Implementing a Custom MySQL ASP.NET Identity Storage Provider

05/22/2015 • 6 minutes to read • +5

In this article

by Raquel Soares De Almeida, Suhas Joshi, Tom FitzMacken

ASP.NET Identity is an extensible system which enables you to create your own storage provider and plug it into your application without re-working the application. This topic describes how to create a MySQL storage provider for ASP.NET Identity. For an overview of creating custom storage providers, see Overview of Custom Storage Providers for ASP.NET Identity.

To complete this tutorial, you must have Visual Studio 2013 with Update 2.

This tutorial will:

- Show how to create a MySQL database instance on Azure.

> - Show how to use a MySQL client tool (MySQL Workbench) to create tables and manage your remote database on Azure.
> - Show how to replace the default ASP.NET Identity storage implementation with our custom implementation on a MVC application project.
>
> This tutorial was originally written by Raquel Soares De Almeida and Rick Anderson (@RickAndMSFT). The sample project was updated for Identity 2.0 by Suhas Joshi. The topic was updated for Identity 2.0 by Tom FitzMacken.

Download completed project

At the end of this tutorial, you will have an MVC application project with ASP.NET Identity working with a MySQL database hosted on Azure.

You can download the completed MySQL storage provider at AspNet.Identity.MySQL (GitHub).

The steps you will perform

In this tutorial you will:

1. Create a MySQL database on Azure
2. Create the ASP.NET Identity tables in MySQL
3. Create an MVC application and configure it to use the MySQL provider
4. Run the app

This topic does not cover the architecture of ASP.NET Identity and the decisions you must make when implementing a customer storage provider. For that information, see Overview of Custom Storage Providers for ASP.NET Identity.

Review MySQL storage provider classes

Before jumping into the steps to create the MySQL storage provider, let's look at the classes that make up the storage provider. You will need classes that manage the database operations and classes that are called from the application to manage users and roles.

Storage classes

- IdentityUser - contains properties for the user.
- UserStore - contains operations for adding, updating or retrieving users.
- IdentityRole - contains properties for roles.
- RoleStore - contains operations for adding, deleting, updating and retrieving roles.

Data access layer classes

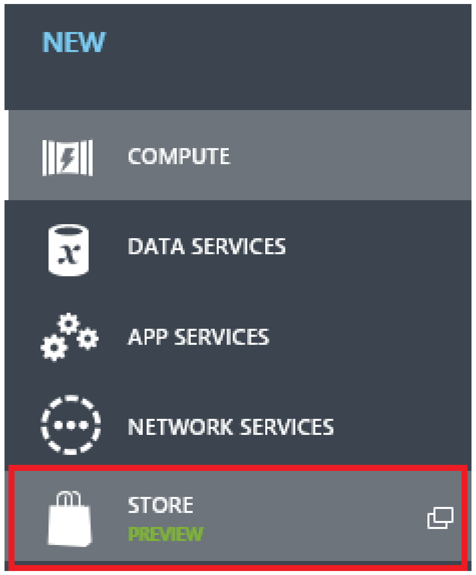
For this example, the data access layer classes contain SQL statements for working with the tables; however, in your code you might want to use object-relational mapping (ORM) such as Entity Framework or NHibernate. In particular, your application may experience poor performance without an ORM that includes lazy loading and object caching. For more information, see ASP.NET Identity 2.0 without Entity Framework?

- MySQLDatabase - contains the MySQL database connection and methods for performing database operations. UserStore and RoleStore are both instantiated with an instance of this class.
- RoleTable - contains database operations for the table that stores roles.
- UserClaimsTable - contains database operations for the table that stores user claims.
- UserLoginsTable - contains database operations for the table that stores user login information.
- UserRoleTable - contains database operations for the table that stores which users are assigned to which roles.
- UserTable - contains database operations for the table that stores users.

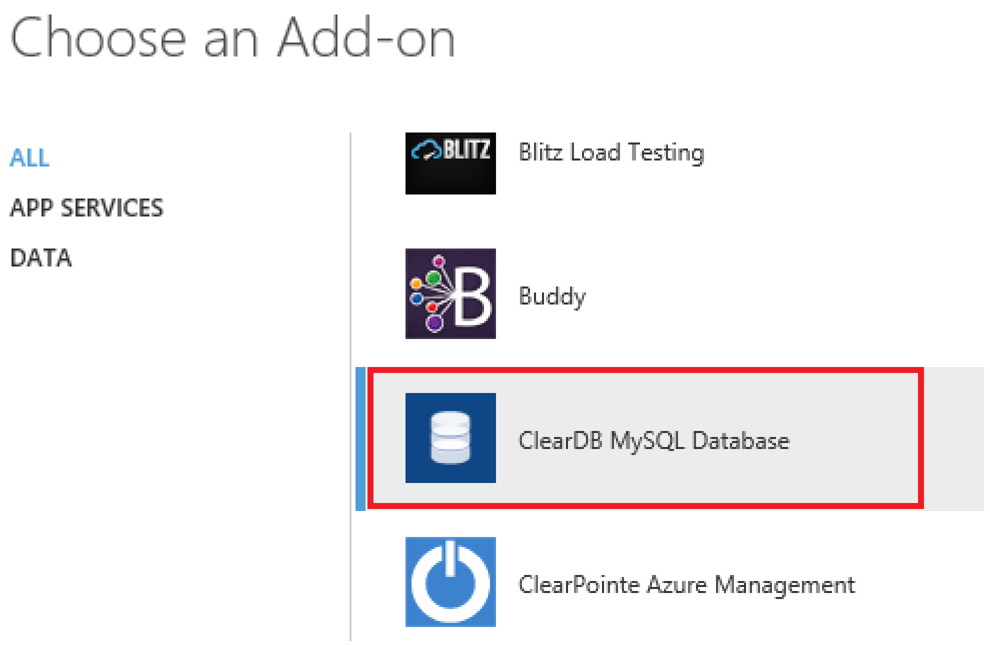
Create a MySQL database instance on Azure

1. Log in to the Azure Portal.

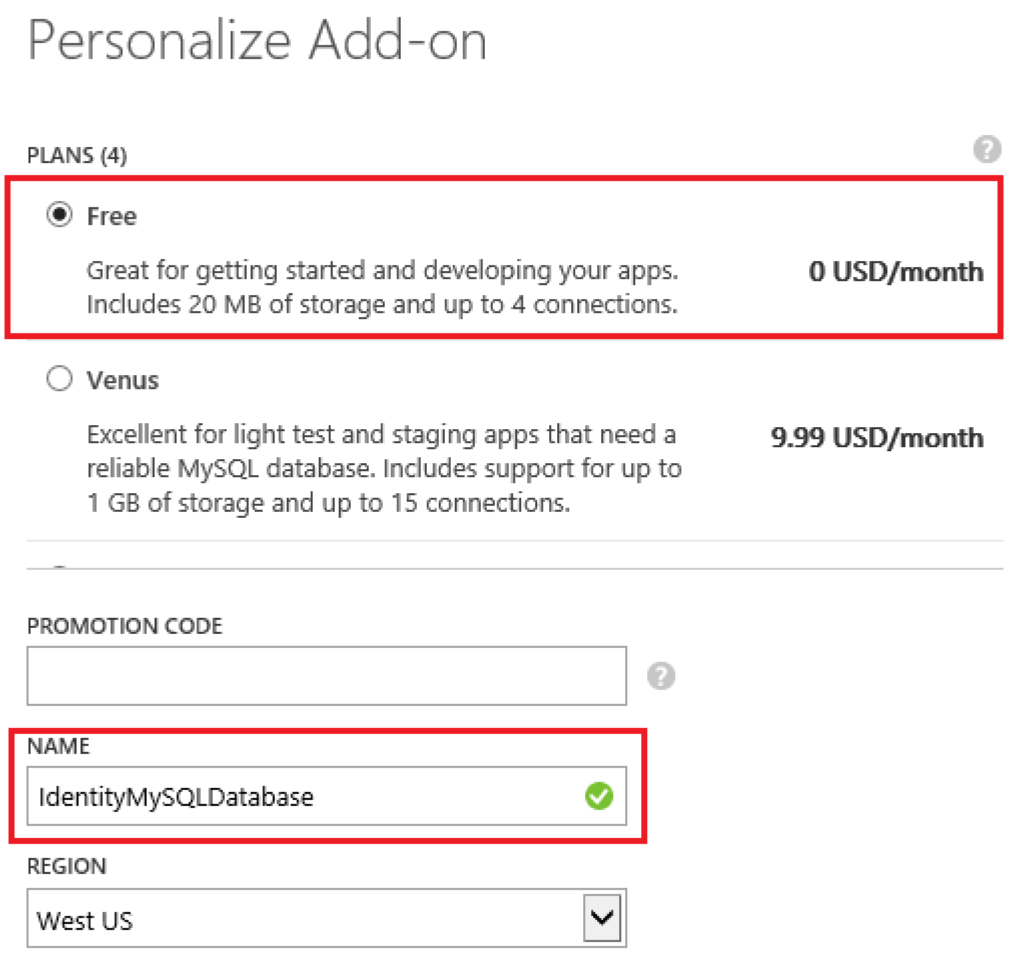
2. Click **+NEW** at the bottom of the page, and then select **STORE**.



3. In the **Choose and Add-on** wizard, select **ClearDB MySQL Database** and click on the next arrow at the bottom right of the dialog.



4. Keep the default **Free** plan and change the **Name** to **IdentityMySQLDatabase**. Select the region nearest you and then click the next arrow.



Personalize Add-on

PLANS (4)

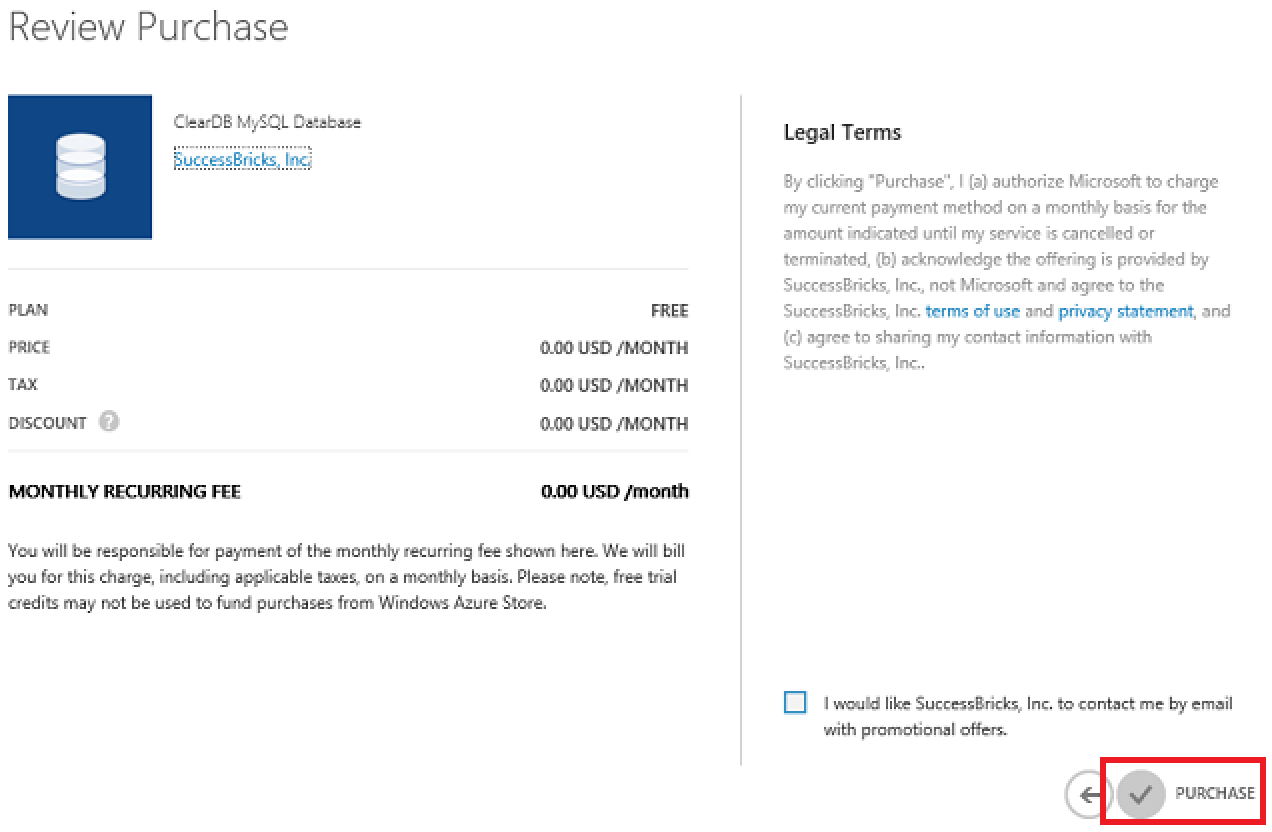
●	**Free** Great for getting started and developing your apps. Includes 20 MB of storage and up to 4 connections.	0 USD/month
○	**Venus** Excellent for light test and staging apps that need a reliable MySQL database. Includes support for up to 1 GB of storage and up to 15 connections.	9.99 USD/month

PROMOTION CODE

NAME
IdentityMySQLDatabase ✓

REGION
West US

5. Click the checkmark to complete the database creation.



Review Purchase

ClearDB MySQL Database
SuccessBricks, Inc.

PLAN	FREE
PRICE	0.00 USD /MONTH
TAX	0.00 USD /MONTH
DISCOUNT	0.00 USD /MONTH
MONTHLY RECURRING FEE	0.00 USD /month

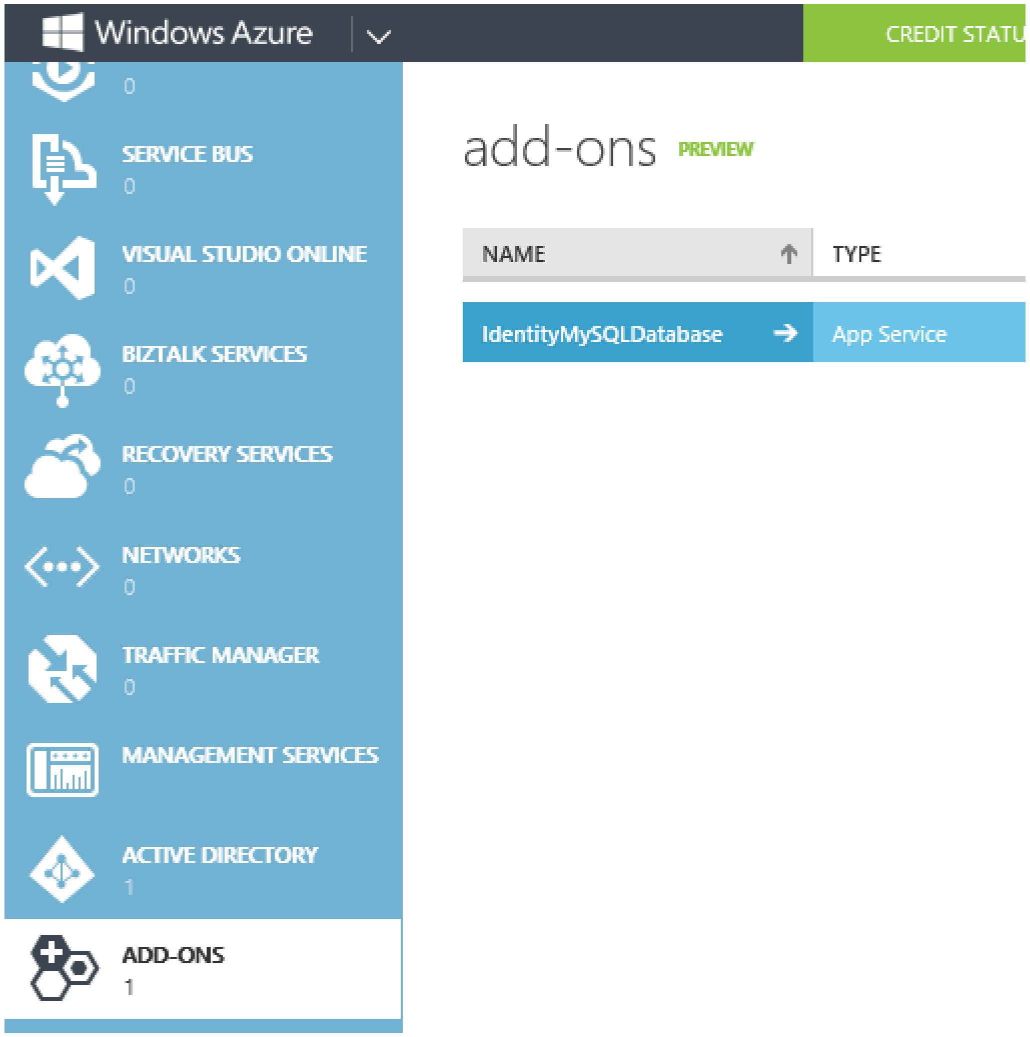
You will be responsible for payment of the monthly recurring fee shown here. We will bill you for this charge, including applicable taxes, on a monthly basis. Please note, free trial credits may not be used to fund purchases from Windows Azure Store.

Legal Terms

By clicking "Purchase", I (a) authorize Microsoft to charge my current payment method on a monthly basis for the amount indicated until my service is cancelled or terminated, (b) acknowledge the offering is provided by SuccessBricks, Inc., not Microsoft and agree to the SuccessBricks, Inc. terms of use and privacy statement, and (c) agree to sharing my contact information with SuccessBricks, Inc..

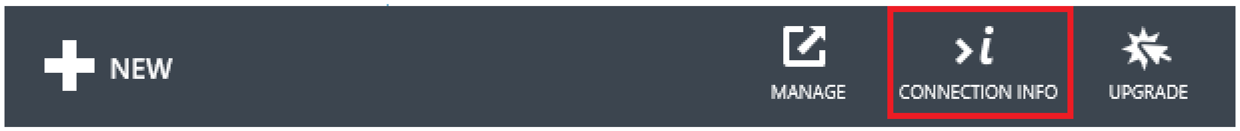
☐ I would like SuccessBricks, Inc. to contact me by email with promotional offers.

6. After your database has been created, you can manage it from the **ADD-ONS** tab in the management portal.

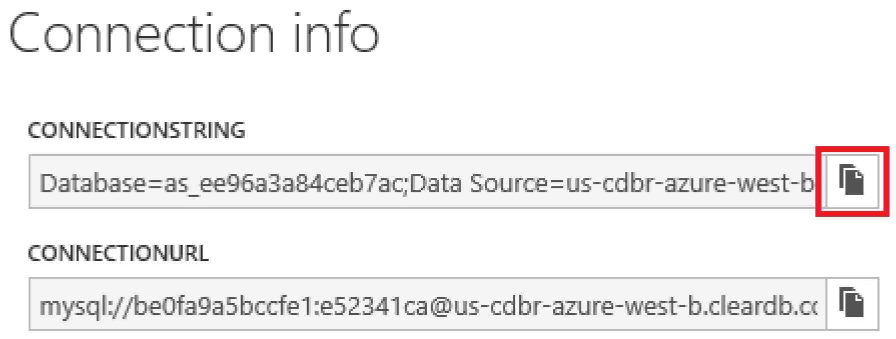


NAME	TYPE
IdentityMySQLDatabase	App Service

7. You can get the database connection information by clicking on **CONNECTION INFO** at the bottom of the page.



8. Copy the connection string by clicking on the copy button and save it so you can use later in your MVC application.



Database=as_ee96a3a84ceb7ac;Data Source=us-cdbr-azure-west-b

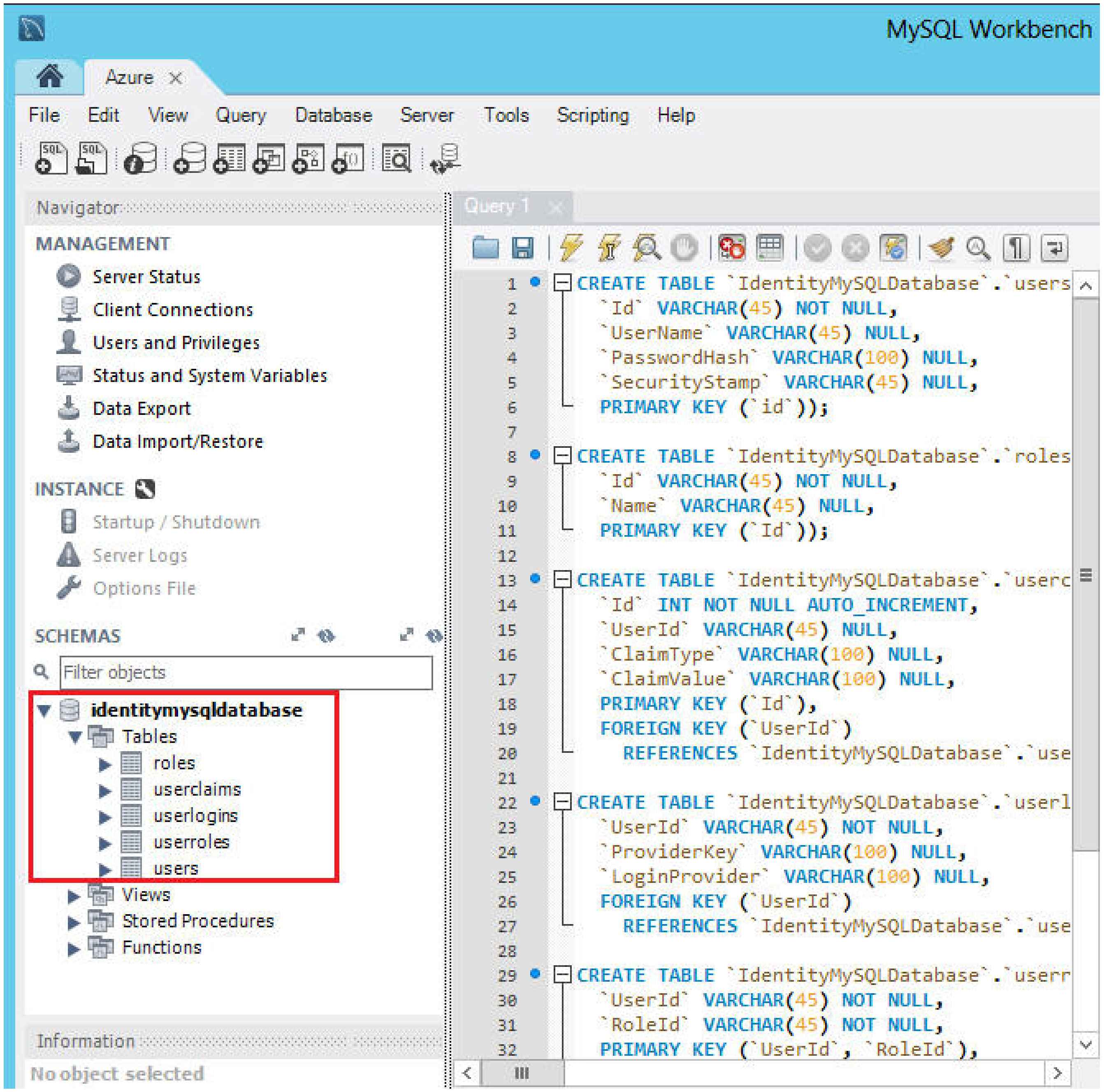
mysql://be0fa9a5bccfe1:e52341ca@us-cdbr-azure-west-b.cleardb.cc

Create the ASP.NET Identity tables in a MySQL database

Install MySQL Workbench tool to connect and manage MySQL database

1. Install the **MySQL Workbench** tool from the MySQL downloads page
2. Launch the app and add click on the **MySQLConnections +** button to add a new connection. Use the connection string data you copied from the Azure MySQL database you created earlier in this tutorial.
3. After establishing the connection, open a new **Query** tab; paste the commands from MySQLIdentity.sql into the query and execute it in order to create the database tables.

4. You now have all the ASP.NET Identity necessary tables created on a MySQL database hosted on Azure as shown below.



```
CREATE TABLE `IdentityMySQLDatabase`.`users
  `Id` VARCHAR(45) NOT NULL,
  `UserName` VARCHAR(45) NULL,
  `PasswordHash` VARCHAR(100) NULL,
  `SecurityStamp` VARCHAR(45) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `IdentityMySQLDatabase`.`roles
  `Id` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NULL,
  PRIMARY KEY (`Id`));

CREATE TABLE `IdentityMySQLDatabase`.`userc
  `Id` INT NOT NULL AUTO_INCREMENT,
  `UserId` VARCHAR(45) NULL,
  `ClaimType` VARCHAR(100) NULL,
  `ClaimValue` VARCHAR(100) NULL,
  PRIMARY KEY (`Id`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`use

CREATE TABLE `IdentityMySQLDatabase`.`userl
  `UserId` VARCHAR(45) NOT NULL,
  `ProviderKey` VARCHAR(100) NULL,
  `LoginProvider` VARCHAR(100) NULL,
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`use

CREATE TABLE `IdentityMySQLDatabase`.`userr
  `UserId` VARCHAR(45) NOT NULL,
  `RoleId` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserId`, `RoleId`),
```

Create an MVC application project from template and configure it to use MySQL provider

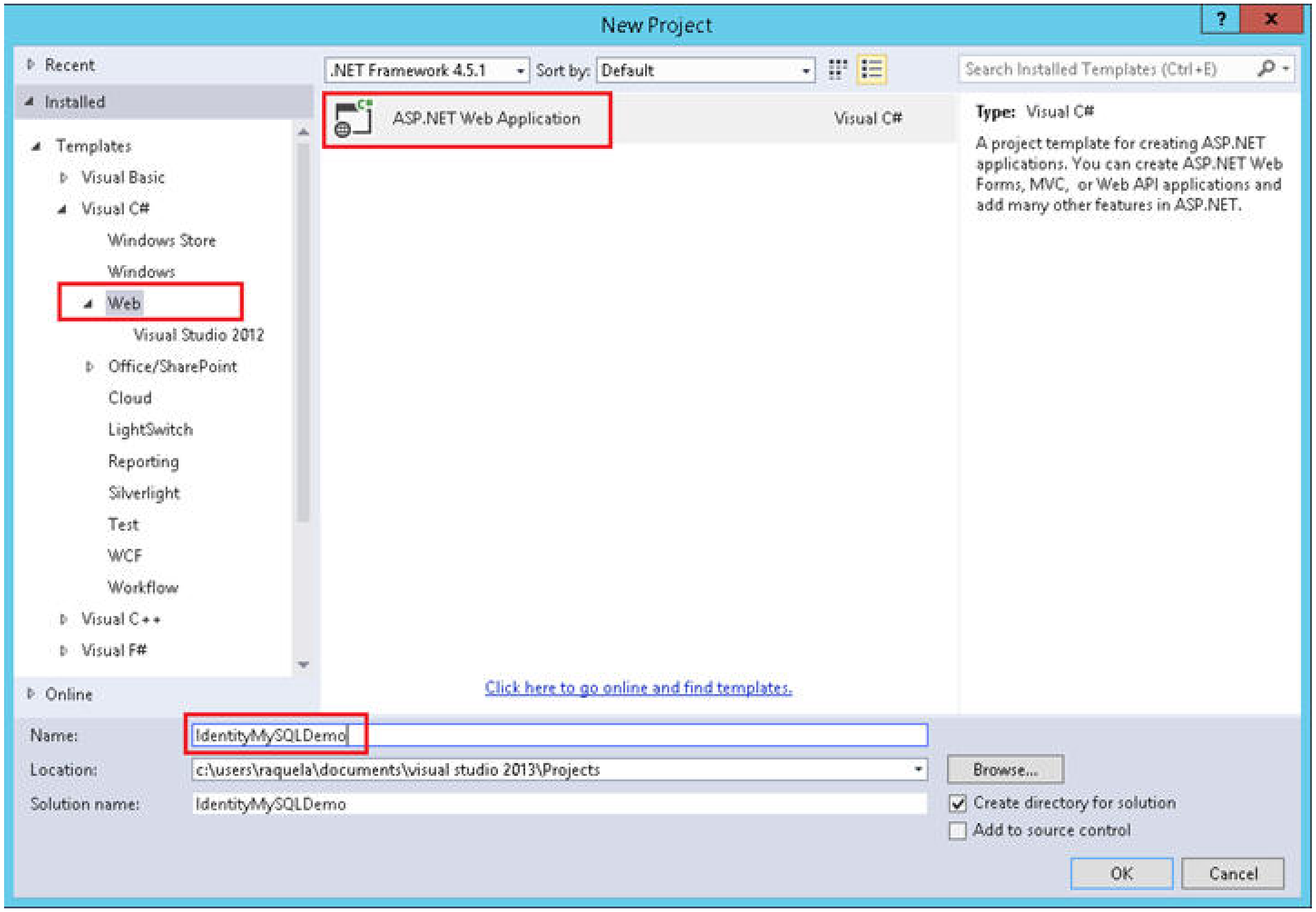
If needed, install either Visual Studio Express 2013 for Web or Visual Studio 2013 with Update 2.

Download the ASP.NET.Identity.MySQL project from GitHub

1. Browse to the repository URL at AspNet.Identity.MySQL (GitHub).
2. Download the source code.
3. Extract the .zip file into a local folder.
4. Open the AspNet.Identity.MySQL solution and build it.

Create a new MVC application project from template

1. Right click the **AspNet.Identity.MySQL** solution and **Add**, **New Project**
2. In the **Add New Project** Dialog select **Visual C#** on the left, then **Web** and then select **ASP.NET Web Application**. Name your project **IdentityMySQLDemo**; and then click OK.



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: IdentityMySQLDemo

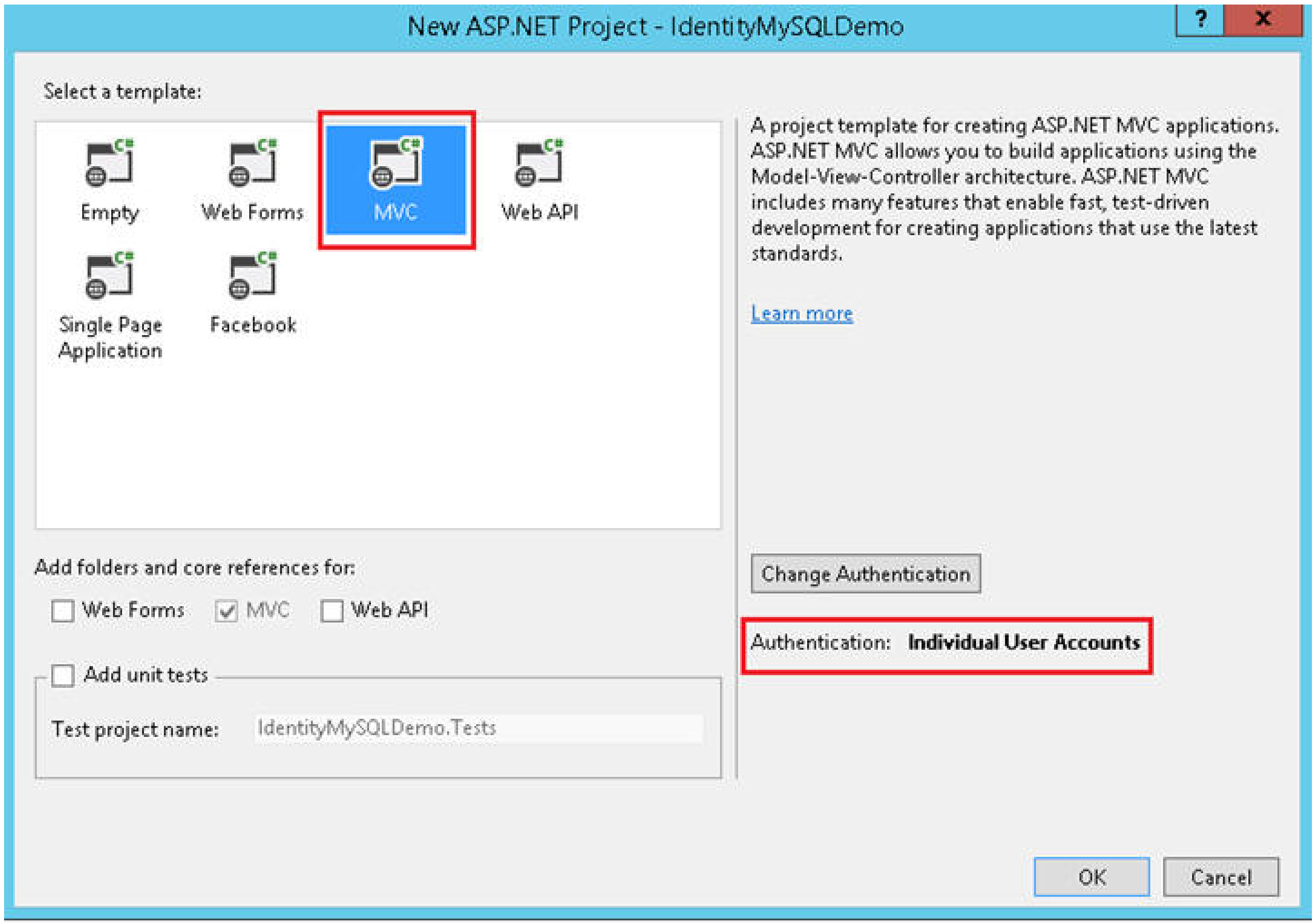
Location: c:\users\raquela\documents\visual studio 2013\Projects

Solution name: IdentityMySQLDemo

[✓] Create directory for solution

[ ] Add to source control

3. In the **New ASP.NET Project** dialog, select the MVC template with the default options (that includes **Individual User Accounts** as authentication method) and click **OK**.



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Add folders and core references for:

☐ Web Forms ☑ MVC ☐ Web API

☐ Add unit tests

Test project name: IdentityMySQLDemo.Tests

Authentication: **Individual User Accounts**

4. In Solution Explorer, right-click your IdentityMySQLDemo project and select **Manage NuGet Packages**. In the search text box dialog, type **Identity.EntityFramework**. Select this package in the list of results and click **Uninstall**. You will be prompted to uninstall the dependency package EntityFramework. Click on Yes as we will no longer this package on this application.

5. Right click the IdentityMySQLDemo project, select **Add**, **Reference, Solution, Projects;** select the AspNet.Identity.MySQL project and click **OK**.

6. In the IdentityMySQLDemo project, replace all references to `using Microsoft.AspNet.Identity.EntityFramework;` with `using AspNet.Identity.MySQL;`

7. In IdentityModels.cs, set **ApplicationDbContext** to derive from **MySqlDatabase** and include a constructor that take a single parameter with the connection name.

C# Copy

```csharp
public class ApplicationDbContext : MySQLDatabase
{
    public ApplicationDbContext(string connectionName)
        : base(connectionName)
    {
    }

    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext("DefaultConnection");
    }
}
```

8. Open the IdentityConfig.cs file. In the **ApplicationUserManager.Create** method, replace instantiating UserManager with the following code:

C# Copy

```csharp
var manager = new ApplicationUserManager(
    new UserStore<ApplicationUser>(
    context.Get<ApplicationDbContext>() as MySQLDatabase));
```

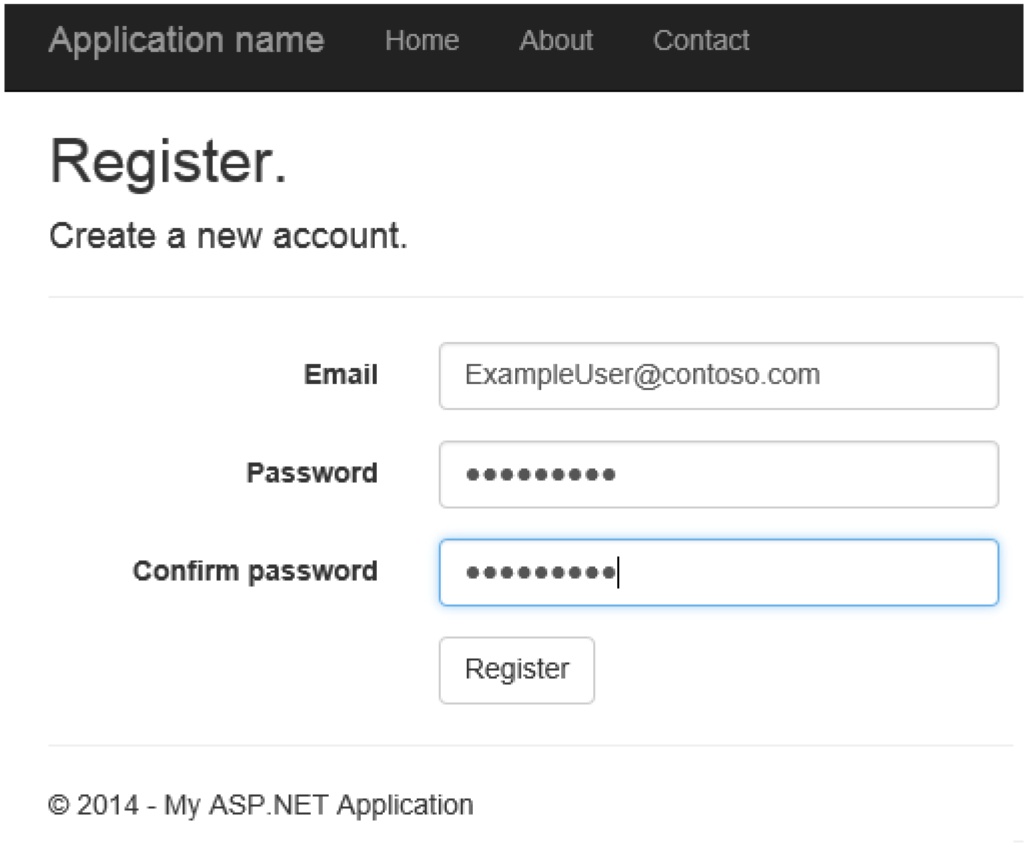
9. Open the web.config file and replace the DefaultConnection string with this entry replacing the highlighted values with the connection string of the MySQL database you created on previous steps:

XML

```xml
<add name="DefaultConnection" connectionString="Database=IdentityMySQLDatabase;
Data Source=<DataSource>;User Id=<UserID>;Password=<Password>"
providerName="MySql.Data.MySqlClient" />
```

Run the app and connect to the MySQL DB

1. Right click the **IdentityMySQLDemo** project and select **Set as Startup Project**
2. Press **Ctrl + F5** to build and run the app.
3. Click on **Register** tab on the top of the page.
4. Enter a new user name and password and then click on **Register**.



Register.

Create a new account.

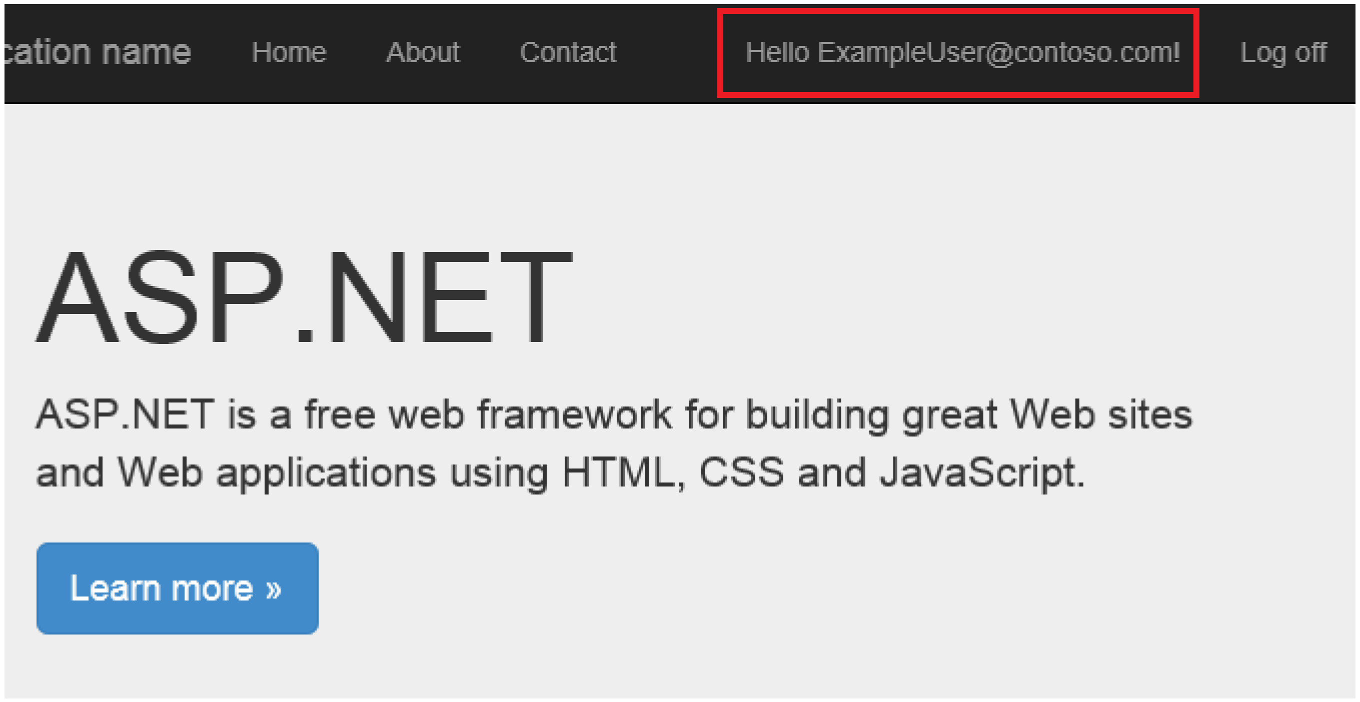
Email: ExampleUser@contoso.com

Password: ●●●●●●●●●

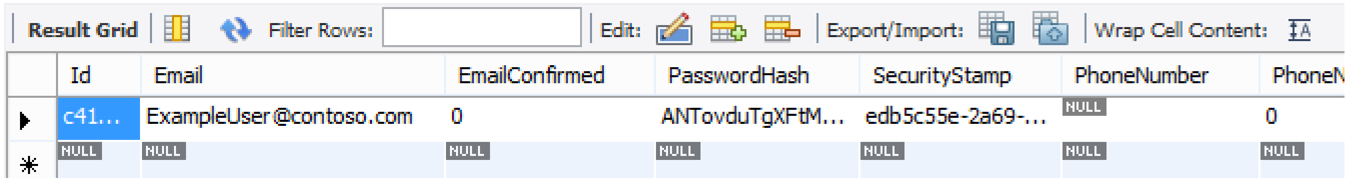
Confirm password: ●●●●●●●●●

© 2014 - My ASP.NET Application

5. The new user is now registered and logged in.


cation name
Home
About
Contact
Hello ExampleUser@contoso.com!
Log off
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »

6. Go back to the MySQL Workbench tool and inspect the **IdentityMySQLDatabase** table's contents. Inspect the users table for the entries as you register new users.



Id	Email	EmailConfirmed	PasswordHash	SecurityStamp	PhoneNumber	PhoneN
c41...	ExampleUser@contoso.com	0	ANTovduTgXFtM...	edb5c55e-2a69-...	NULL	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Next Steps

For more information on how to enable other authentication methods on this app, refer to Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on.

To learn how to integrate your DB with OAuth and to set up roles to limit users access to your app, see Deploy a Secure ASP.NET MVC 5 app with Membership, OAuth, and SQL Database to Azure.

Is this page helpful?

Yes No