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>

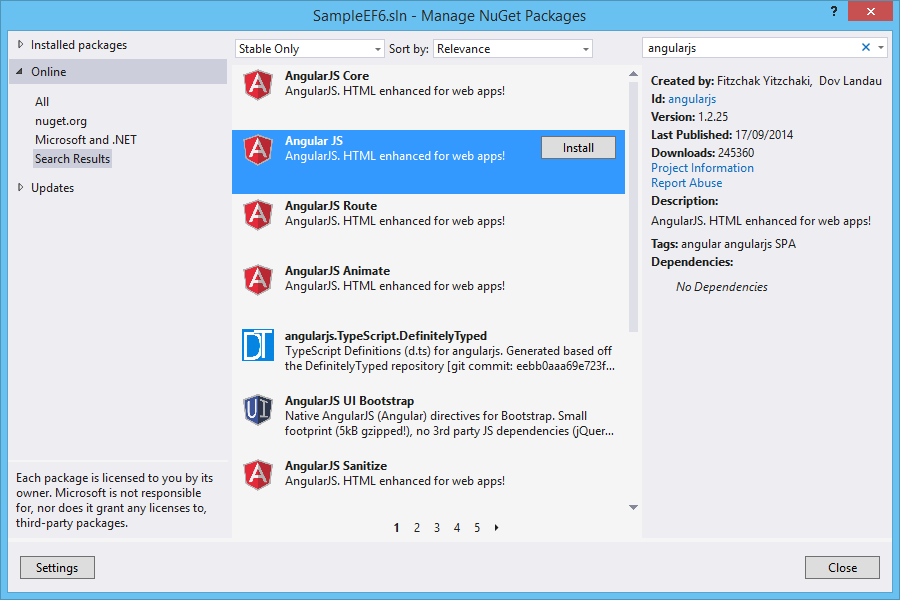
**STEP 2 - Install Nuget**

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.



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**](https://code.msdn.microsoft.com/ASPNET-MVC-AngularJS-Web-e071f83d/description)

# JavaScript

# **Editar Script|Remove**

# 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! " + data;**

# **$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! " + data;**

# **$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! " + data;**

# **$scope.loading = false;**

# }**);**

# }**;**

# }

**STEP 4- Create View**

On BundlesConfig file add these lines of code.

# [**Copy code**](https://code.msdn.microsoft.com/ASPNET-MVC-AngularJS-Web-e071f83d/description)

# C#

# **Editar Script|Remove**

# **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](https://code.msdn.microsoft.com/ASPNET-MVC-AngularJS-Web-e071f83d/description)

**JavaScript**

Editar Script|Remove

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

 Now let’s work on View.

[Copy code](https://code.msdn.microsoft.com/ASPNET-MVC-AngularJS-Web-e071f83d/description)

**HTML**

Editar Script|Remove

@{

    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>

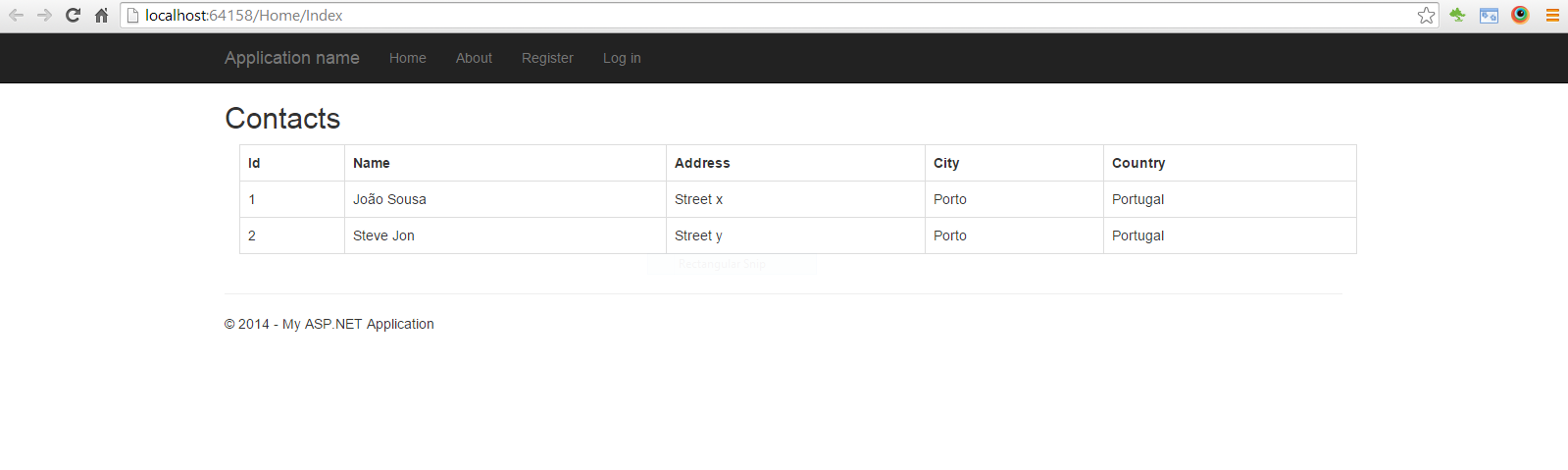
        </div>

    </div>

</div>

**STEP 5 - Run application**

Run application to see output:



**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](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#Introduction)
* [STEP 1 - Create ASP.NET WebAPI 2 Application](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_1_Create_ASP_NET_WebAPI_2_Application)
* [STEP 2 - Install Nuget](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_2_Install_Nuget)
* [STEP 3 - Create Data Model](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_3_Create_Data_Model)
* [STEP 4 - Configure Data Context](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_4_Configure_Data_Context)
* [STEP 5 - Create Contact controller](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_5_Create_Contact_controller)
* [STEP 6 - Test WebApi](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#STEP_6_Test_WebApi)
* [Resources](http://social.technet.microsoft.com/wiki/contents/articles/26795.asp-net-webapi-entity-framework-code-first.aspx#Resources)

# Introduction

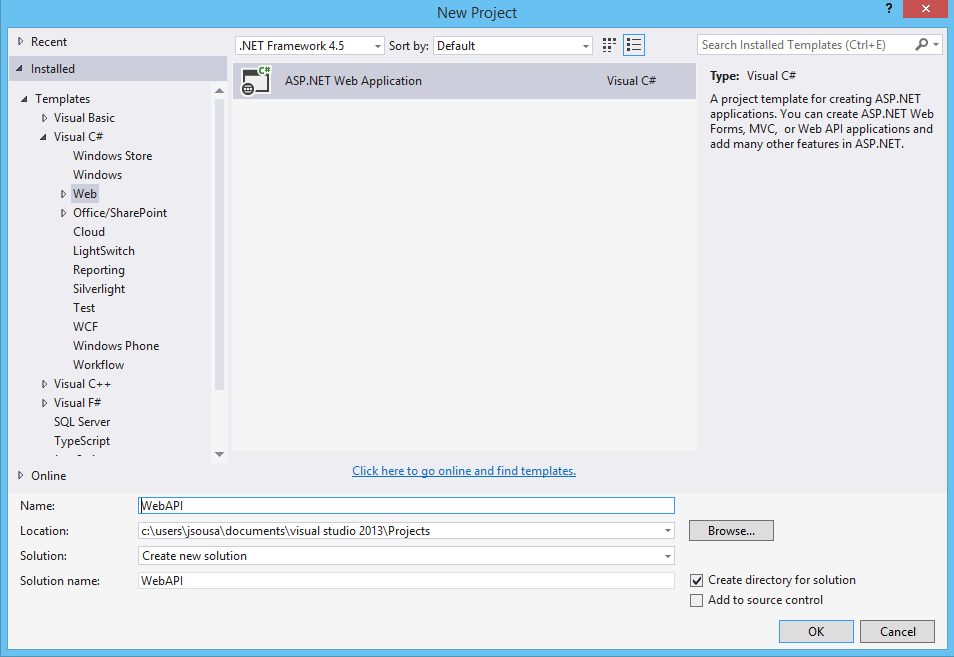
This article walks you through configuration of Entity framework 6.1 on a WebAPi project.

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

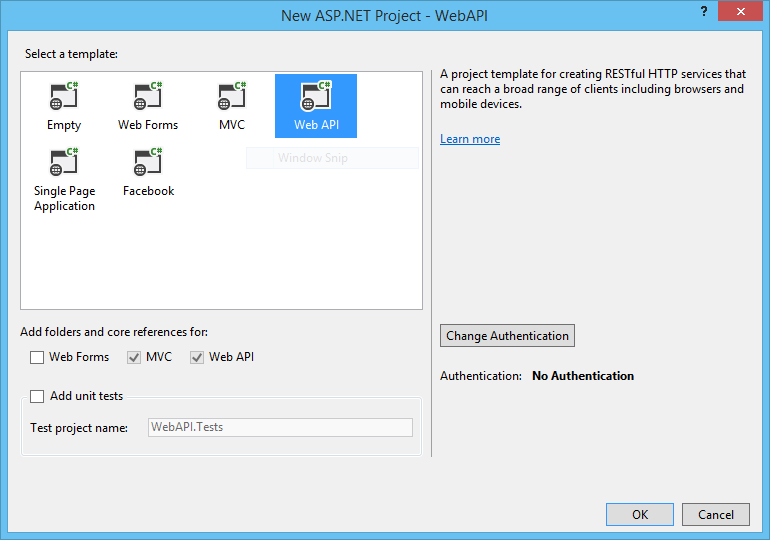
# STEP 1 - Create ASP.NET WebAPI 2 Application

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.



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



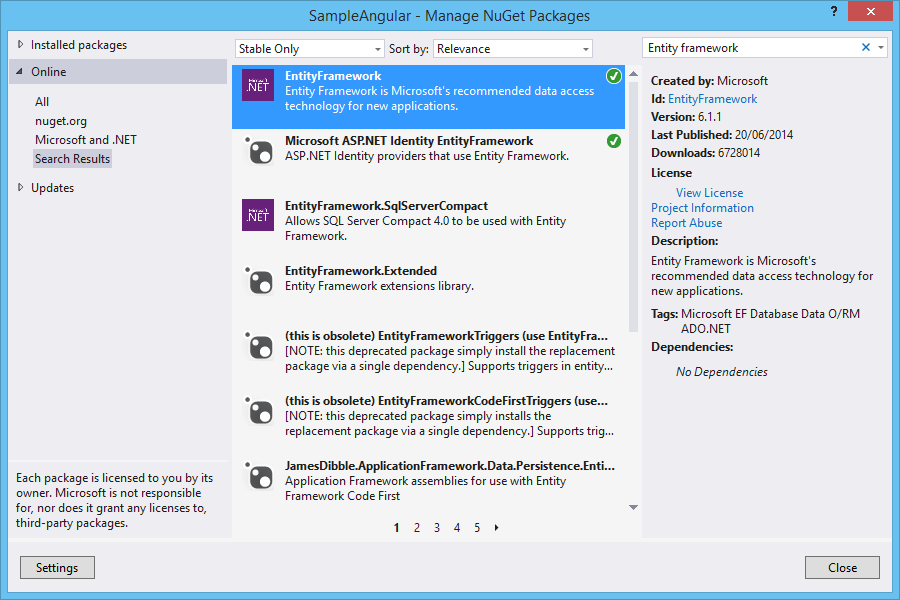
* The solution will be created.

# 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.



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.EFContext>

    {

**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", Country = "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);

        }

    }

}

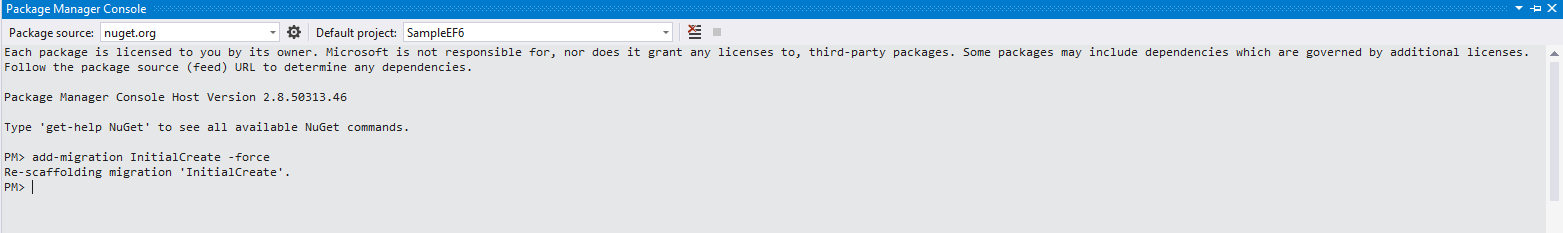
<**connectionStrings**>

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

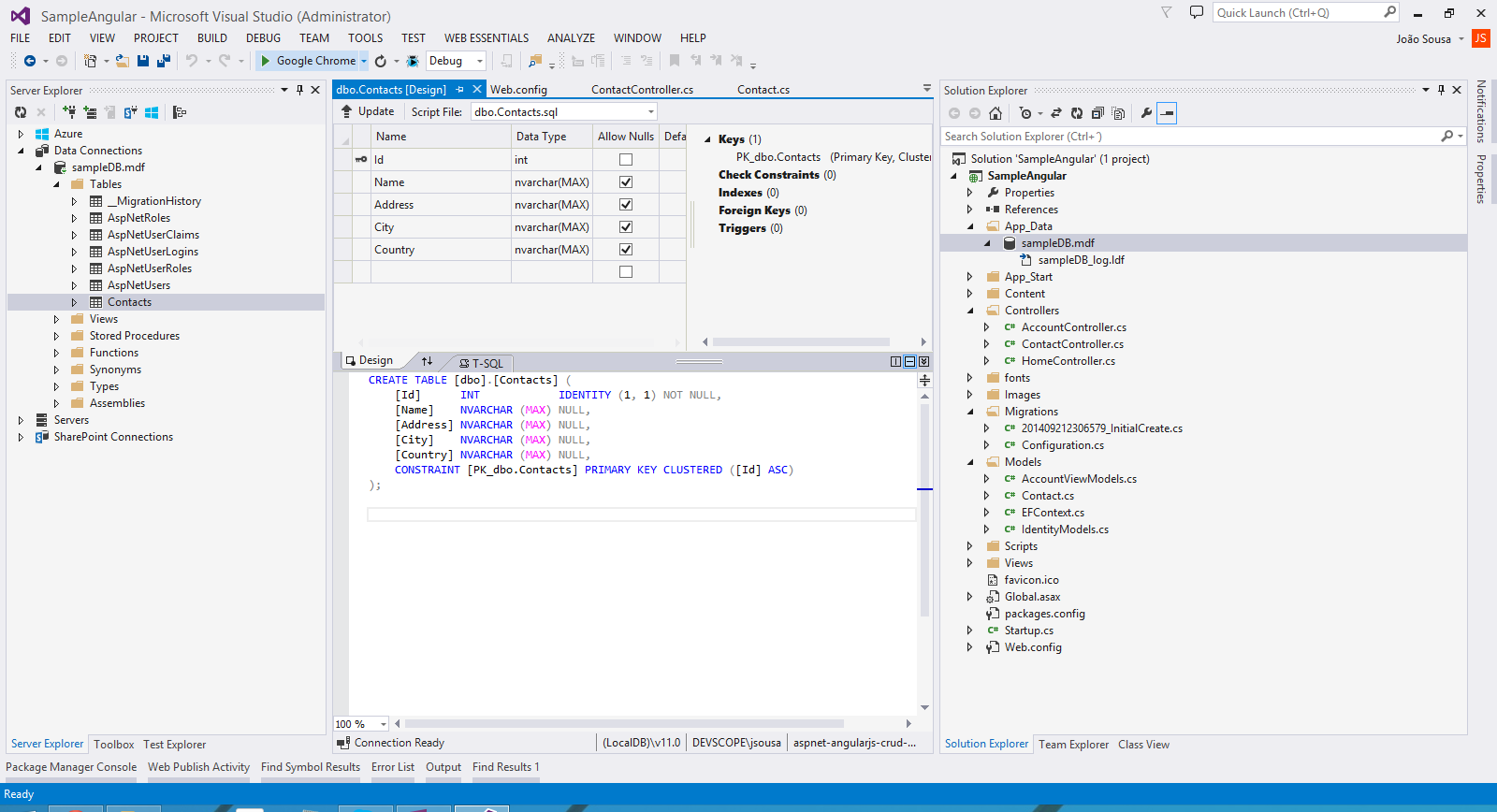
  </**connectionStrings**>

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

Enter the following instruction:



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



# 

# STEP 6 - Test WebApi

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



# 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>