location = new Uri(Url.Link("DefaultApi", new
{ id = Contact.Id }));

                return response;

            }

            else

            {

                return Request.CreateErrorResponse(HttpStatusCode.BadRequest,
ModelState);

            }

        }


        // PUT api/<controller>/5

        public HttpResponseMessage Put(int id, Contact Contact)

        {

            if (!ModelState.IsValid)

            {

                return Request.CreateErrorResponse(HttpStatusCode.BadRequest,
ModelState);

            }


            if (id != Contact.Id)

            {

                return Request.CreateResponse(HttpStatusCode.BadRequest);

            }



            db.Entry(Contact).State = EntityState.Modified;



            try

            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException ex)
            {
                return Request.CreateErrorResponse(HttpStatusCode.NotFound,
ex);
            }

            return Request.CreateResponse(HttpStatusCode.OK);
        }

        // DELETE api/<controller>/5
        public HttpResponseMessage Delete(int id)
        {
            Contact Contact = db.Contacts.Find(id);
            if (Contact == null)
            {
                return Request.CreateResponse(HttpStatusCode.NotFound);
            }

            db.Contacts.Remove(Contact);

            try
            {
                db.SaveChanges();
            }
            catch (DbUpdateConcurrencyException ex)
            {
                    return Request.CreateErrorResponse(HttpStatusCode.NotFound, 
ex);
                }

                return Request.CreateResponse(HttpStatusCode.OK, Contact);
            }

        protected override void Dispose(bool disposing)
        {
                db.Dispose();
                base.Dispose(disposing);
        }
    }
}
```

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

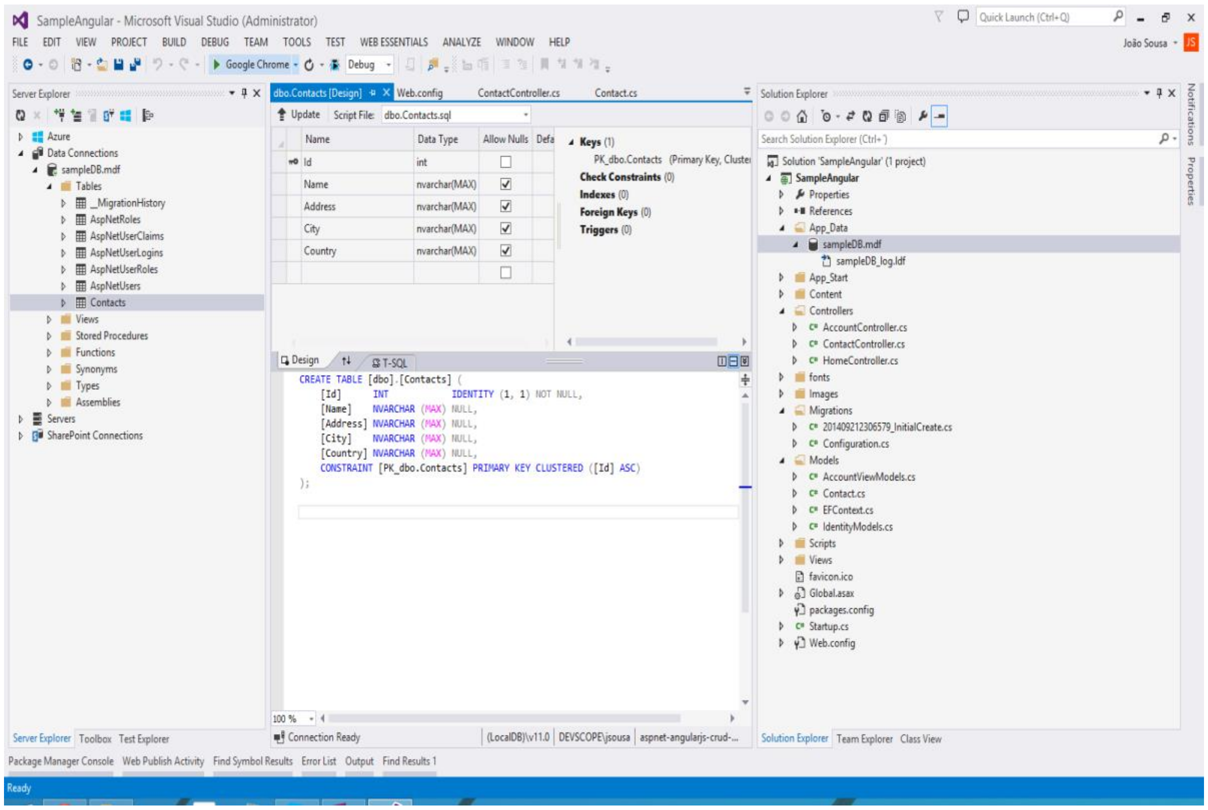
To create the database we need to run some command line instructions on the package manager console.

Enter the following instruction:



```
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages. Some packages may include dependencies which are governed by additional licenses.
Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 2.8.50313.46

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration InitialCreate -force
Re-scaffolding migration 'InitialCreate'.
PM> |
```

The database will be created with the data model Contacts as we see on the next image.

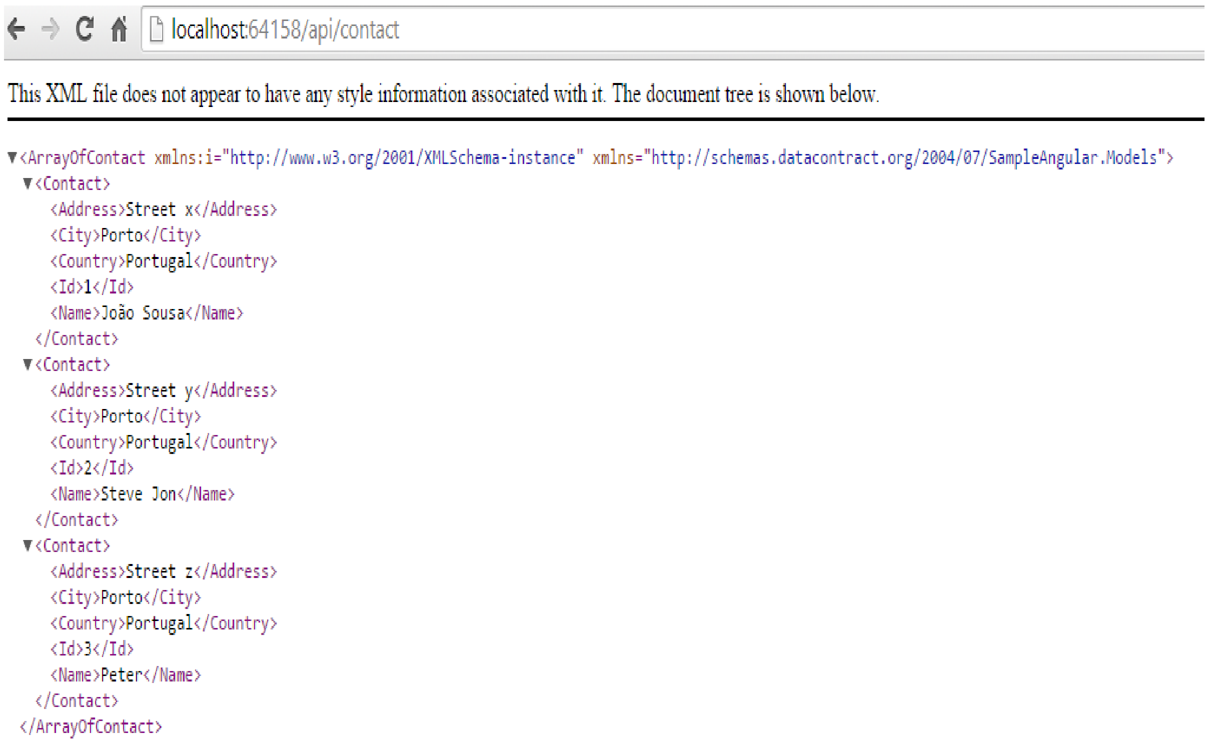


Name	Data Type	Allow Nulls	Defa
Id	int	☐	
Name	nvarchar(MAX)	✓	
Address	nvarchar(MAX)	✓	
City	nvarchar(MAX)	✓	
Country	nvarchar(MAX)	✓	
		☐	

```
CREATE TABLE [dbo].[Contacts] (
    [Id]      INT            IDENTITY (1, 1) NOT NULL,
    [Name]    NVARCHAR (MAX) NULL,
    [Address] NVARCHAR (MAX) NULL,
    [City]    NVARCHAR (MAX) NULL,
    [Country] NVARCHAR (MAX) NULL,
    CONSTRAINT [PK_dbo.Contacts] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

STEP 6 - Test WebApi

We can now call api/contact to receive every contact on database.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ArrayOfContact xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/SampleAngular.Models">
  ▼<Contact>
      <Address>Street x</Address>
      <City>Porto</City>
      <Country>Portugal</Country>
      <Id>1</Id>
      <Name>João Sousa</Name>
    </Contact>
  ▼<Contact>
      <Address>Street y</Address>
      <City>Porto</City>
      <Country>Portugal</Country>
      <Id>2</Id>
      <Name>Steve Jon</Name>
    </Contact>
  ▼<Contact>
      <Address>Street z</Address>
      <City>Porto</City>
      <Country>Portugal</Country>
      <Id>3</Id>
      <Name>Peter</Name>
    </Contact>
 </ArrayOfContact>
```

Resources

Some good resources about Windows Azure could be found here:

- My personal blog: http://joaoeduardosousa.wordpress.com/
- Download Code: http://code.msdn.microsoft.com/ASPNET-WebAPI-Entity-2f8797a1/view/Reviews