Microsoft Azure

Step by step guide to setting up Azure SQL and accessing the new database from your web application

A step by step guide

Timengo
June 2015

Index

Introduction

This guide will show how to setup Azure SQL and how to access the new database from your web application.

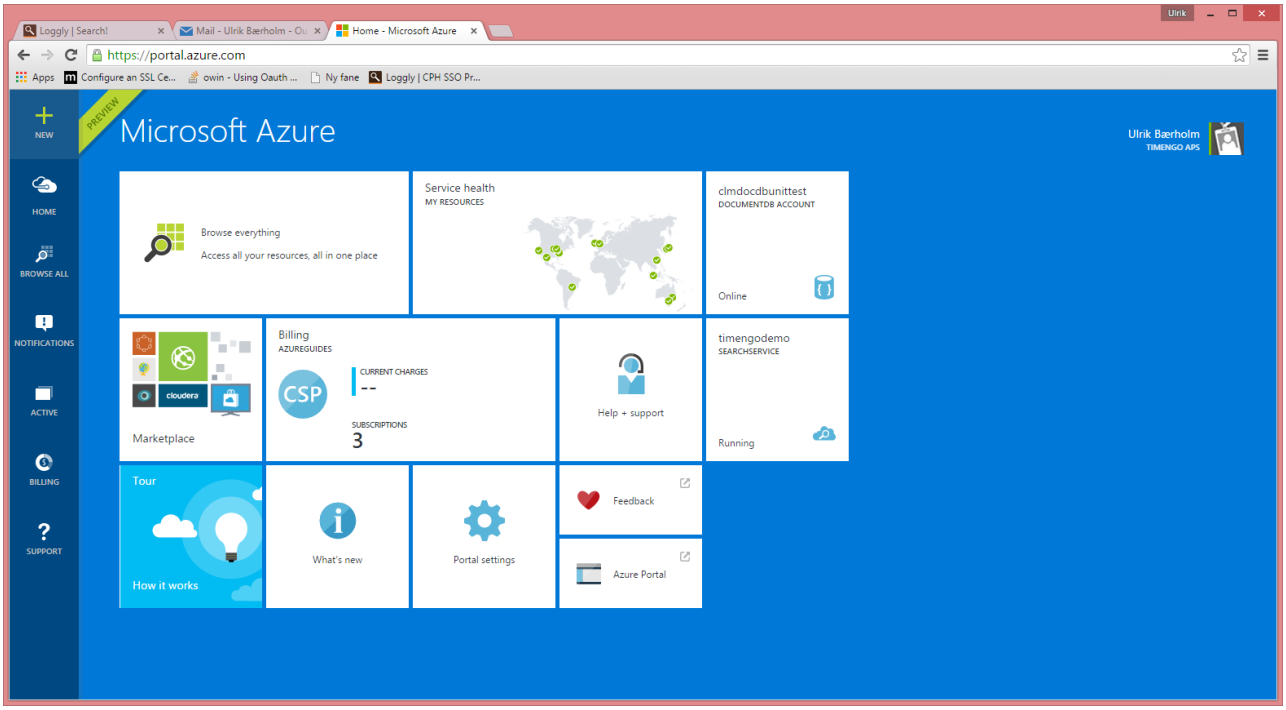
Prerequisite for completing this task is:

- An Azure Subscription
- Azure SDK
- Visual Studio 2013

Step by Step Guide:

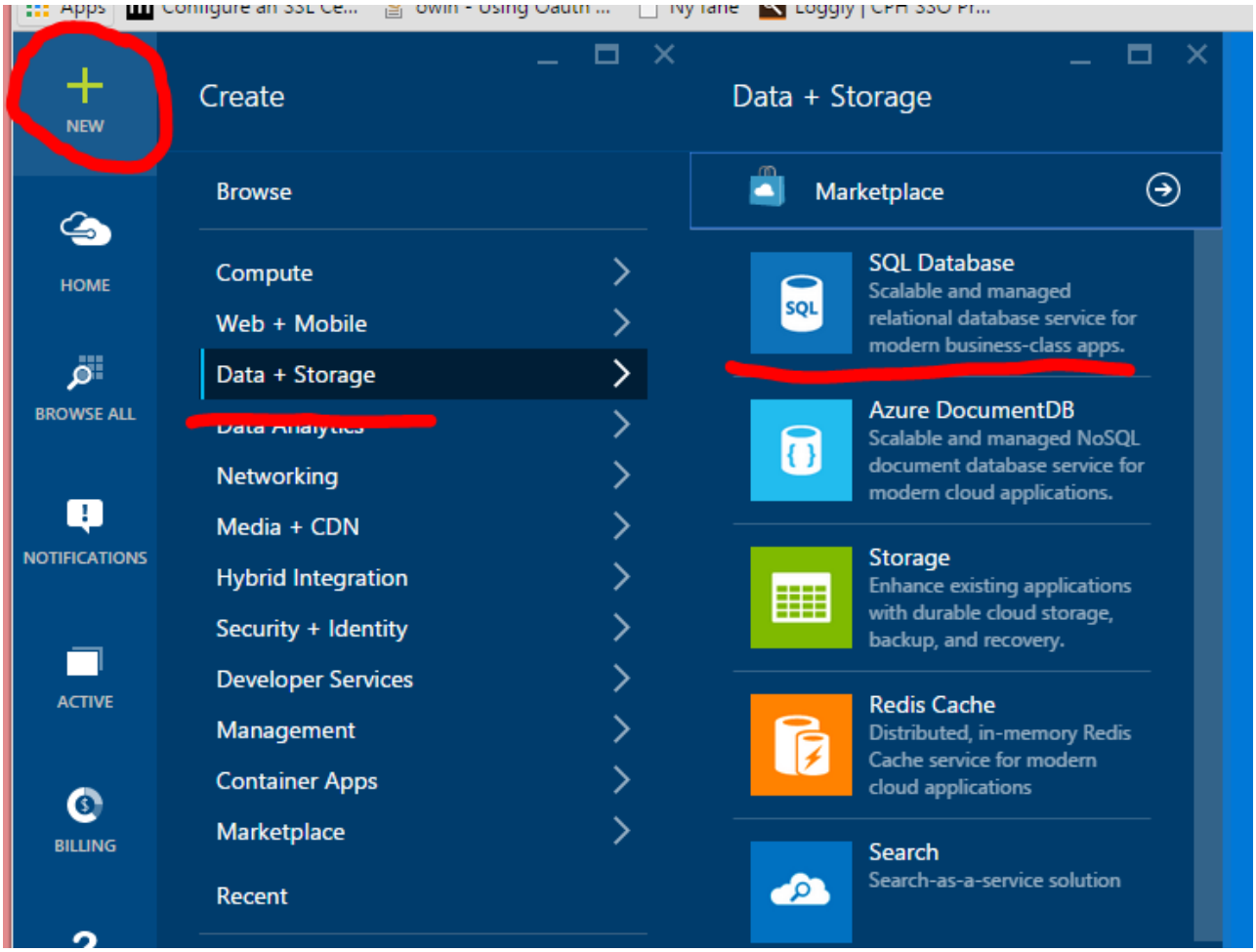
The following guide requires that you already have an Azure Subscription, if you don't have a subscription already you can signup for a free trial account here: https://azure.microsoft.com/en-us/pricing/free-trial/

For the rest of the guide we will be using the new Azure portal (currently in preview), to access the Azure portal navigate to: https://portal.azure.com/



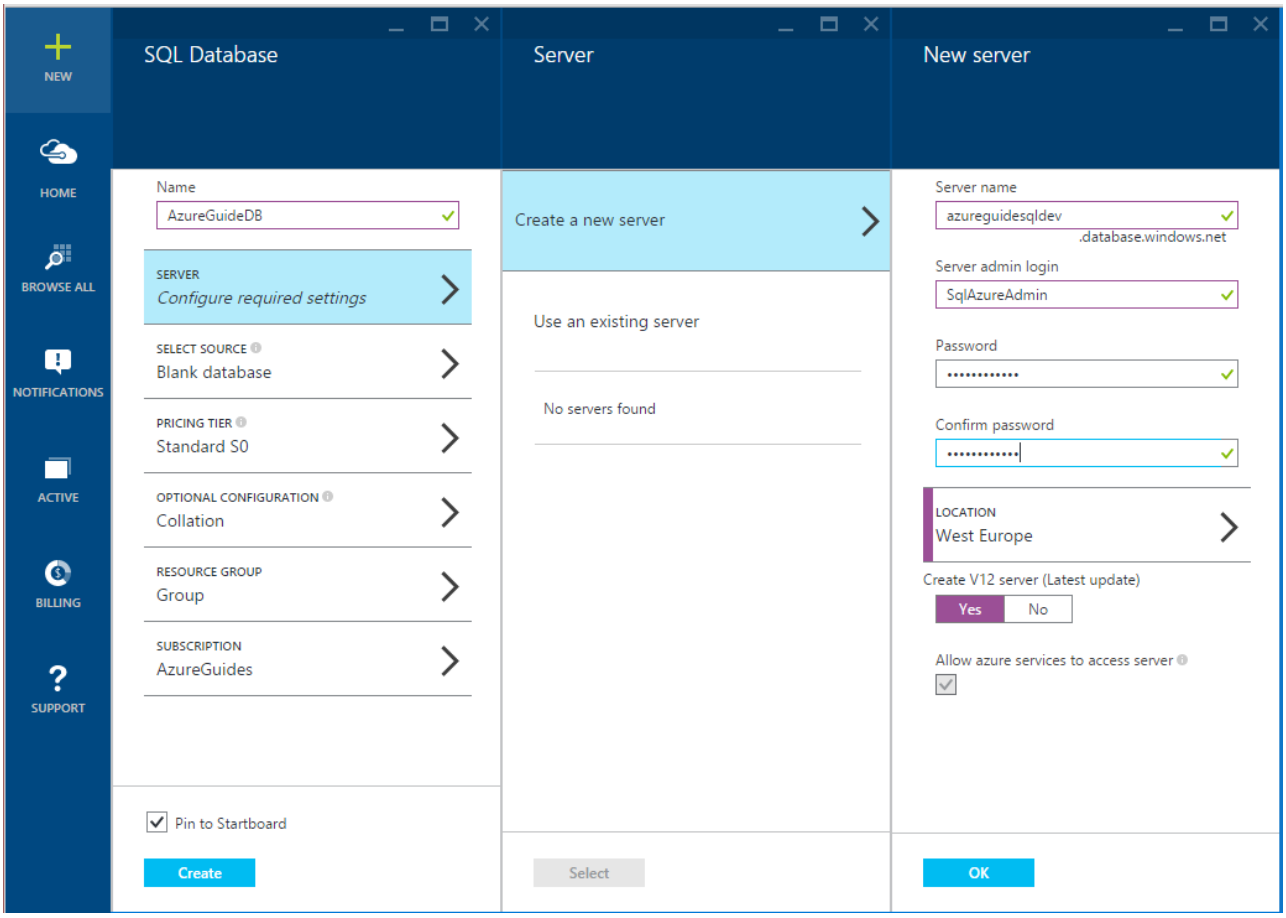
Step 1: Creating a new Azure SQL Database

To get us started we will create a new Azure SQL database; select "New" in the upper left hand corner in the menu, and then select "Data + Storage" and "SQL Database" in the sub menu:

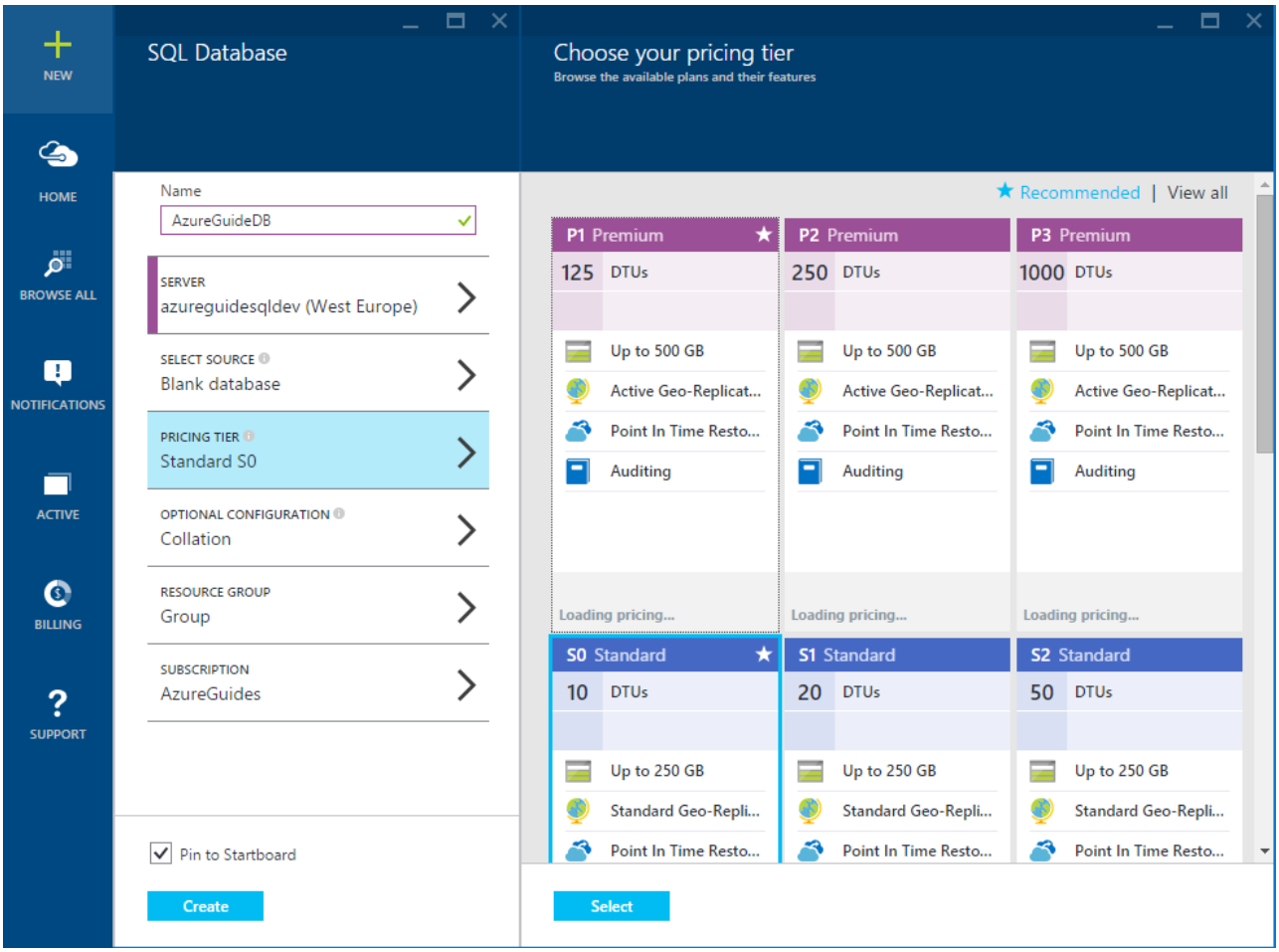


1. In the "SQL Database" panel enter a name for your new database, in this guide we will name the database "AzureGuideDB".
2. We now need to create a logical server for the database, click "Server", then "Create a new server", this opens the "New Server" panel.
3. In the Server panel, enter a name for your new database server. The "Server Name" is a unique name across all tenants on the azure platform, so is a good idea to use a naming convention to help you remember what each database server is used for.

Note: I always recommend using separate services for development, test and production, so In this guide, we will name the database "azureguidesqldev" (must be lower case).

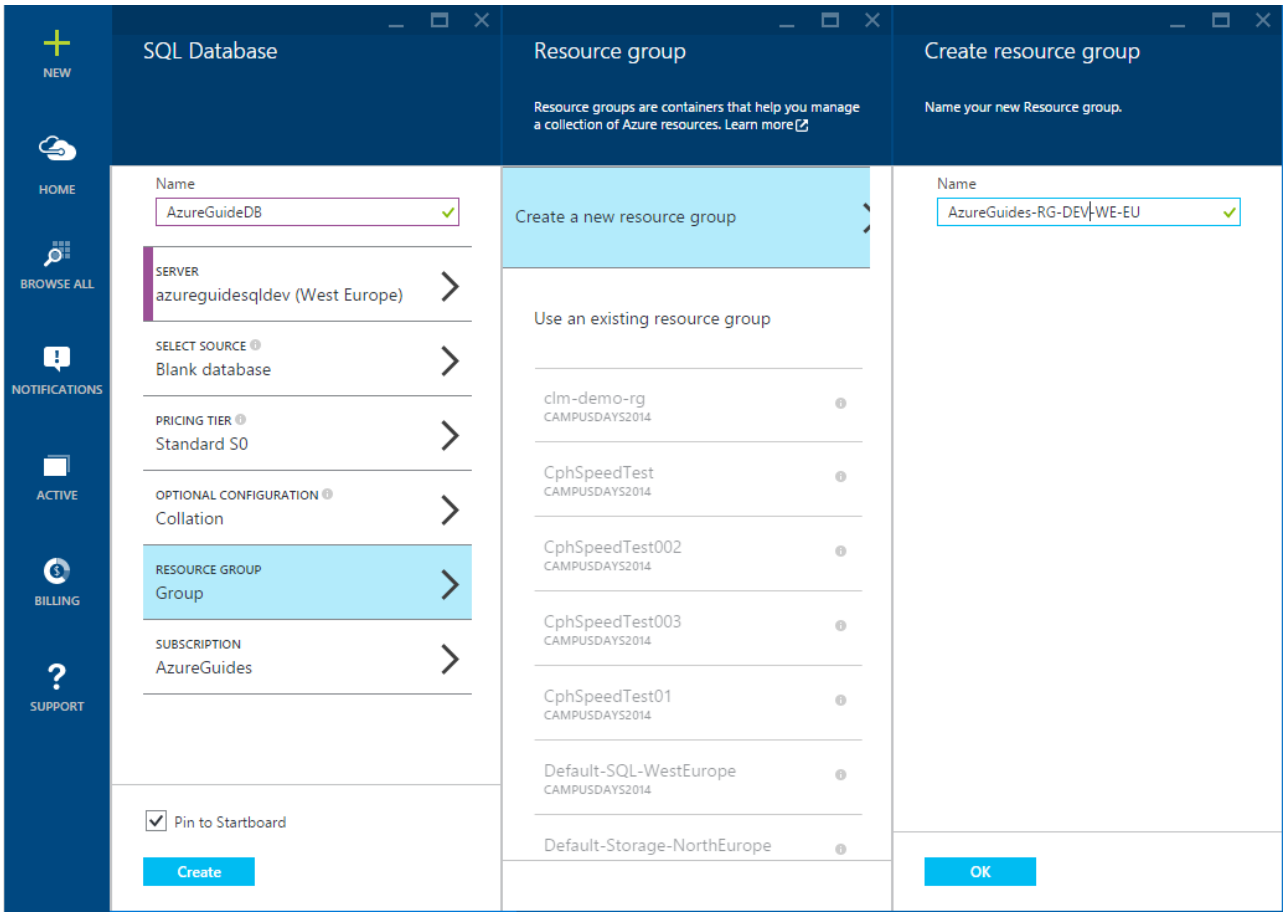


4. Enter the Server Admin Login as "SqlAzureAdmin" and set a Password and Confirm the Password. You will need the password later to access the database.
5. Finally select the preferred geographic Location for the database. You should always select the location that is closest to your primary users, and place the application in the same location.
6. Click "OK".



In this guide we will create an empty database, so we leave the default value for “Select source”.

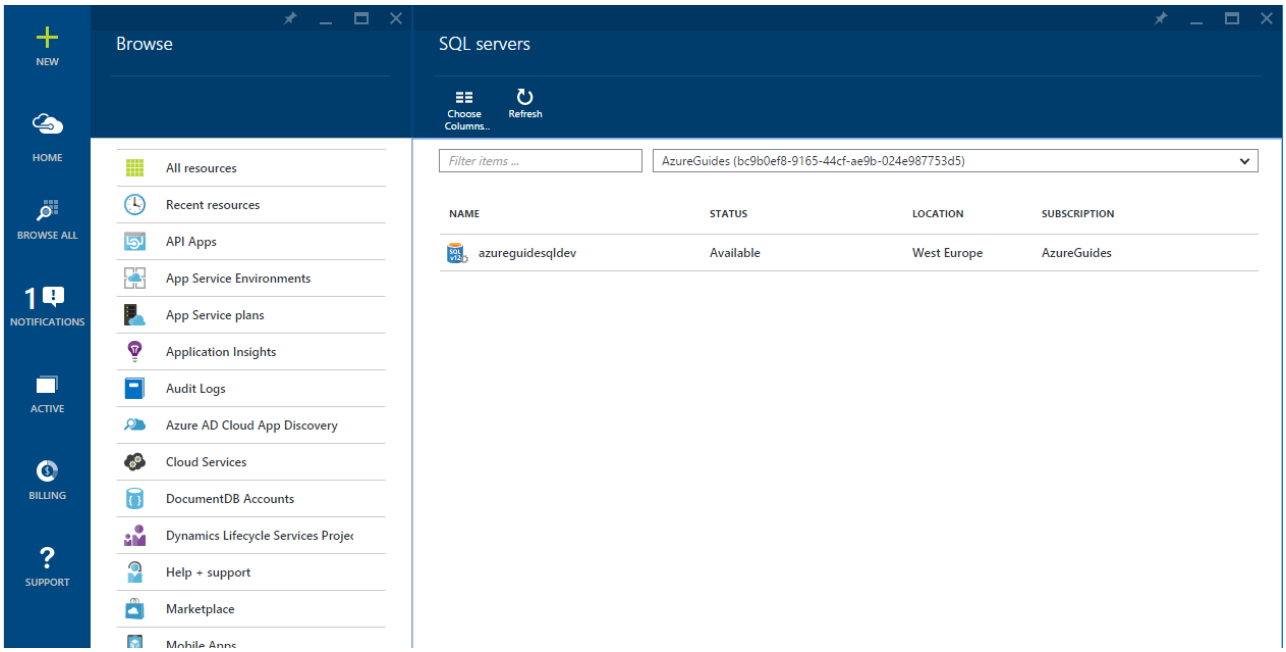
7. Click the “Pricing tier” option if you wish to change the default selected “Standard S0” the pricing tier controls the resource allocation to the new database server.



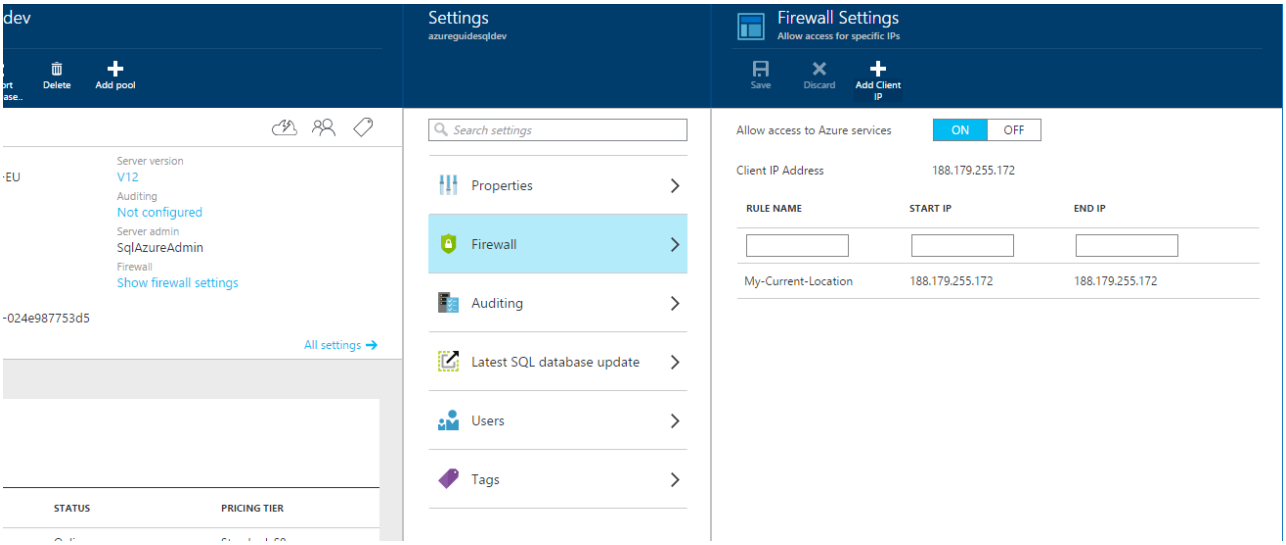
8. Click “Resource Group” in the menu, and then select “Create new resource group” and enter a name for the new resource group. In this guide we have named the resource group “AzureGuides-RG-DEV-WE-EU”. We recommend that you just like with the server names create your own naming convention to make management easier as the number of services you run on the Azure platform grow.
9. Click “Create” to provision the new server and database.

Note: Before you can access your new Azure SQL database you will need to configure he firewall and specify which IP addresses are allowed to access your database.

10. In the menu on the left hand side of the screen, click "Browse All" and then select "SQL Servers".
11. A list of the available options is then displayed; click the SQL server that you just created.



12. To configure the firewall settings click "Settings", then select "Firewall".



13. By default “Allow access to Azure Services” is ON, so you do not need to add any IP addresses for your application when it’s running on Azure.
14. If you need to access the database from jour local development environment, you must first add your current public IP address.
15. The portal displays you current public IP address, and you can add your current location by clicking “Add Client ID” and save the default settings – this will add your current IP address to the list off allowed IP addresses for the SQL server.
16. If you do not have a fixed public IP address, you might need to add additional IP addresses later, if your Internet Service Provider assign you a new IP address.

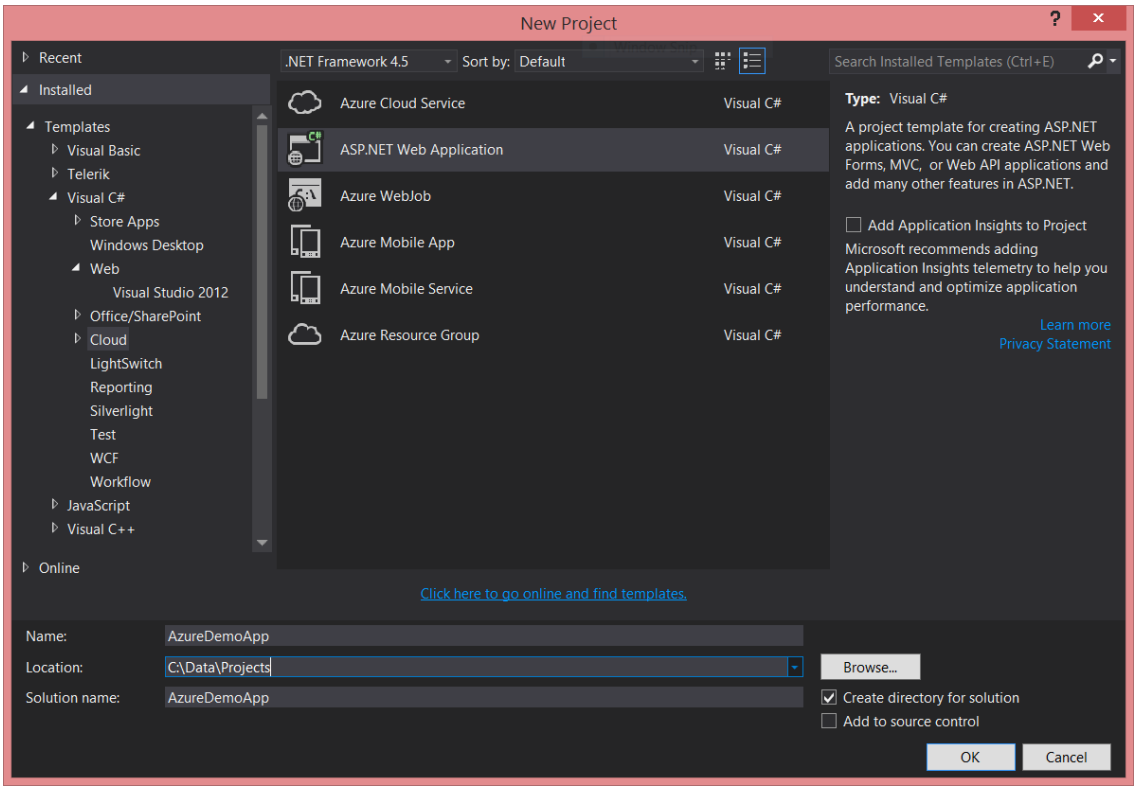
17. Congratulations, if you have completed the steps described above, you are now ready to move on to the next step and write a small awesome client program that can connect to the database you just created.

Step 2: Creating a web application

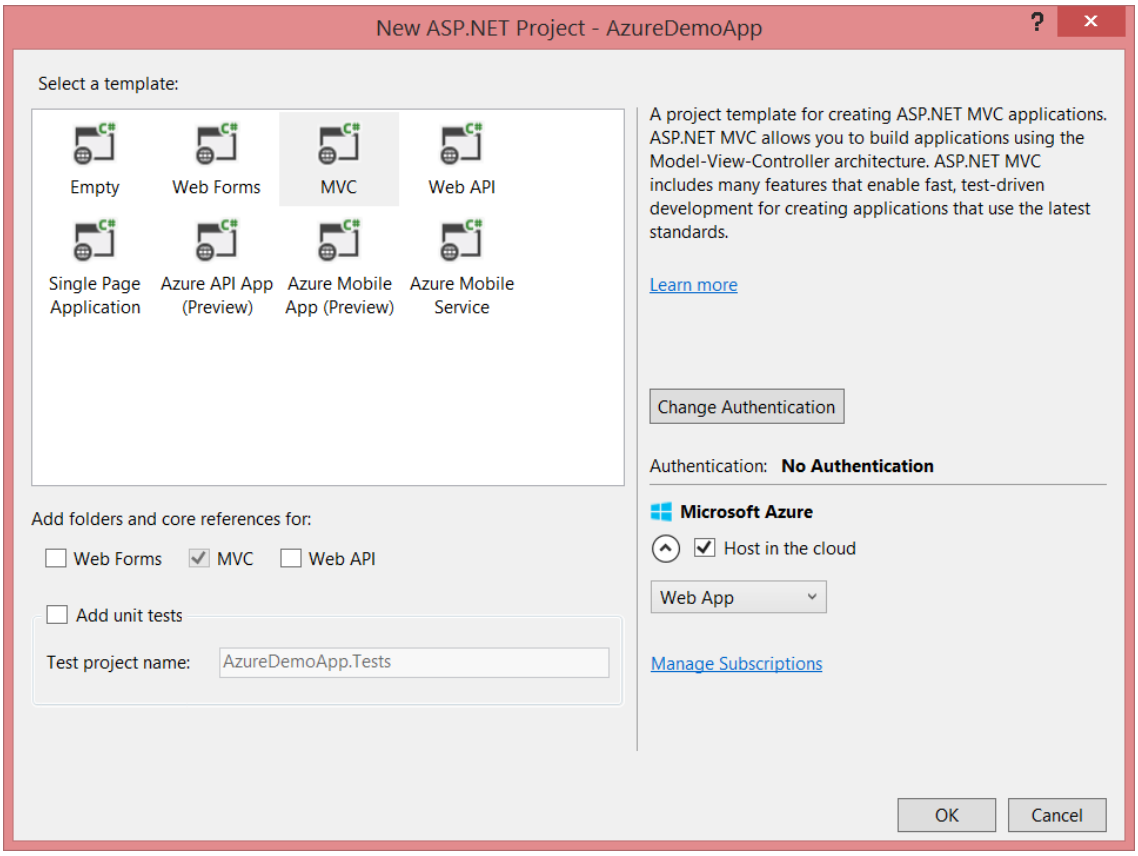
The following step requires Visual Studio 2013 and that you have installed the Azure SDK. You can install the SDK from here: http://azure.microsoft.com/en-us/downloads/

You can use Visual Studio 2013 or Visual Studio 2013 Express for Web for the following step.

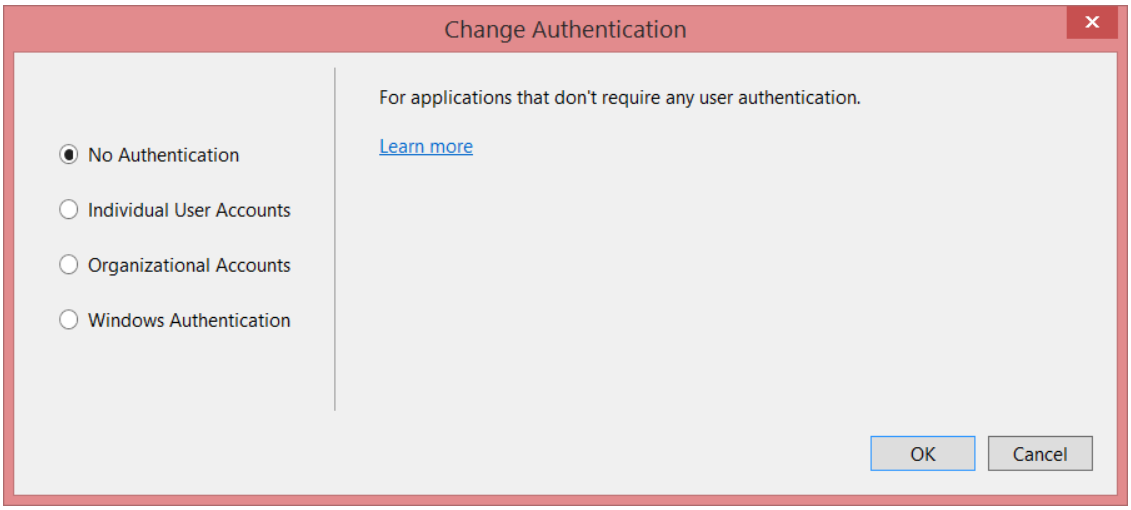
1. To get started open Visual Studio and click the "New Project" in the file menu or from the Visuat Studio welcome page. In the "New Project" dialog box, click "Visual C#" => "Cloud" => "ASP.NET Web Application" and then click “OK”



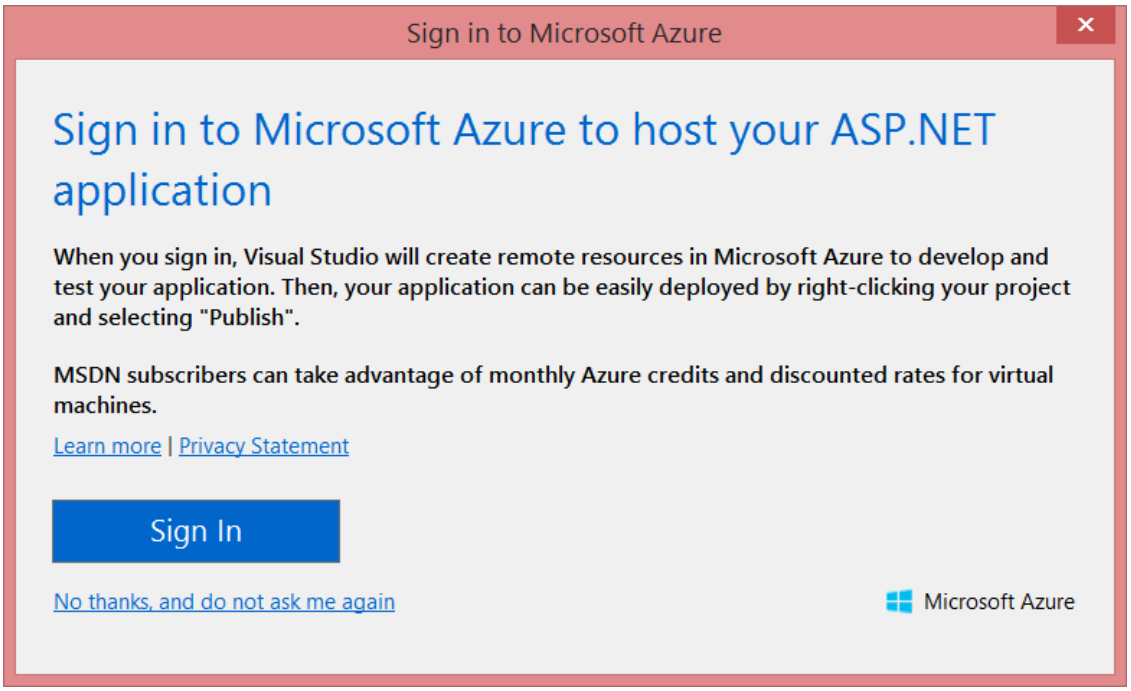
2. In the "New ASP.NET Project" dialog box, select the "MVC" template.



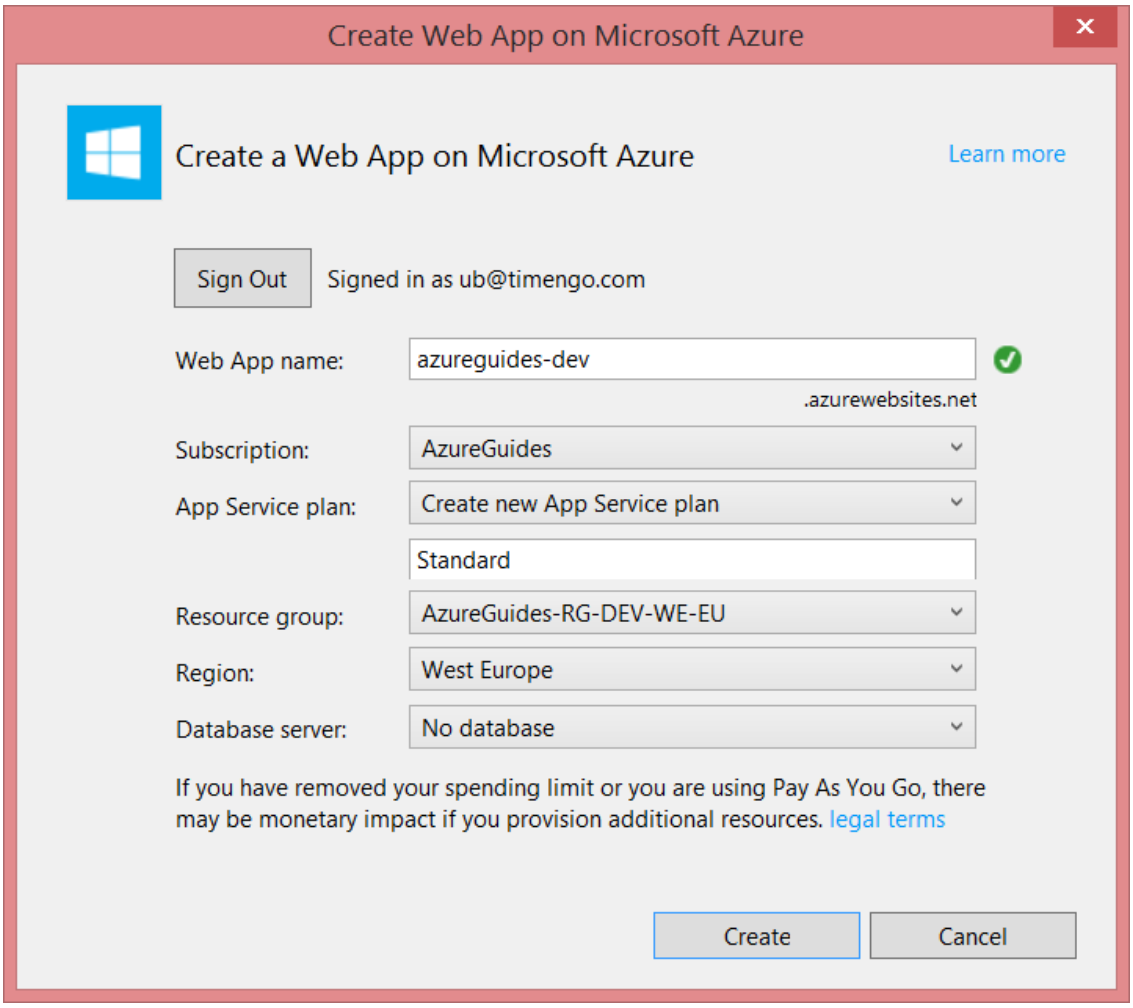
3. By default the MVC project template is set to use "Authentication: Individual Accounts", for this sample project we don't need any authentication for our new MCV application, click the "Change Authentication" button.



4. In the "Change Authentication" dialog box, select the "No Authentication" option, and then click the "OK" button.
5. If you have not already signed in to Azure, Visual Studio will prompt you to do so. Sign in with the ID and password of the account that you used when creating the Azure SQL database.



6. When you're signed in, the Configure Microsoft Azure Web App Settings dialog box asks you what resources you want to create.

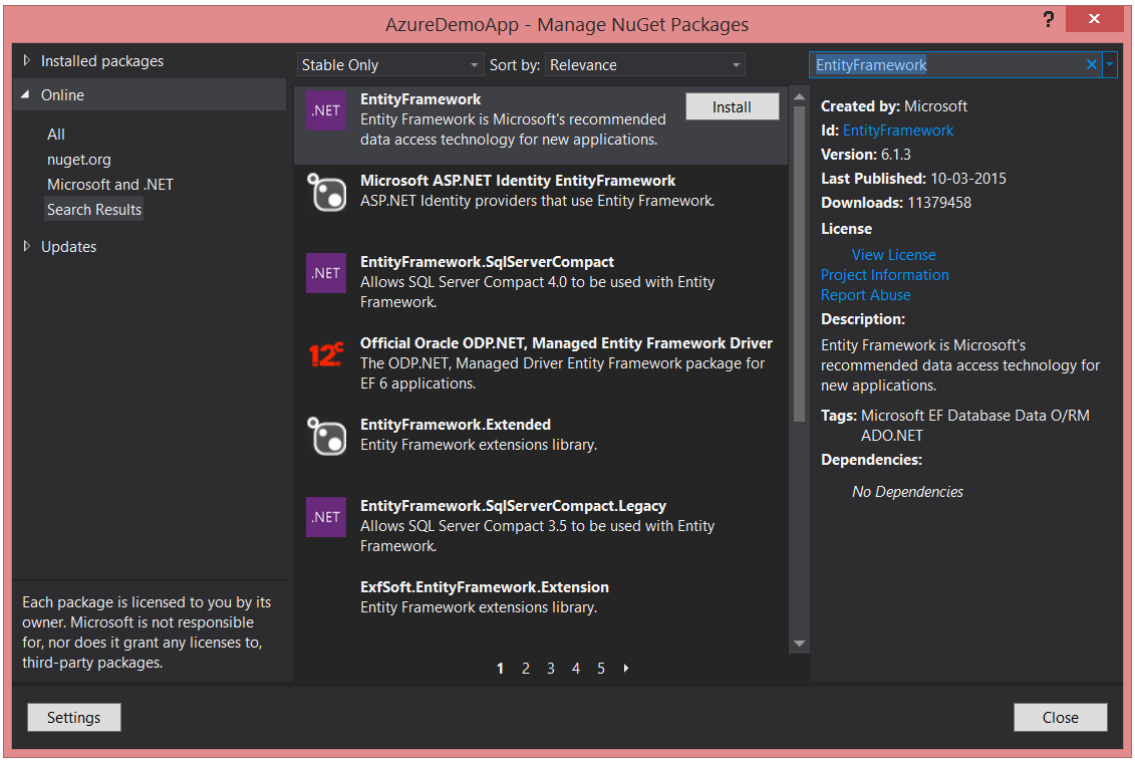


7. In the "Configure Microsoft Azure Web App Settings" dialog box, we assign the needed information. You can enter a different "Web App name" if you prefer, but the name must be unique within the "*.azurewebsites.net" domain. You can assign a custom domain to the web application later.
8. In the "App Service plan" drop-down, select "Create new App Service plan" and give the service plan a name.
9. In the "Resource group" drop-down, select the resource group you created earlier.
10. In the "Region" drop-down list, choose the location that is closest to you. I't importante that you select the same region as the region you selected when creating the Azure SQL Database.
11. Leave the database field unchanged.

Visual studio creates our basic application based on the selected template, and we can finally start writing some code for our new awesome application.

12. For this step by step guide we will Use Entity Framework and "Code First" to define the model in code for our application and then generate the database. So we need to add EntityFramework via

NuGet, right click on the solution and select “Manage NuGet packages for solution” and enter “EntityFramework” in the search and click “Install”.



We can now start creating our model for the application.

13. We start by adding a very simple class objects to the “Models” folder in out MVC application, named “Person.cs”.

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

namespace AzureDemoApp.Models
{
    public class Person
    {
        public int PersonId { get; set; }

        public string Name { get; set; }
    }
}
```

14. Now we can define a "context", which represents a session with the database, allowing us to query and save data to our database. We define a context that derives from the "System.Data.Entity.DbContext" namespace and exposes the typed DbSet<TEntity> for each class in our model. We create a class called “DataContext.cs”.

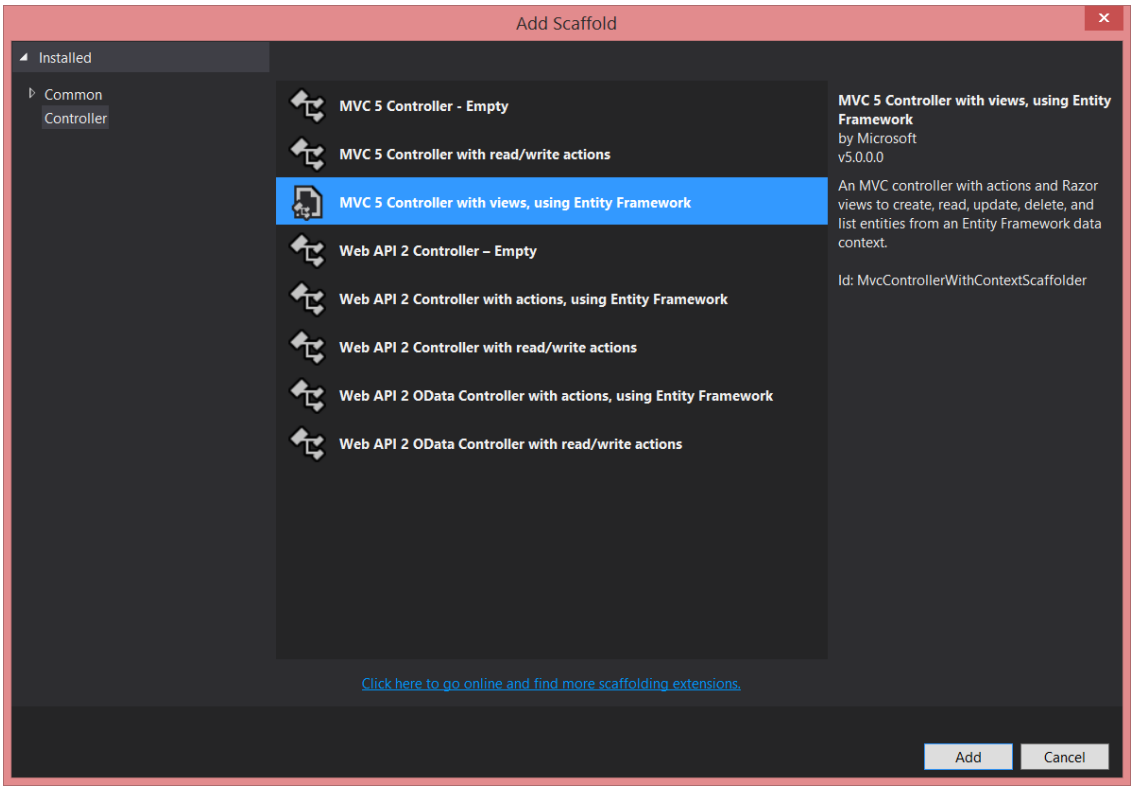
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;
using AzureDemoApp.Models;

namespace AzureDemoApp
{
    public class DataContext : DbContext
    {
        public DbSet<Person> Persons { get; set; }
    }
}
```

15. We also need to provide our application with a connection string to the database, so we all the following to the web.config file:

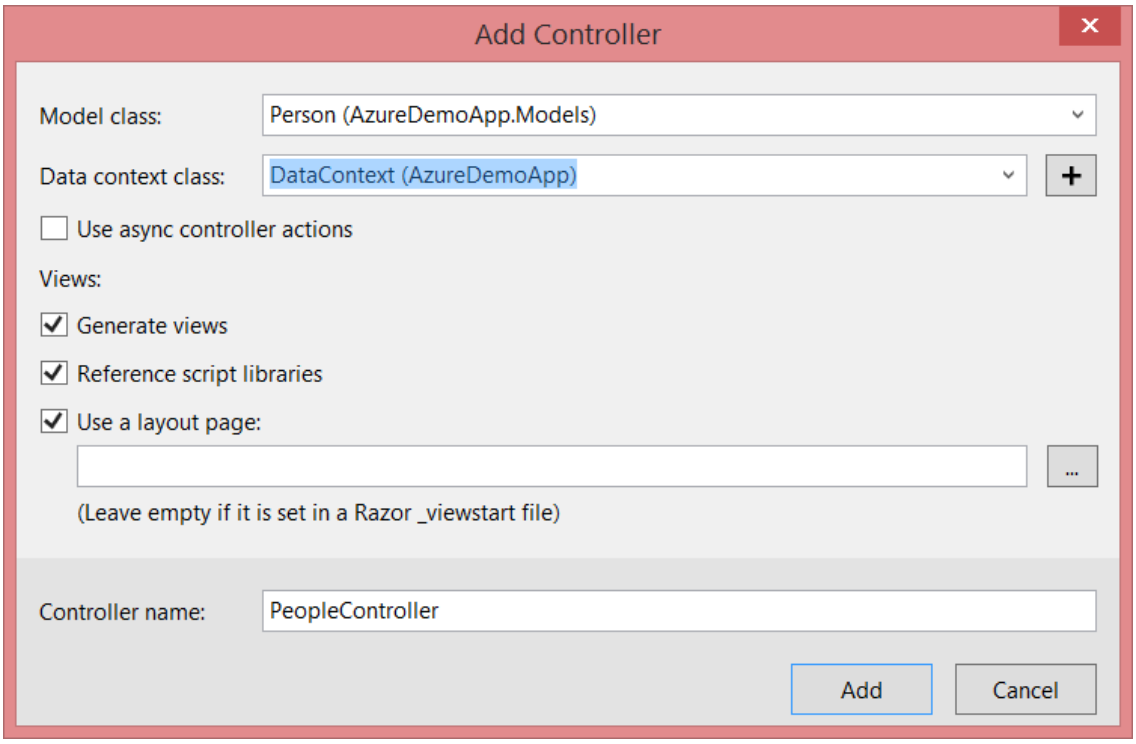
```
<connectionStrings>
    <add name="DefaultConnection"
connectionString="Server=tcp:azureguidesqldev.database.windows.net,1433;Databas
e=AzureGuideDB;User
ID=SqlAzureAdmin@azureguidesqldev;Password={YourSqlPassword};Encrypt=True;Trust
ServerCertificate=False;Connection Timeout=30;"
providerName="System.Data.SqlClient" />
</connectionStrings>
```

All that is missing now is a simple UI so we are able to save and edit the data in our database.

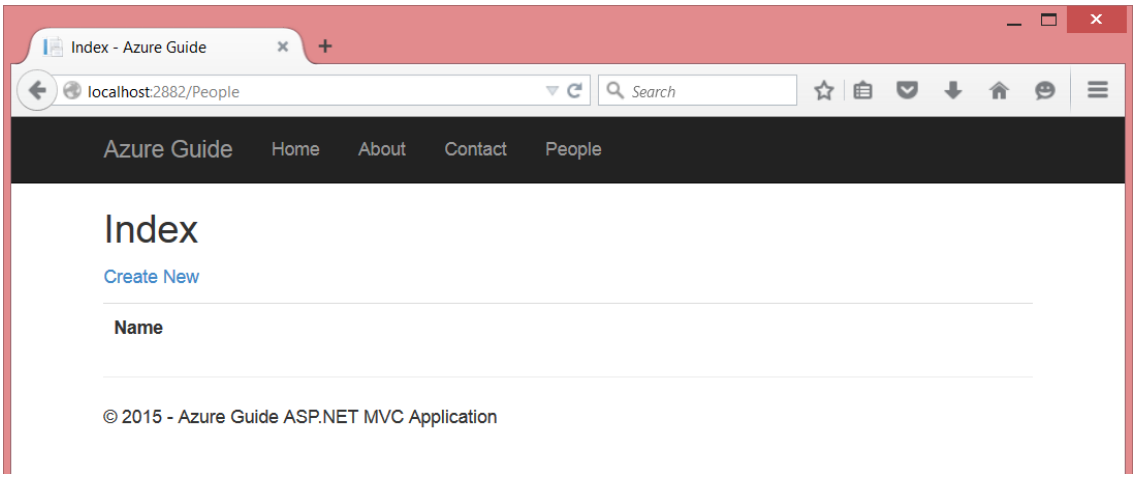
16. To add a new controller, we right click on the “Controllers” folder in our solution and select “Controller”.
17. In the “Add Scaffold” menu select “MVC 5 Controller with views, using Entity Framework” and click “Add”



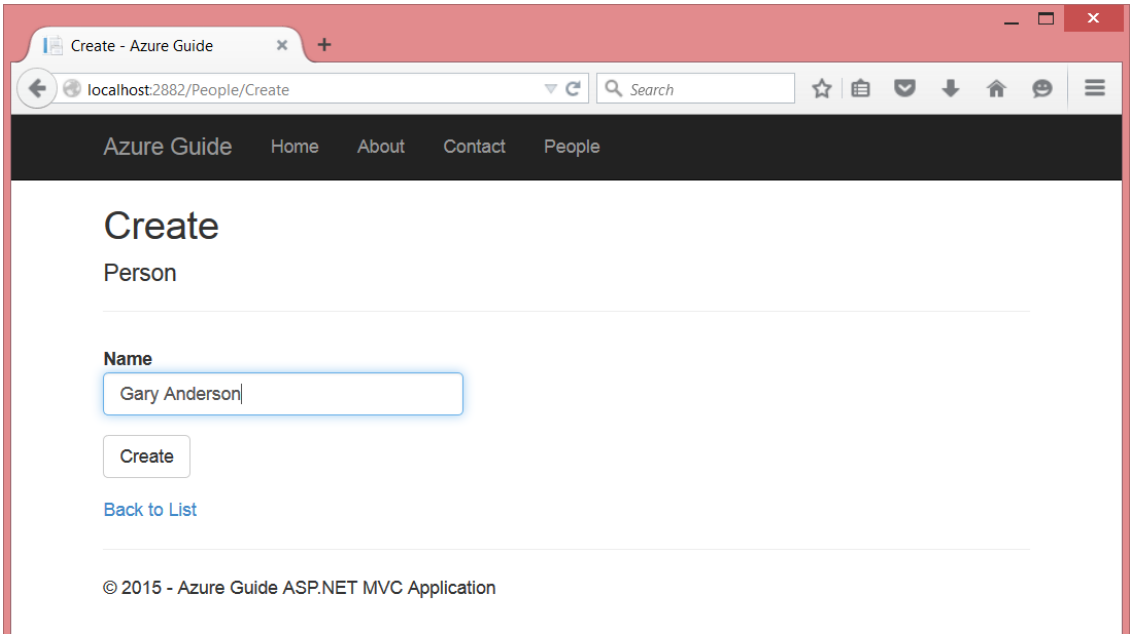
18. In the “Add Controller” dialog select the model class (Person) and select the data Context class (DataContext) and click the “Add” button.



19. This creates a new MVC controller and the required views the list, view, edit and delete our model data in the SQL Azure database.
20. EntityFramework takes care of creating the table and the basic Create, Read, Update and Delete operations.
21. Finally, we add a link to the new controller in the menu by adding the link to the shared layout view (views/shared/_layout.cshtml).
22. If you click debug within Visual Studio, you should see something like this:



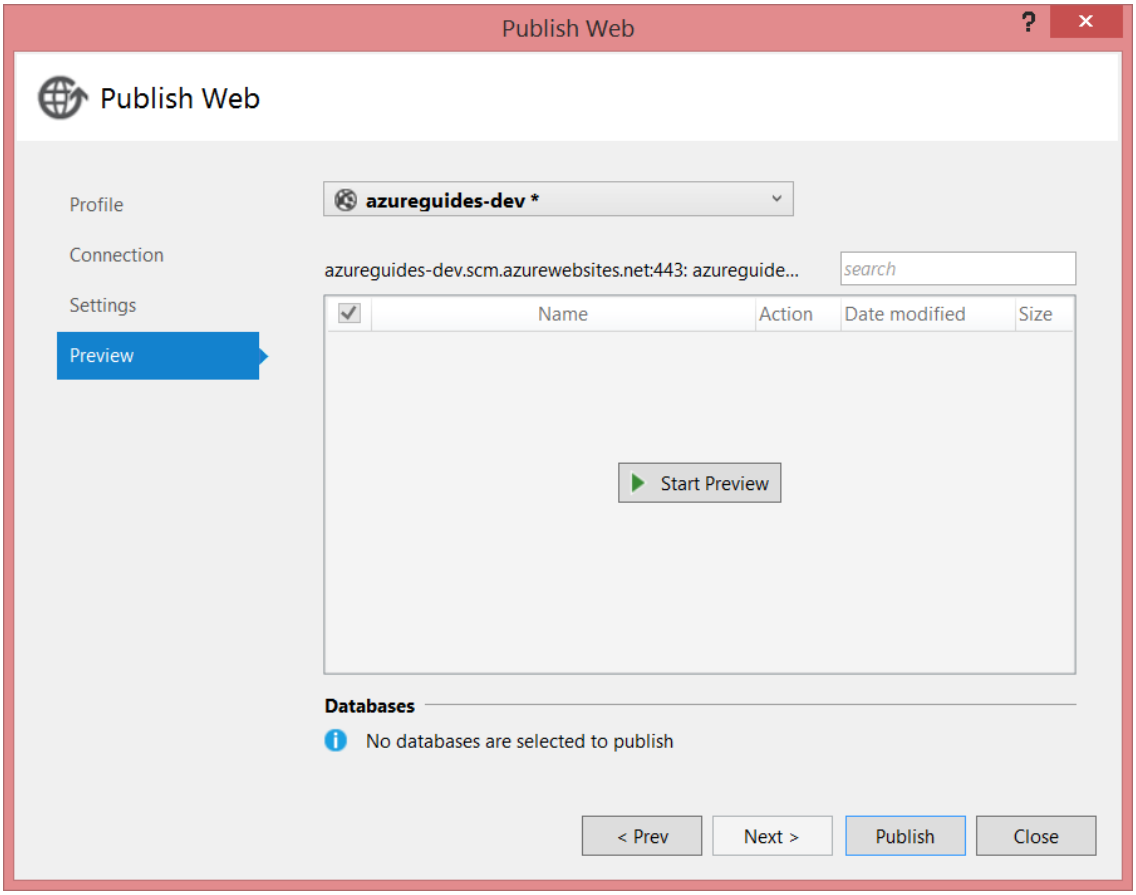
23. Where you can add a new name to the database by clicking “Create New”.



Step 3: Deploying your web application to Azure

Our application is now ready, and we can start the deployment of the application to Azure.

1. In Solution Explorer, right-click the MVC web app project, and select "Publish".
2. The "Preview" tab of the "Publish Web" wizard appears.
3. In the "Publish Web" wizard, click "Publish".



4. Visual Studio now starts the deployment process, when the deployment is completed; a browser is opened displaying the new web application.