

MySQL مدتی است که جزو یکی از محصولات شرکت اوراکل محسوب شده و توسعه دهندگان تجاری باید برای استفاده از آن هزینه کنند. این هزینه نیز اخیراً افزایش یافته و به حداقل 2000 دلار به ازای هر سرور رسیده است ( + ). این عدد واقعاً رقم بالایی برای محصولی محسوب می‌شود که بسیاری از توسعه دهنده‌ها تصور می‌کنند رایگان است. استفاده از این محصول با توجه به مدل تجاری جدید آن فقط در پروژه‌های سورس باز رایگان است (بله فقط در پروژه‌هایی که با مجوز GPL منتشر شوند) و اگر شما یک سیستم تجاری کلاینت سرور را بر این اساس طراحی کنید حتماً باید هزینه‌های مرتبط را نیز پرداخت نمایید ( + ). توضیحی در مورد GPL و MySQL

MySQL AB offers a commercial license for organizations that

do not want

to release the source code for their application.

The change from the LGPL to the GPL for the client libraries was made in 2001 during the development of MySQL 4.0 to help MySQL AB more easily differentiate between a proprietary user who

should

buy a commercial license and a free software user who should use the

GPL

license.

MySQL با توجه به مجوز GPL آن در شرایط زیر رایگان خواهد بود:

- قصد توزیع مجدد آن را نداشته باشید.

- همچنین برنامه‌ی شما نیز به صورت سورس باز تحت مجوز GPL ارائه گردد.

و تنها زمانی در مورد MySQL باید هزینه کنید که:

- قصد توزیع مجدد آن را داشته باشید.

- برنامه‌ی شما سورس باز نبوده و قصد ندارید آن را تحت مجوز GPL ارائه دهید. (که عموماً در مورد برنامه‌های تجاری به همین صورت است)

نکته‌ی دیگری را که باید به آن دقت داشت این است که برای واگذاری MySQL به شرکت اوراکل، اتحادیه اروپا نیز با توجه به وجود بیش از 50 هزار توسعه دهنده‌ی اروپایی که از MySQL استفاده می‌کنند، شرکت اوراکل را موظف کرده است تا این dual licensing (تجاری و سورس باز) را تا سال 2015 حفظ کرده و ادامه دهد ( + ). به این معنا که شرکت اوراکل پس از سال 2015 هیچگونه تعهدی به ارائه‌ی نگارش سورس باز این محصول به هیچ نهاد و یا سازمانی ندارد.

البته این‌ها به معنای پایان دنیا نیست. هم اکنون چهار fork سورس باز از این محصول وجود دارند ( , [Drizzle](#) , [MariaDB](#) , [Percona Server](#) و [OurDelta](#) ) ولی تنها آینده است که میزان موفقیت، پایداری و تداوم آن‌ها را مشخص خواهد کرد.

## نظرات خوانندگان

نویسنده: ghafoori

تاریخ: ۲۱:۴۶:۴۶ ۱۳۸۹/۰۹/۲۰

آخر سر من نفهمیدم مولا به سر سایتهایی که از Mysql استفاده می کنند چی پیش میاد

نویسنده: وحید نصیری

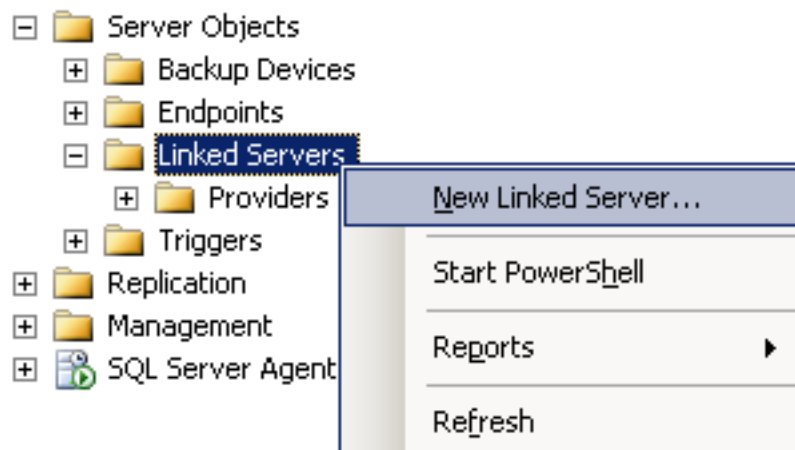
تاریخ: ۲۲:۲۹:۴۲ ۱۳۸۹/۰۹/۲۰

اکثر سایتهایی که مشاهده می کنید (و عموماً بر مبنای PHP هستند) مثل فروم‌ها، وبلاگ‌ها و غیره، سورس باز هستند و یا مجوز GPL دارند یا احتمالاً هم خانواده‌اند و مشکلی نخواهند داشت. در غیراینصورت اگر کار نهایی تجاری و سورس بسته باشد و شخصی هم گزارش بدهد و هاست هم در کشوری باشد که مسایل کپی رایت را رعایت می‌کند، سایت را معلق می‌کنند، به علاوه سایر مسایل قانونی مرتبط.

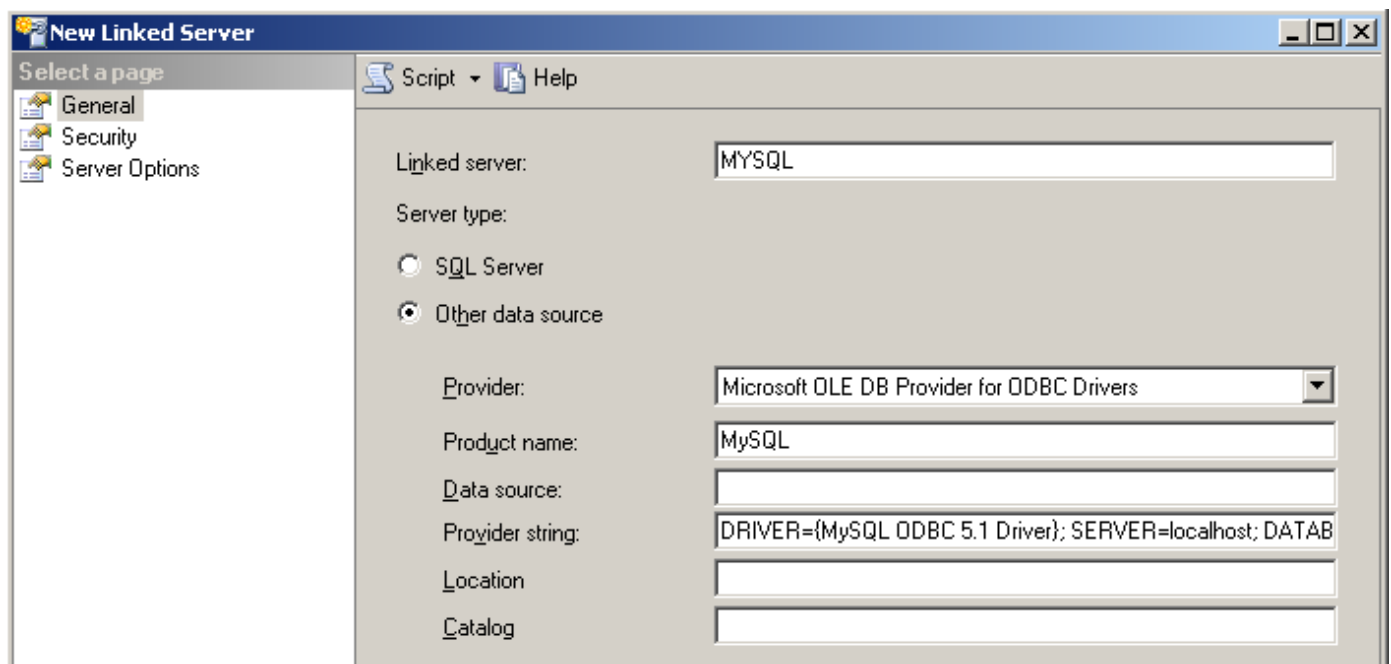
اگر SQL Server و MySQL بر روی سیستم شما نصب است، روشی ساده برای انتقال اطلاعات بین این دو وجود دارد که نیازی به دخالت هیچ نوع برنامه‌ی جانبی نداشته و با امکانات موجود قابل مدیریت است.

ایجاد یک Linked server

برای اینکه SQL Server را به MySQL متصل کنیم می‌توان بین این دو یک Linked server تعریف کرد و سپس دسترسی به بانک‌های اطلاعاتی MySQL همانند یک بانک اطلاعاتی محلی SQL Server خواهد شد که شرح آن در ادامه ذکر می‌شود. ابتدا نیاز است تا درایور ODBC مربوط به MySQL دریافت و نصب شود. آن‌را می‌توانید از اینجا دریافت کنید: (+) سپس در management studio را گشوده و در قسمت Server objects، بر روی گزینه‌ی Linked servers کلیک راست نمائید. از منوی ظاهر شده، گزینه‌ی New linked server را انتخاب کنید:



در ادامه، باید تنظیمات زیر را در صفحه‌ی باز شده وارد کرد:

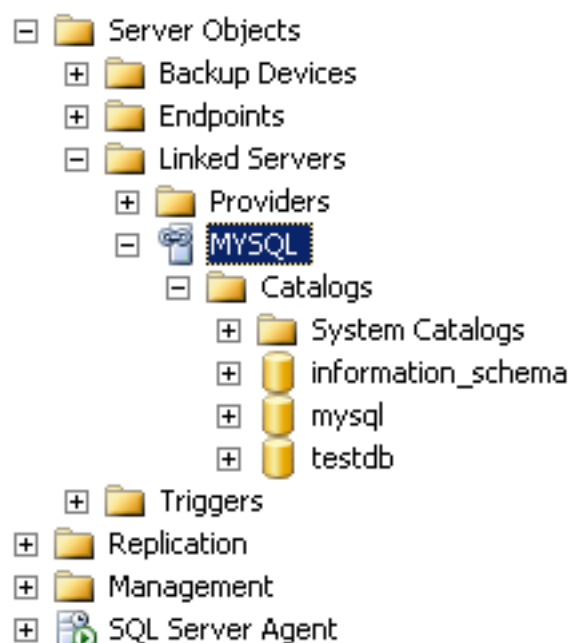


در قسمت Linked server و Product name نام دلخواهی را وارد کنید.  
 Provider انتخابی باید از نوع Microsoft OLE DB Provider for ODBC Drivers باشد.  
 مهم‌ترین تنظیم آن، قسمت Provider string است که باید به صورت زیر وارد شود (در غیر اینصورت کار نمی‌کند):

```
DRIVER={MySQL ODBC 5.1 Driver}; SERVER=localhost; DATABASE=testdb; USER=root; PASSWORD=mypass;  

OPTION=3;PORT=3306; CharSet=UTF8;
```

در اینجا نام دیتابیس پیش فرض، نام کاربری اتصال به MySQL و Password و غیره را می‌توان تنظیم کرد.  
 پس از انجام این تنظیمات بر روی دکمه‌ی Ok کلیک کنید تا Linked server ساخته شود:



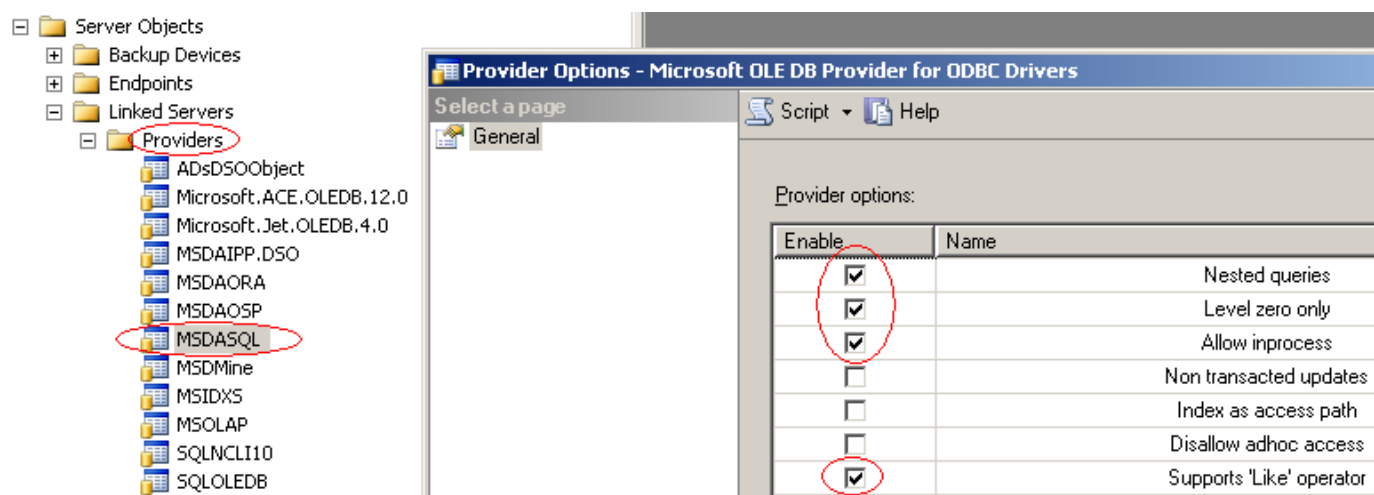
اگر لیست بانک‌های اطلاعاتی را مشاهده نمودید، یعنی اتصال به درستی برقرار شده است.

تنظیمات ثانویه:

تا اینجا اس کیوال سرور به MySQL متصل شده است، اما برای استفاده بهینه از امکانات موجود نیاز است تا یک سری تغییرات دیگر را هم اعمال کرد.

تنظیم MSDASQL Provider :

در همان قسمت Linked provider ، ذیل قسمت Providers ، گزینه‌ی MSDASQL را انتخاب کرده و بر روی آن کلیک راست نمایید. سپس صفحه‌ی خواص آن را انتخاب کنید تا بتوان تنظیمات زیر را به آن اعمال کرد. این پروایدر جهت اتصال به MySQL مورد استفاده قرار می‌گیرد.



فعال سازی RPC :

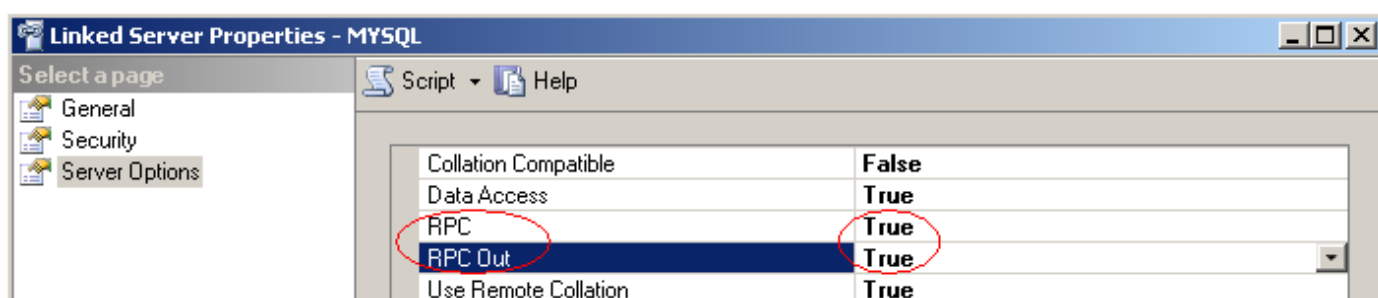
برای اینکه بتوان از طریق SQL Server رکوردی را در یکی از جداول بانک‌های اطلاعاتی MySQL متصل شده ثبت نمود، می‌توان از دستور زیر استفاده کرد:

```
EXECUTE('insert into testdb.testtable(f1,f1) values(1, 'data')') at mysql
```

اینجا testdb نام بانک اطلاعاتی اتصالی MySQL است و testTable هم نام جدول مورد نظر. MySQL ایی که در آخر عبارت ذکر شده همان نام linked server ایی است که پیشتر تعریف کردیم. به محض سعی در اجرای این کوئری خطای زیر ظاهر می‌شود:

```
Server 'mysql' is not configured for RPC.
```

برای رفع این مشکل، مجدداً به صفحه‌ی خواص همان linked server ایجاد شده مراجعه کنید. در قسمت Server options دو گزینه مرتبط به RPC باید فعال شوند:



و اکنون برای کوئری گرفتن از اطلاعات ثبت شده هم از عبارت زیر می‌توان استفاده کرد:

```
SELECT * FROM OPENQUERY(mysql, 'SELECT * FROM testdb.testtable')
```

در این کوئری، MySQL نام Linked server ثبت شده است و testdb هم یکی از بانک‌های اطلاعاتی MySQL مورد نظر.

انتقال تمام اطلاعات یک جدول از بانک اطلاعاتی MySQL به SQL Server

پس از برقراری اتصال، اکنون import کامل یک جدول MySQL به SQL Server به سادگی اجرای کوئری زیر می‌باشد:

```
SELECT * INTO MyDb.dbo.testtable FROM openquery(MYSQL, 'SELECT * FROM testdb.testtable')
```

در این کوئری، MySQL همان Linked server تعریف شده است. MyDB نام بانک اطلاعاتی موجود در SQL Server جاری است و testtable هم جدولی است که قرار است اطلاعات testdb.testtable بانک اطلاعاتی MySQL به آن وارد شود.

با اطلاعات فارسی هم (در سمت SQL Server) مشکلی ندارد. همانطور که مشخص است، در اطلاعات provider string ذکر شده، مقدار charset به utf8 تنظیم شده و همچنین اگر نوع collation فیلدهای تعریف شده در MySQL نیز به utf8\_persian\_ci تنظیم شده باشد، با مشکل ثبت اطلاعات فارسی به صورت ??? مواجه نخواهید شد.

نکته:

اگر بانک اطلاعاتی MySQL شما بر روی local host نصب نیست، جهت فعال سازی دسترسی ریموت به آن، می‌توان به یکی از نکات زیر مراجعه کرد و سپس این اطلاعات جدید باید در همان قسمت provider string مرتبط با تعریف linked server وارد شوند:

[مشکل Permission در Mysql برای کاربر راه دور](#)

[How to access MySQL remotely](#)

مطالب مشابه:

[HOWTO: Setup SQL Server Linked Server to MySQL](#)

[Creating Linked server to MYSQL from SQL Server](#)

[MySQL Linked Server on SQL Server 2008](#)

هیچکدام از این روش‌ها قابل استفاده نبودند چون provider string صحیحی را نهایتاً تولید نمی‌کنند. همچنین تمام این روش‌ها

مبتنی است بر ایجاد DSN در کنترل پنل که اصلاً نیازی به آن نیست و اضافی است.

## نظرات خوانندگان

نویسنده: وحید نصیری  
تاریخ: ۱۳:۳۳:۴۵ ۱۳۹۰/۰۵/۱۱

دو نکته تکمیلی:

- برای انتقال کل اطلاعات یک جدول از SQL Server به MySQL لینک شده با اجرای فقط یک کوئری:

```
insert into openquery(mysql, 'select f1,f2 from testdb.testtable') select f1,f2 from testdb.dbo.myTable
```

در اینجا testdb.testtable مربوط به طرف MySQL است و testdb.dbo.myTable مربوط به طرف SQL Server .

- کوئری گرفتن از Linked server به صورت زیر هم می‌تواند باشد (بر اساس دیتابیس پیش فرض ذکر شده در پروایدر استرینگ):

```
SELECT * FROM mysql...testtable
```

نویسنده: Mohsen  
تاریخ: ۰۸:۵۲:۵۶ ۱۳۹۰/۰۵/۱۲

خیلی عالی بود. ممنون از مطالب کاربردی و بروز شما مهندس نصیری!

نویسنده: Hamedpalik  
تاریخ: ۱۹:۰۶:۳۰ ۱۳۹۰/۰۵/۲۱

سلام جناب نصیری  
از مطالب بسیار خوبتون ممنون هستم.  
خیلی کمکم کرد.



در این مقاله قصد داریم اطلاعات مفیدی را در مورد طراحی دیتابیس‌های چند زبانه، در اختیار شما بگذاریم. مدتی قبل به طراحی دیتابیس‌های چند زبانه بودن توضیحات کالا را برای مشتریانی از کشورهای مختلف پشتیبانی می‌کرد، نیاز داشتم. وقتی شروع به پیاده سازی طرح دیتابیس کردم، جواب سرراست نبود. زمانیکه در وب برای بهترین راه جستجو می‌کردم، با نظرات و روش‌های زیادی مواجه شدم. در ادامه بعضی از روش‌های محبوب را بیان می‌کنم.

**ستون اضافی :** این ساده‌ترین راه است و به ازای هر ستونی که نیاز به ترجمه داشته باشد، ستون اضافی در نظر می‌گیریم.

```
CREATE TABLE app_product (
  Id Int IDENTITY NOT NULL,
  Description_en Text,
  Description_pl Text,
  PRIMARY KEY (Id)
);
```

مزایا :

سادگی

کوئری‌های آسان (بدون نیاز به join)

معایب :

اضافه کردن زبان جدید نیاز به تغییر جداولی که چند زبانه هستند دارد

اگر وارد کردن داده برای همه زبان‌ها الزامی نباشد (بعضی جاها فقط زبان پیش فرض الزامی است) ممکن است داده‌های زیاد و یا فیلدهای خالی در دیتابیس ایجاد شود

نگهداری آن مشکل است

یک جدول ترجمه : این روش تمیزترین راه از دیدگاه ساختار دیتابیس به نظر می‌رسد. شما همه متن‌هایی را که نیاز به ترجمه دارد، در یک جدول ذخیره می‌کنید.

```
CREATE TABLE ref_language (
  Code Char(2) NOT NULL,
  Name Varchar(20) NOT NULL,
  PRIMARY KEY (Code)
);

CREATE TABLE app_translation (
  Id Int IDENTITY NOT NULL,
  PRIMARY KEY (Id)
);

CREATE TABLE app_translation_entry (
  TranslationId Int NOT NULL,
  LanguageCode Char(2) NOT NULL,
  Text Text NOT NULL,
  FOREIGN KEY (TranslationId) REFERENCES app_translation(Id),
  FOREIGN KEY (LanguageCode) REFERENCES ref_language(Code)
);

CREATE TABLE app_product (
  Id Int IDENTITY NOT NULL,
  Description Int NOT NULL,
  PRIMARY KEY (Id),
  FOREIGN KEY (Description) REFERENCES app_translation(Id)
);
```

مزایا :

اضافه کردن زبان جدید به تغییر طرح دیتابیس نیاز ندارد

به نظر تمیز است و رویکرد رابطه‌ای دارد

همه ترجمه‌ها در یک مکان است (بعضی‌ها می‌گویند این جز معایب است چون امکان خوانایی و نگهداری کمتر است)

معایب :

کوئری‌های پیچیده (به join های چندگانه نیاز دارد تا شرح کالا را به درستی نمایش دهد)

پیچیدگی زیاد

**جدول ترجمه اضافی به ازای هر جدول چند زبانه :** برای هر جدولی که نیاز به ترجمه دارد یک جدول اضافی ساخته می‌شود.

جدول اصلی داده‌های غیر مرتبط به زبان و جدول دوم همه اطلاعات ترجمه شده را ذخیره می‌کند.

```
CREATE TABLE ref_language (
    Code Char(2) NOT NULL,
    Name Varchar(20) NOT NULL,
    PRIMARY KEY (Code)
);

CREATE TABLE app_product (
    Id Int IDENTITY NOT NULL,
    PRIMARY KEY (Id)
);

CREATE TABLE app_product_translation (
    ProductId Int NOT NULL,
    LanguageCode Char(2) NOT NULL,
    Description Text NOT NULL,
    FOREIGN KEY (ProductId) REFERENCES app_product(Id),
    FOREIGN KEY (LanguageCode) REFERENCES ref_language(Code)
);
```

مزایا :

اضافه کردن زبان جدید به تغییر طرح دیتابیس نیاز ندارد

کوئری‌های نسبتاً ساده است

معایب :

ممکن است تعداد جداول دو برابر شود

سه مثال نشان داده شده در بالا به ما ایده می‌دهند که چگونه روش‌های مختلف ممکن است استفاده شوند. البته اینجا همه گزینه‌های ممکن نیستند، فقط محبوبترین روش‌ها هستند و شما می‌توانید آنها را ویرایش کنید؛ به عنوان مثال با تعریف View های اضافی که join های پیچیده شما را در کدها، ذخیره می‌کنند. راه حلی که شما انتخاب می‌کنید به نیازمندی‌های پروژه وابسته است. اگر شما به سادگی نیاز دارید و مطمئن هستید تعداد زبان‌های پشتیبانی کم و ثابت است، می‌توانید راه حل اول را انتخاب کنید. اگر به انعطاف پذیری بیشتری نیاز دارید، می‌توانید join های ساده در کوئری‌های چند زبانه را انتخاب کنید. راه حل سوم انتخاب مناسبی است.

منبع :

<http://fczaja.blogspot.com/2010/08/multilanguage-database-design.html>

## نظرات خوانندگان

نویسنده: ناصر فرجی  
تاریخ: ۱۶:۱۴ ۱۳۹۲/۰۸/۰۹

روشی که من خودم مدتی استفاده میکردم اضافه کردن یک فیلد language به هر تیبیل بود. موقع ثبت دیتا هر زبانی بود همون زبان رو تو این فیلد نگه میداشتم. مثلا برای فارسی fa و انگلیسی en و ... , موقع نمایش هم بر اساس زبان سایت یک کوئری ساده گرفته می‌شد و اطلاعات زبان مورد نظر لود می‌شد.

نویسنده: محسن موسوی  
تاریخ: ۱۹:۲۹ ۱۳۹۲/۰۸/۰۹

با سلام و تشکر از مطلب خوبتون  
طراحی با یک جدول زبان و نگه داشتن کلید خارجی در جداول مربوطه بهتر میشه  
چندید ساله که از این طراحی استفاده میکنیم و جواب داده.  
یکی از مزایایی که داره میتونی مدیریت سامانه را نسبت به هر زبان بطور مستقل انجام بدی  
و هر جا که نیاز داشتی همزمان چند رکورد را درج کنید.  
ساده و روان.  
البته استراتژی سیستم استفاده از الگوی مناسب رو توجیه میکنه.

نویسنده: محمد پهلوان  
تاریخ: ۱۰:۵۰ ۱۳۹۲/۰۸/۱۰

استفاده از یک فیلد language در هر تیبیل باعث می‌شود شما برای یک موجودیت کالا مثلا در سه زبان مختلف 3 رکورد درج کنید که در ارتباط این کالا با دیگر جداول دچار مشکل خواهید شد

نویسنده: محمد پهلوان  
تاریخ: ۱۱:۳۶ ۱۳۹۲/۰۸/۱۰

روش دوم واقعا روش تمیز و جمع و جوریه اما کوئری هاش واقعا پیچیده اس و انعطاف نداره. به نظر من مخصوصا برای استفاده از EF روش سوم روش مناسبتری باشه. این کوئری‌ها رو با توجه به حجم داده زیاد و سایت پرتراфик بررسی کنید.  
خودم بین روش دو و سه مرددم. از دوستان می‌خوام نظراتشون را با دلائل بیان کنن تا به نتیجه خوبی برسیم

نویسنده: محسن خان  
تاریخ: ۱۱:۵۱ ۱۳۹۲/۰۸/۱۰

مطلب [Globalization در ASP.NET MVC - قسمت ششم](#) در همین راستا مفید است.

نویسنده: بختیاری  
تاریخ: ۱۵:۱۶ ۱۳۹۲/۰۸/۱۰

سلام  
من روش آقای موسوی را منطقی می‌دانم خودم هم با این روش یک سایت طراحی کردم که الان درست و بدون مشکل کار می‌کند

در این مقاله جایگزینی پایاده سازی پیش فرض [ASP.NET Identity](http://ASP.NET Identity) را بررسی می‌کنیم. در ادامه خواهید خواند:

جزئیات نحوه پایاده سازی یک Storage Provider برای ASP.NET Identity

تشریح اینترفیس هایی که باید پایاده سازی شوند، و نحوه استفاده از آنها در ASP.NET Identity

ایجاد یک دیتابیس MySQL روی Windows Azure

نحوه استفاده از یک ابزار کلاینت (MySQL Workbench) برای مدیریت دیتابیس مذکور

نحوه جایگزینی پایاده سازی سفارشی با نسخه پیش فرض در یک اپلیکیشن ASP.NET MVC

در انتهای این مقاله یک اپلیکیشن ASP.NET MVC خواهیم داشت که از ASP.NET Identity و تامین کننده سفارشی جدید استفاده می‌کند. دیتابیس اپلیکیشن MySQL خواهد بود و روی Windows Azure میزبانی می‌شود. سورس کد کامل این مثال را هم می‌توانید از [این لینک](#) دریافت کنید.

### پایاده سازی یک Storage Provider سفارشی برای ASP.NET Identity

ASP.NET Identity سیستم توسعه پذیری است که می‌توانید بخش‌های مختلف آن را جایگزین کنید. در این سیستم بناهای سطح بالایی مانند Managers و Stores وجود دارند.

Managers کلاس‌های سطح بالایی هستند که توسعه دهندگان از آنها برای اجرای عملیات مختلف روی ASP.NET Identity استفاده می‌کنند. مدیریت کننده‌های موجود عبارتند از UserManager و RoleManager. کلاس UserManager برای اجرای عملیات مختلف روی کاربران استفاده می‌شود، مثلاً ایجاد کاربر جدید یا حذف آنها. کلاس RoleManager هم برای اجرای عملیات مختلف روی نقش‌ها استفاده می‌شود.

Stores کلاس‌های سطح پایین‌تری هستند که جزئیات پایاده سازی را در بر می‌گیرند، مثلاً اینکه موجودیت‌های کاربران و نقش‌ها چگونه باید ذخیره و بازیابی شوند. این کلاس‌ها با مکانیزم ذخیره و بازیابی تلفیق شده اند. مثلاً

Microsoft.AspNet.Identity.EntityFramework کلاسی با نام UserStore دارد که برای ذخیره و بازیابی Userها و داده‌های مربوطه توسط EntityFramework استفاده می‌شود.

Managers از Stores تفکیک شده اند و هیچ وابستگی ای به یکدیگر ندارند. این تفکیک بدین منظور انجام شده که بتوانید مکانیزم ذخیره و بازیابی را جایگزین کنید، بدون اینکه اپلیکیشن شما از کار بیافتد یا نیاز به توسعه بیشتر داشته باشد. کلاس‌های Manager می‌توانند با هر Store ای ارتباط برقرار کنند. از آنجا که شما از APIهای سطح بالای UserManager برای انجام عملیات CRUD روی کاربران استفاده می‌کنید، اگر UserStore را با پایاده سازی دیگری جایگزین کنید، مثلاً AzureTable Storage یا MySql، نیازی به بازنویسی اپلیکیشن نیست.

در مثال جاری پایاده سازی پیش فرض Entity Framework را با یک تامین کننده MySQL جایگزین می‌کنیم.

پایاده سازی کلاس‌های Storage

برای پایاده سازی تامین کننده‌های سفارشی، باید کلاس هایی را پایاده سازی کنید که همتای آنها در

Microsoft.AspNet.Identity.EntityFramework وجود دارند:

UserStore<TUser>

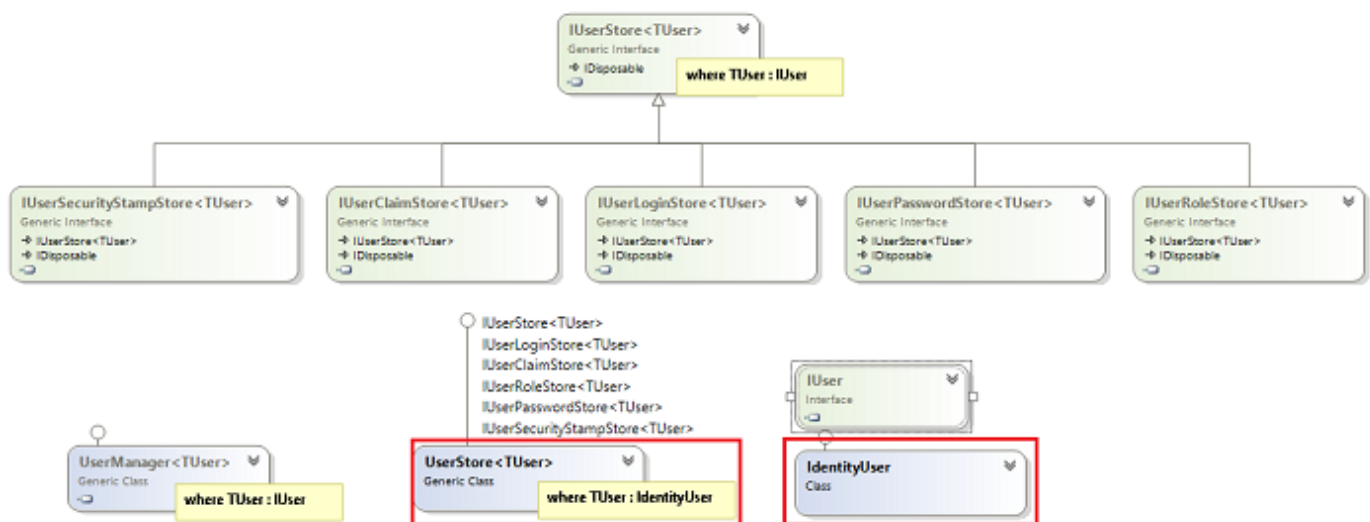
IdentityUser

RoleStore<TRole>

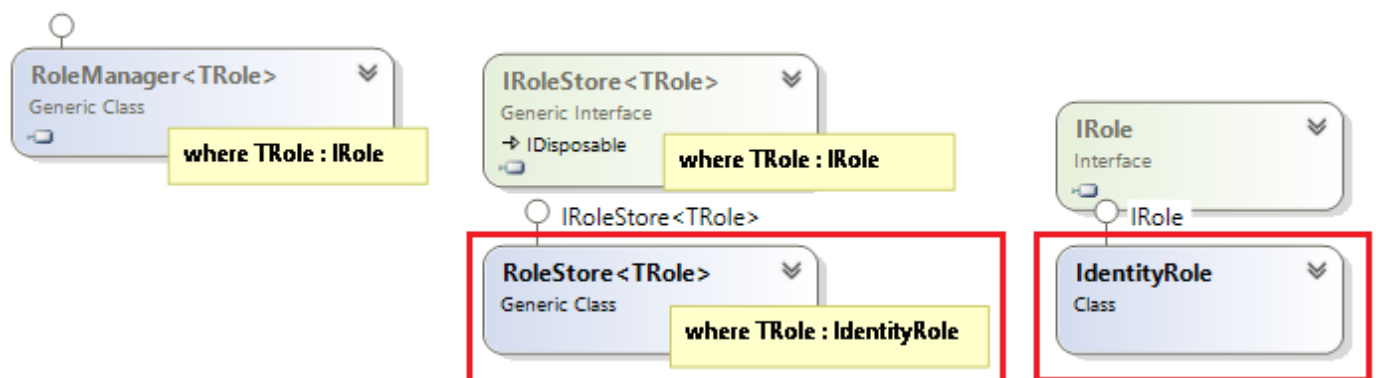
IdentityRole

پایاده سازی پیش فرض Entity Framework را در تصاویر زیر مشاهده می‌کنید.

Users



Roles



در مخزن پیش فرض ASP.NET Identity EntityFramework کلاس‌های بیشتری برای موجودیت‌ها مشاهده می‌کنید.

IdentityUserClaim

IdentityUserLogin

IdentityUserRole

همانطور که از نام این کلاس‌ها مشخص است، اختیارات، نقش‌ها و اطلاعات ورود کاربران توسط این کلاس‌ها معرفی می‌شوند. در مثال جاری این کلاس‌ها را پیاده سازی نخواهیم کرد، چرا که بارگذاری اینگونه رکوردها از دیتابیس به حافظه برای انجام عملیات پایه (مانند افزودن و حذف اختیارات کاربران) سنگین است. در عوض کلاس‌های backend store اینگونه عملیات را بصورت مستقیم روی دیتابیس اجرا خواهند کرد. بعنوان نمونه متد `UserStore.GetClaimsAsync()` را در نظر بگیرید. این متد به نوبه خود متد `userClaimTable.FindByUserId(user.Id)` را فراخوانی می‌کند که یک کوئری روی جدول مربوطه اجرا می‌کند و لیستی از اختیارات کاربر را بر می‌گرداند.

```
public Task<IList<Claim>> GetClaimsAsync(IdentityUser user)
{
    ClaimsIdentity identity = userClaimsTable.FindByUserId(user.Id);
    return Task.FromResult<IList<Claim>>(identity.Claims.ToList());
}
```

```
}
```

برای پیاده سازی یک تامین کننده سفارشی MySQL مراحل زیر را دنبال کنید.  
1. کلاس کاربر را ایجاد کنید، که اینترفیس **IUser** را پیاده سازی می کند.

```
public class IdentityUser : IUser
{
    public IdentityUser(){...}
    public IdentityUser(string userName) (){...}
    public string Id { get; set; }
    public string Username { get; set; }
    public string PasswordHash { get; set; }
    public string SecurityStamp { get; set; }
}
```

2. کلاس User Store را ایجاد کنید، که اینترفیس های **IUserStore** , **IUserClaimStore** , **IUserLoginStore** , **IUserRoleStore** و **IUserPasswordStore** را پیاده سازی می کند. توجه کنید که تنها اینترفیس **IUserStore** را باید پیاده سازی کنید، مگر آنکه بخواهید از امکاناتی که دیگر اینترفیس ها ارائه می کنند هم استفاده کنید.

```
public class UserStore : IUserStore<IdentityUser>,
                        IUserClaimStore<IdentityUser>,
                        IUserLoginStore<IdentityUser>,
                        IUserRoleStore<IdentityUser>,
                        IUserPasswordStore<IdentityUser>
{
    public UserStore(){...}
    public Task CreateAsync(IdentityUser user){...}
    public Task<IdentityUser> FindByIdAsync(string userId){...}
    ...
}
```

3. کلاس Role را ایجاد کنید که اینترفیس **IRole** را پیاده سازی می کند.

```
public class IdentityRole : IRole
{
    public IdentityRole(){...}
    public IdentityRole(string roleName) (){...}
    public string Id { get; set; }
    public string Name { get; set; }
}
```

4. کلاس Role Store را ایجاد کنید که اینترفیس **IRoleStore** را پیاده سازی می کند. توجه داشته باشید که پیاده سازی این مخزن اختیاری است و در صورتی لازم است که بخواهید از نقش ها در سیستم خود استفاده کنید.

```
public class RoleStore : IRoleStore<IdentityRole>
{
    public RoleStore(){...}
    public Task CreateAsync(IdentityRole role){...}
    public Task<IdentityRole> FindByIdAsync(string roleId){...}
    ....
}
```

کلاس های بیشتری هم وجود دارند که مختص پیاده سازی مثال جاری هستند.

**MySQLDatabase:** این کلاس اتصال دیتابیس MySQL و کوئری‌ها را کپسوله می‌کند. کلاس‌های UserStore و RoleStore توسط نمونه ای از این کلاس و هله سازی می‌شوند.

**RoleTable:** این کلاس جدول Roles و عملیات CRUD مربوط به آن را کپسوله می‌کند.

**UserClaimsTable:** این کلاس جدول UserClaims و عملیات CRUD مربوط به آن را کپسوله می‌کند.

**UserLoginsTable:** این کلاس جدول UserLogins و عملیات CRUD مربوط به آن را کپسوله می‌کند.

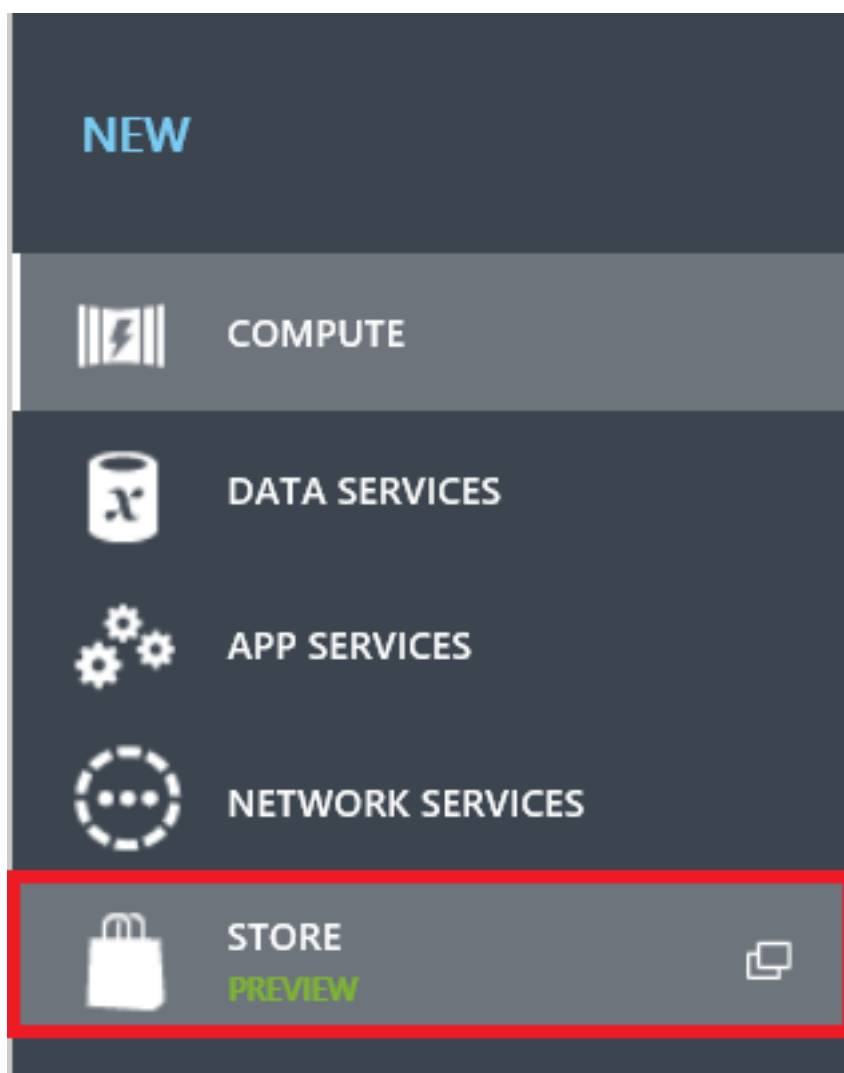
**UserRolesTable:** این کلاس جدول UserRoles و عملیات CRUD مربوطه به آن را کپسوله می‌کند.

**UserTable:** این کلاس جدول Users و عملیات CRUD مربوط به آن را کپسوله می‌کند.

ایجاد یک دیتابیس MySQL روی Windows Azure

1. به [پورتال مدیریتی Windows Azure](#) وارد شوید.

2. در پایین صفحه روی **+NEW** کلیک کنید و گزینه **STORE** را انتخاب نمایید.



در ویزارد **Choose Add-on** به سمت پایین اسکرول کنید و گزینه **ClearDB MySQL Database** را انتخاب کنید. سپس به مرحله بعد بروید.

# Choose an Add-on

ALL

APP SERVICES

DATA



ClearDB MySQL Database



ClearPointe Azure Management



cloudinary



D&B Business Insight





Embarke Email Analytics



Engine Yard Platform as a Service








## ClearDB MySQL Database

SuccessBricks, Inc.

ClearDB is a powerful, fault-tolerant database-as-a-service in the cloud for your MySQL powered applications.

PUBLISHED DATE

10/10/2012



2

3

4. راهکار Free بصورت پیش فرض انتخاب شده، همین گزینه را انتخاب کنید و نام دیتابیس را به IdentityMySQLDatabase تغییر دهید. نزدیک ترین ناحیه (region) به خود را انتخاب کنید و به مرحله بعد بروید.

PURCHASE FROM STORE

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

✓

REGION

▼

5. روی علامت checkmark کلیک کنید تا دیتابیس شما ایجاد شود. پس از آنکه دیتابیس شما ساخته شد می توانید از قسمت ADD-

The screenshot shows the Azure portal interface. On the left sidebar, the 'ADD-ONS' menu item is highlighted with a red box. The main content area displays a table of add-ons. The 'IdentityMySQLDatabase' add-on is highlighted with a red box. The 'CONNECTION INFO' button in the bottom right corner is also highlighted with a red box.

NAME	TYPE
IdentityMySQLDatabase	App Service

6. همانطور که در تصویر بالا می بینید، می توانید اطلاعات اتصال دیتابیس (connection info) را از پایین صفحه دریافت کنید.


7. اطلاعات اتصال را با کلیک کردن روی دکمه مجاور کپی کنید تا بعداً در اپلیکیشن MVC خود از آن استفاده کنیم.

×

## Connection info


**CONNECTIONSTRING**  


Database=IdentityMySQLDatabase;Data Source=us-cdb-azure-west-b.cleardb.com;User Id=root@us-cdb-azure-west-b.cleardb.com;Password=us-cdb-azure-west-b.cleardb.com;Persist Security Info=True;Server=us-cdb-azure-west-b.cleardb.com;SslMode=VerifyCA



**CONNECTIONURL**  

mysql://bc554a1b496531:a7277b9f@us-cdb-azure-west-b.cleardb.com:3306/IdentityMySQLDatabase





### ایجاد جداول ASP.NET Identity در یک دیتابیس MySQL

ابتدا ابزار MySQL Workbench را نصب کنید.

1. ابزار مذکور را [از اینجا](#) دانلود کنید.

2. هنگام نصب، گزینه **Setup Type: Custom** را انتخاب کنید.

3. در قسمت انتخاب قابلیت ها، گزینه های **MySQLWorkbench** و **Applications** را انتخاب کنید و مراحل نصب را به اتمام برسانید.

4. اپلیکیشن را اجرا کرده و روی **MySQLConnection** کلیک کنید تا رشته اتصال جدیدی تعریف کنید. رشته اتصالی که در مراحل

قبل از Azure MySQL Database کپی کردید را اینجا استفاده کنید. بعنوان مثال:

**Connection Name**

: AzureDB;

**Host Name**

: us-cdb-azure-west-b.cleardb.com;

Username

: <username>;

Password

: <password>;

Default Schema

: IdentityMySQLDatabase

5. پس از برقراری ارتباط با دیتابیس، یک برگ **Query** جدید باز کنید. فرامین زیر را برای ایجاد جداول مورد نیاز کپی کنید.

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

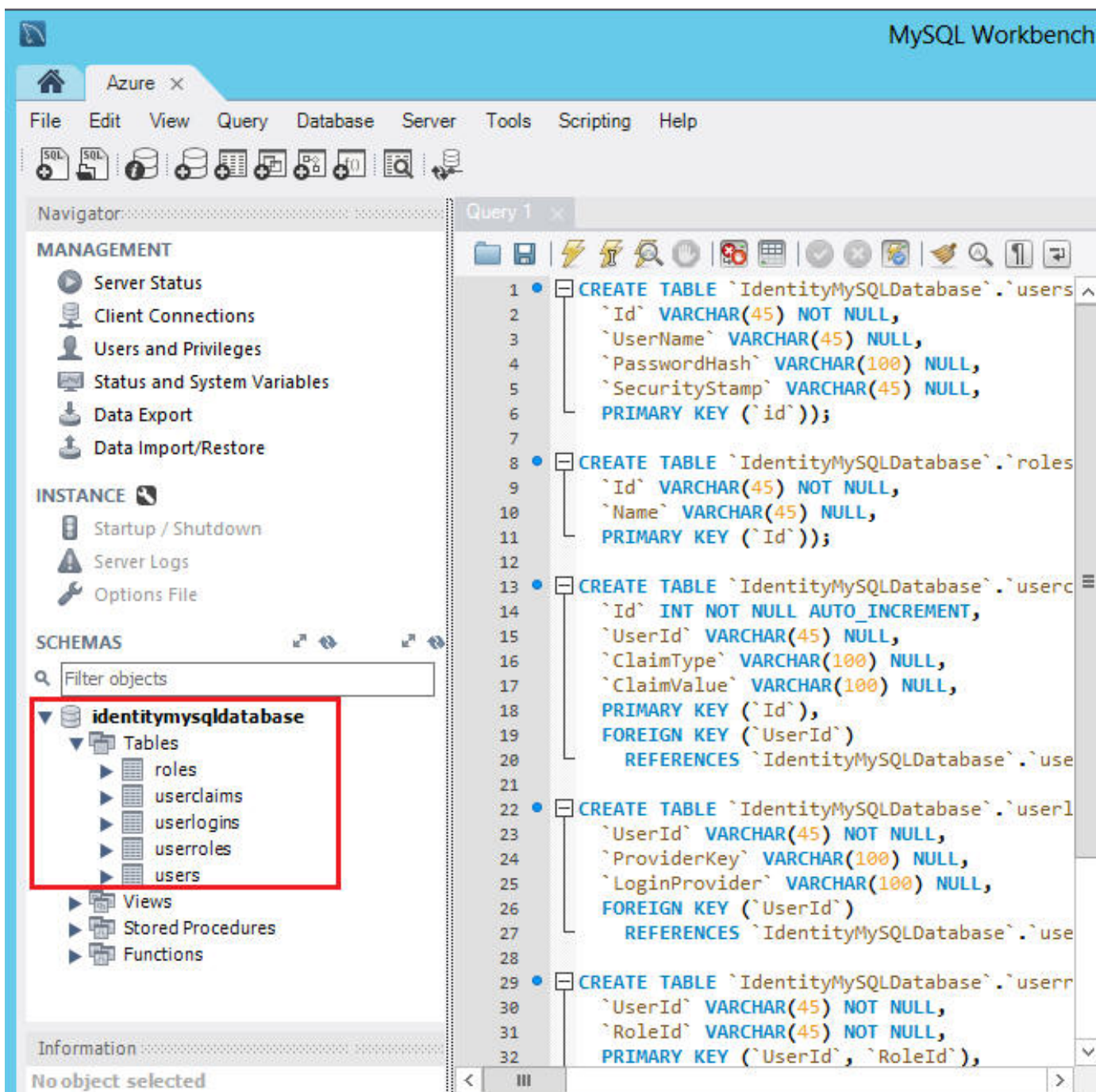
CREATE TABLE `IdentityMySQLDatabase`.`roles` (
  `Id` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NULL,
  PRIMARY KEY (`Id`));

CREATE TABLE `IdentityMySQLDatabase`.`userclaims` (
  `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`.`users` (`Id`) on delete cascade);

CREATE TABLE `IdentityMySQLDatabase`.`userlogins` (
  `UserId` VARCHAR(45) NOT NULL,
  `ProviderKey` VARCHAR(100) NULL,
  `LoginProvider` VARCHAR(100) NULL,
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users` (`Id`) on delete cascade);

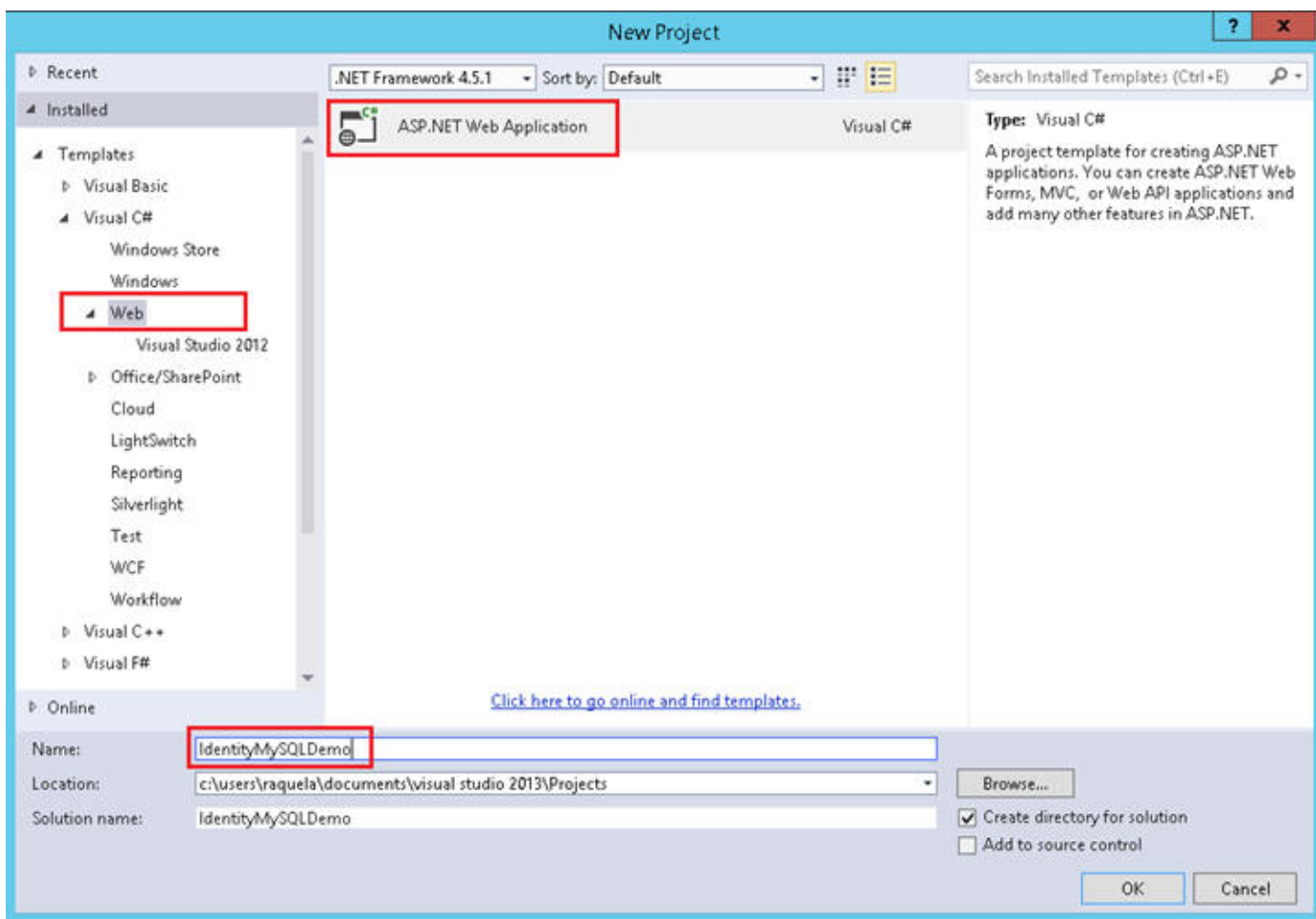
CREATE TABLE `IdentityMySQLDatabase`.`userroles` (
  `UserId` VARCHAR(45) NOT NULL,
  `RoleId` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserId`, `RoleId`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users` (`Id`)
    on delete cascade
    on update cascade,
  FOREIGN KEY (`RoleId`)
    REFERENCES `IdentityMySQLDatabase`.`roles` (`Id`)
    on delete cascade
    on update cascade);
```

6. حالا تمام جداول لازم برای ASP.NET Identity را در اختیار دارید، دیتابیس ما MySQL است و روی Windows Azure میزبانی شده.

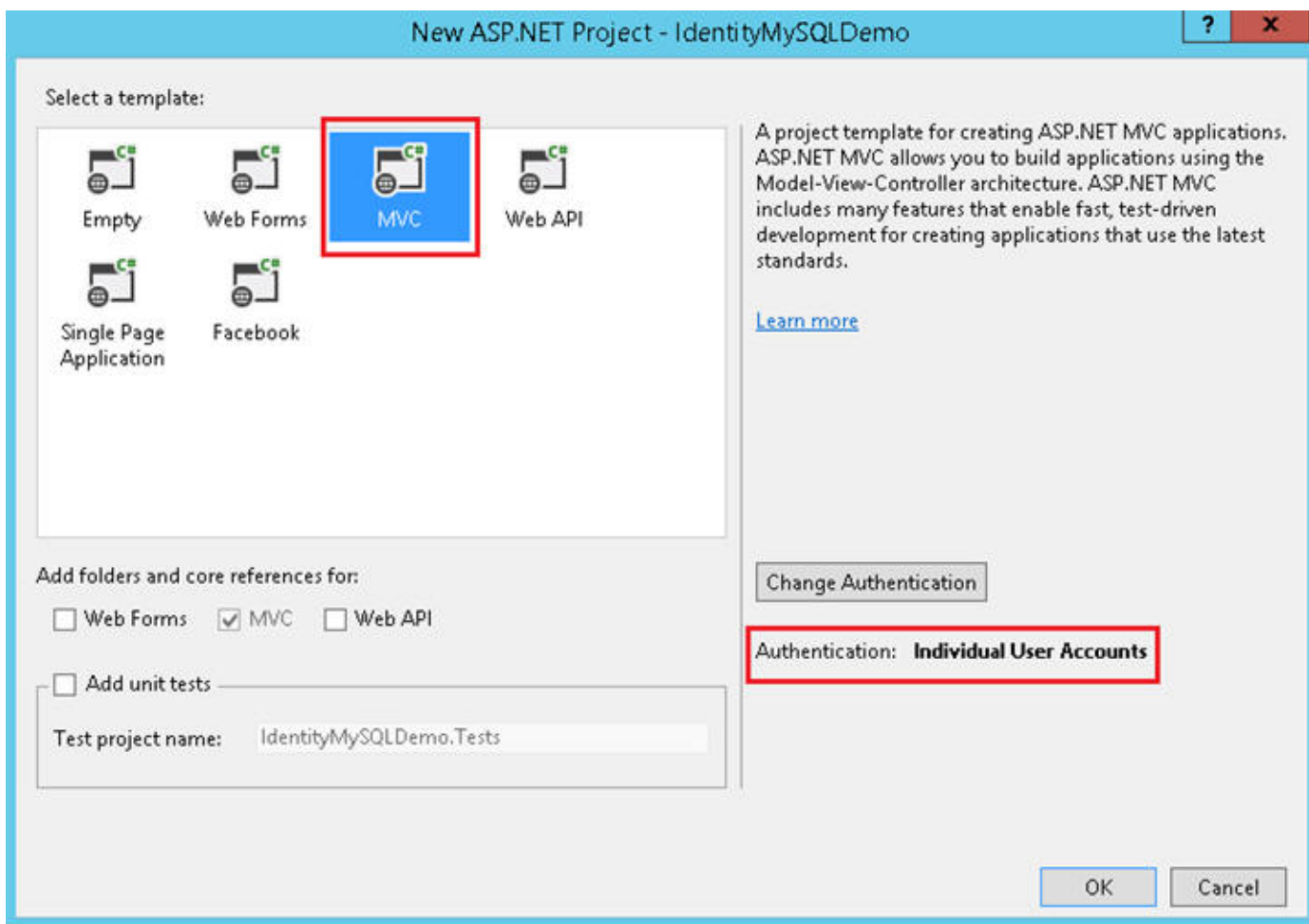


ایجاد یک اپلیکیشن ASP.NET MVC و پیگر بندی آن برای استفاده از MySQL Provider

1. به مخزن <https://github.com/raquelsa/AspNet.Identity.MySQL> بروید.
2. در گوشه سمت راست پایین صفحه روی دکمه Download Zip کلیک کنید تا کل پروژه را دریافت کنید.
3. محتوای فایل دریافتی را در یک پوشه محلی استخراج کنید.
4. پروژه AspNet.Identity.MySQL را باز کرده و آن را کامپایل (build) کنید.
5. روی نام پروژه کلیک راست کنید و گزینه Add, New Project را انتخاب نمایید. پروژه جدیدی از نوع ASP.NET Web Application بسازید و نام آن را به IdentityMySQLDemo تغییر دهید.



6. در پنجره New ASP.NET Project قالب MVC را انتخاب کنید و تنظیمات پیش فرض را بپذیرید.



7. در پنجره Solution Explorer روی پروژه IdentityMySQLDemo کلیک راست کرده و **Manage NuGet Packages** را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Identity.EntityFramework" را وارد کنید. در لیست نتایج این پکیج را انتخاب کرده و آن را حذف (Uninstall) کنید. پیغامی مبنی بر حذف وابستگی‌ها باید دریافت کنید که مربوط به پکیج EntityFramework است، گزینه Yes را انتخاب کنید. از آنجا که کاری با پیاده سازی فرض نخواهیم داشت، این پکیج‌ها را حذف می‌کنیم.

8. روی پروژه IdentityMySQLDemo کلیک راست کرده و **Add, Reference, Solution, Projects** را انتخاب کنید. در دیالوگ باز شده پروژه **AspNet.Identity.MySQL** را انتخاب کرده و OK کنید.

9. در پروژه IdentityMySQLDemo پوشه Models را پیدا کرده و کلاس **IdentityModels.cs** را حذف کنید.

10. در پروژه IdentityMySQLDemo تمام ارجاعات "using Microsoft.AspNet.Identity.EntityFramework;" را با "using;" جایگزین کنید.

11. در پروژه IdentityMySQLDemo تمام ارجاعات به کلاس "ApplicationUser" را با "IdentityUser" جایگزین کنید.

12. کنترلر Account را باز کنید و متد سازنده آنرا مطابق لیست زیر تغییر دهید.

```
public AccountController() : this(new UserManager<IdentityUser>(new UserStore(new MySQLDatabase())))
{
}
```



13. فایل web.config را باز کنید و رشته اتصال DefaultConnection را مطابق لیست زیر تغییر دهید.

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

مقادیر <UserId>, <DataSource> و <Password> را با اطلاعات دیتابیس خود جایگزین کنید.

#### اجرای اپلیکیشن و اتصال به دیتابیس MySQL

1. روی پروژه IdentityMySQLDemo کلیک راست کرده و **Set as Startup Project** را انتخاب کنید.
2. اپلیکیشن را با Ctrl + F5 کامپایل و اجرا کنید.
3. در بالای صفحه روی **Register** کلیک کنید.
4. حساب کاربری جدیدی بسازید.

[Application name](#) [Home](#) [About](#) [Contact](#)

## Register.

Create a new account.

---

User name

Password

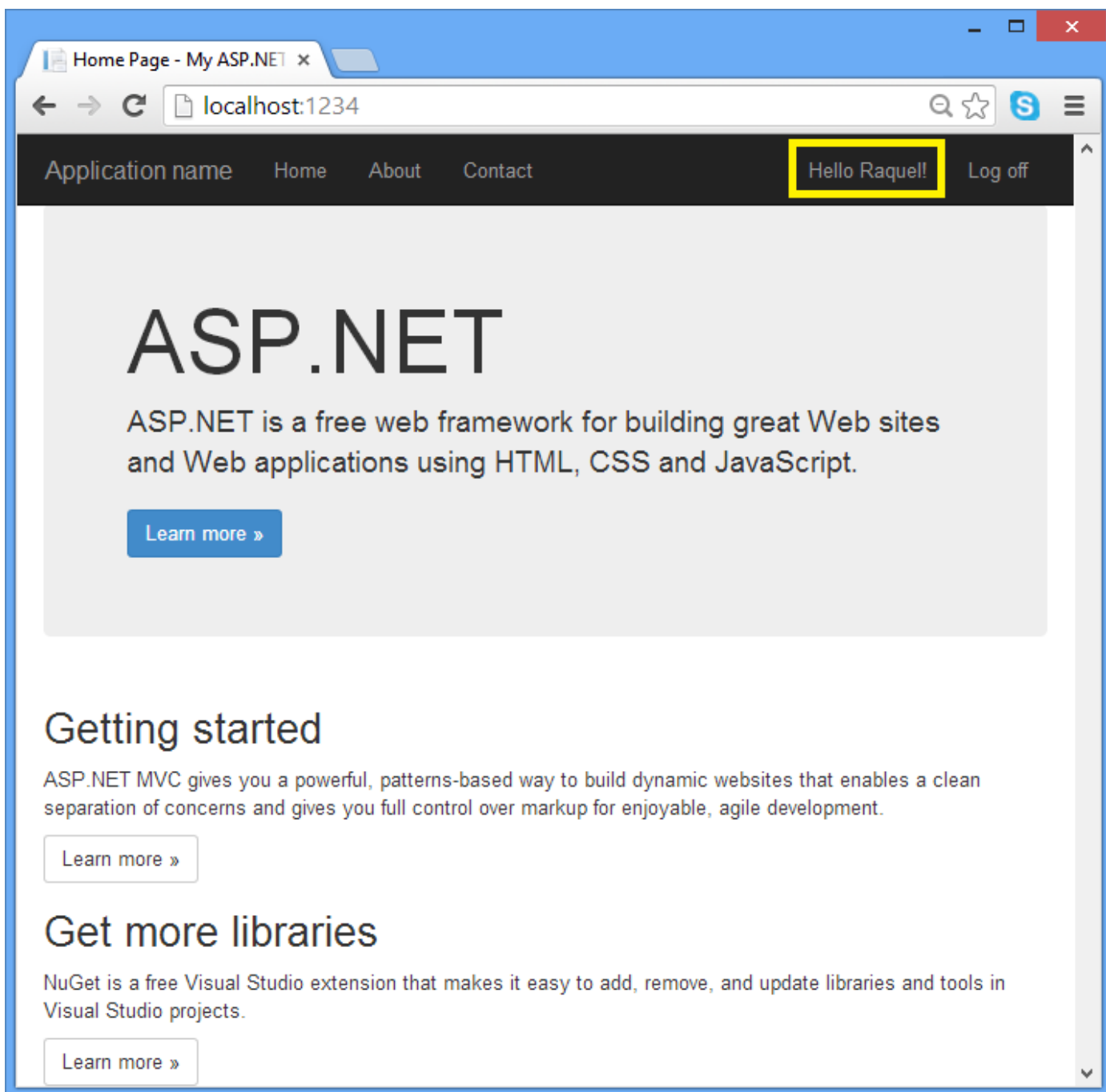
Confirm password

Register

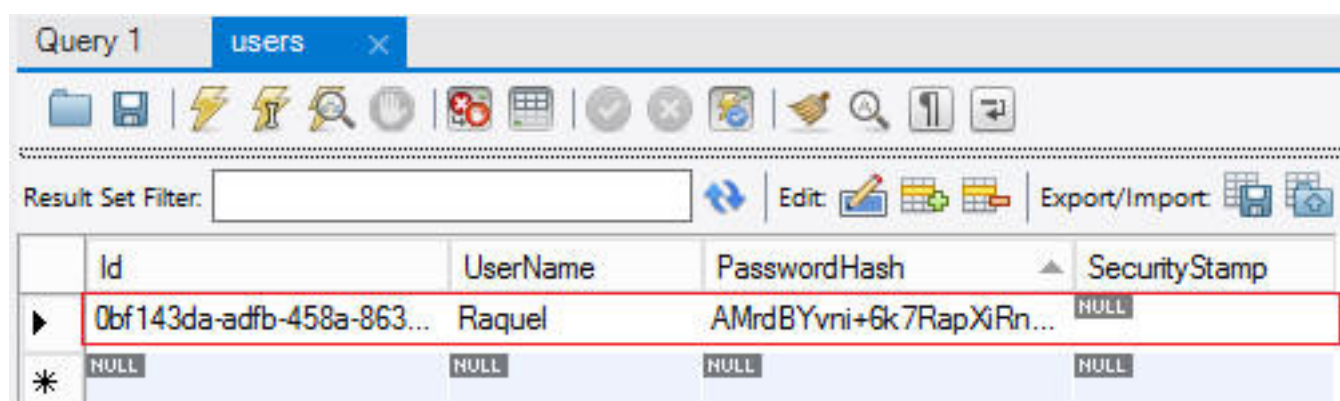
---

© 2013 - My ASP.NET Application

5. در این مرحله کاربر جدید باید ایجاد شده و وارد سایت شود.



6. به ابزار MySQL Workbench بروید و محتوای جداول **IdentityMySQLDatabase** را بررسی کنید. جدول **users** را باز کنید و اطلاعات کاربر جدید را بررسی نمایید.



	Id	UserName	PasswordHash	SecurityStamp
▶	0bf143da-adfb-458a-863...	Raquel	AMrdBYvni+6k7RapXiRn...	NULL
*	NULL	NULL	NULL	NULL

برای ساده نگاه داشتن این مقاله از بررسی تمام کدهای لازم خودداری شده، اما اگر مراحل را دنبال کنید و سورس کد نمونه را دریافت و بررسی کنید خواهید دید که پیاده سازی تامین کنندگان سفارشی برای ASP.NET Identity کار نسبتاً ساده ای است.

## مثال ساده پرداخت بانکی با استفاده از تراکنش و پروسیجر در مای اس کیو ال

عنوان:

ناصر نیازی

نویسنده:

۱۴:۲۵ ۱۳۹۳/۱۰/۳۰

تاریخ:

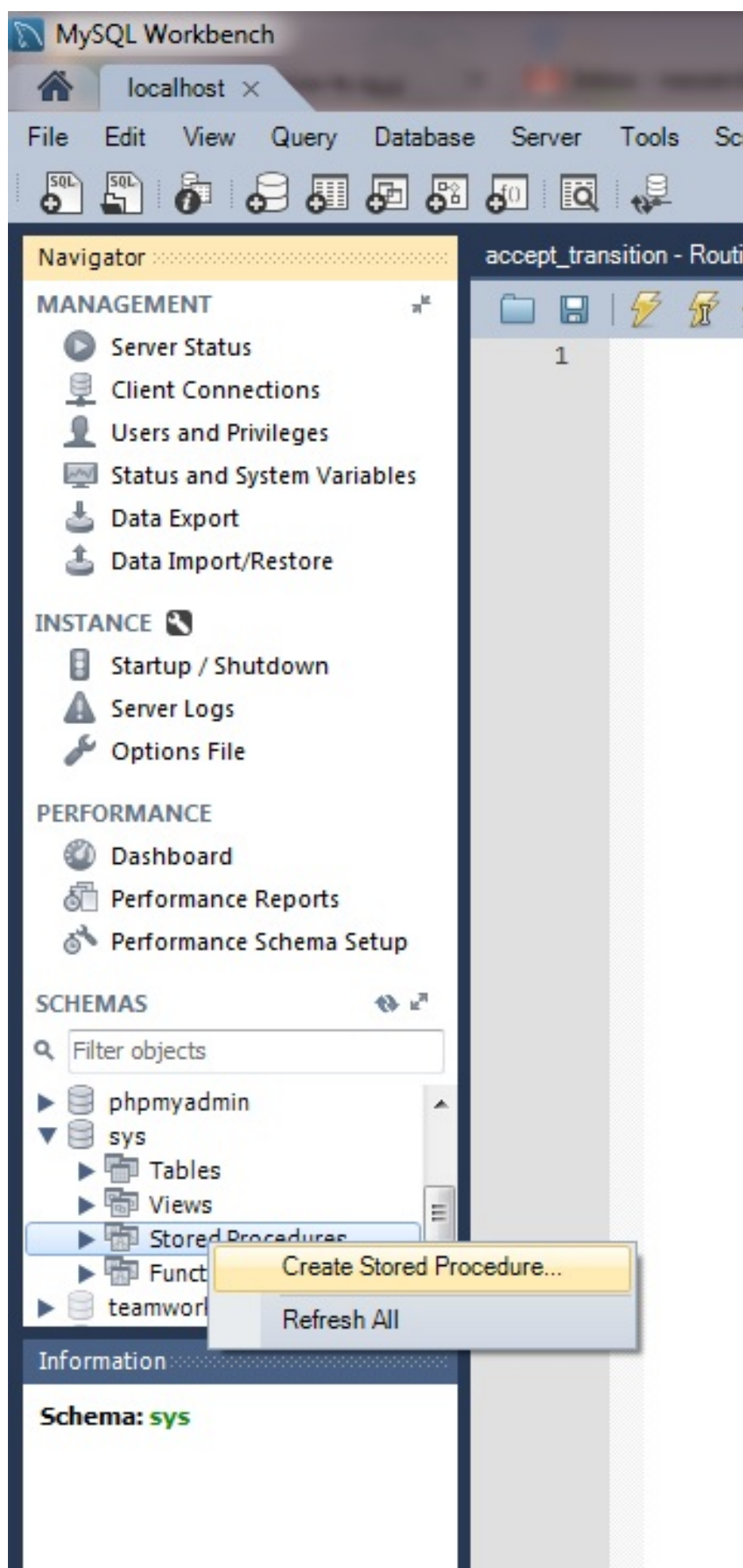
[www.dotnettips.info](http://www.dotnettips.info)

آدرس:

MySQL, transaction, Stored procedure

گروه‌ها:

برای انجام عملیاتی مثل عملیات حسابداری، نیاز به انجام پی در پی چندین دستور می‌باشد و در صورت انجام نشدن یکی از آنها، بقیه نیز نامعتبر خواهند بود که برای پیاده سازی این مکانیزم از تراکنش‌ها در بانک اطلاعاتی استفاده می‌شود. تراکنش‌ها معمولاً در بدنه‌ی توابع ذخیره شده روی بانک (stored procedure) پیاده سازی می‌شوند. برای تعریف یک پروسیجر در مای اس کیو ال من از برنامه‌ی MySQL Workbench به شکل زیر استفاده می‌کنم. البته می‌توان دستور ایجاد تابع را از روش‌های دیگر هم اجرا کرد.



در مای اس کیو ال برای تعریف یک تابع از ساختار زیر استفاده می‌کنیم :

```
DELIMITER $$

CREATE
    DEFINER=`user_name`@`host_name`|CURRENT_USER
    PROCEDURE `transition_name`(
        IN | OUT | INOUT `parameter_name` type(bigint,int , ...)
    )
    SQL SECURITY DEFINER| INVOKER
transition_name: BEGIN
#----procedure_body
END
```

نکات مربوط به تعریف :  
در قسمت

```
DEFINER=`user_name`@`host_name`|CURRENT_USER
```

کسی که تابع را تعریف کرده معرفی می‌شود. اگر شما برای انتقال دیتابیس از جایی به جای دیگر، از روش ایمپورت و اکسپورت استفاده کنید، اگر نام کاربری بانک شما متفاوت باشد، معمولاً این قسمت باعث خطا می‌شود؛ چون شما نمی‌توانید به نام فرد دیگری تابع بسازید. پیش فرض هم مقدار

```
CURRENT_USER
```

در نظر گرفته می‌شود که همان اسم کاربری و هاست شما است.  
نکته بعدی : قسمت

```
SQL SECURITY DEFINER| INVOKER
```

است که استفاده کننده از پروسیجر را مشخص می‌کند. مقدار DEFINER یعنی فقط تعریف کننده حق استفاده از این پروسیجر را دارد و مقدار INVOKER یعنی هر کسی حق استفاده از این تابع را دارد .  
برای شرح تراکنش، مثال پرداخت بانکی را شرح می‌دهیم:

```
DELIMITER $$

CREATE
    DEFINER=CURRENT_USER
    PROCEDURE `transition_pay`(
        #-----input value
        IN `pay_value` bigint,
        IN `admin_id` int,
        #-----result code
        OUT `result` bigint
    )
    SQL SECURITY INVOKER
transition_pay: BEGIN
DECLARE admin_credit DOUBLE DEFAULT 0;
SELECT `Credit`
INTO admin_credit
FROM `Admin`

WHERE `Admin_id` = admin_id
#----- transaction body
END
```

در قسمت بالا متغیری را تعریف کرده و آخرین میزان اعتبار ادمین را داخل آن قرار دادیم تا در قسمت تراکنش، مقدار پرداختی را به آن اضافه کنیم و دو باره ادمین را آپدیت کنیم. اگر بخواهیم به دلیلی قبل از رسیدن به تراکنش آن را کنسل کنیم، می‌توان از دستور LEAVE استفاده کرد: مثال :

```
IF admin_id=0 THEN
set result = -1 ;
#exit procedure
LEAVE transition_pay;
END IF;
```

حال شروع تراکنش حالت ساده :

```
START TRANSACTION;
INSERT INTO
`PayBalance` (`Value` , `Admin_id` )
VALUES (pay_value, admin_id);

UPDATE `Admin`
SET `Credit`=admin_credit + pay_value
WHERE `admin_id`=admin_id;
COMMIT;
```

با پایان تراکنش، تمام مقادیر به درستی در بانک ذخیره می‌گردند. حال اگر بخواهیم به دلیلی داخل تراکنش آن را لغو کنیم از دستور ROLLBACK استفاده می‌کنیم. مثال:

```
IF pay_value=0 THEN
set result = -1 ;
#roolback procedure
ROLLBACK ;
END IF;
```

برای اطمینان از اجرا شدن دستورات در مای اس کیو ال می‌توان از

```
SET autocommit = {0 | 1}
```

نیز استفاده کرد که مقدار پیش فرض آن یک است. یعنی هر دستوری بلافاصله اجرا شود. می‌توان قبل از دستوراتی که می‌خواهیم پی در پی اجرا شوند، یک بار آن را صفر و بعد از اجرای دستورات آنرا یک کنیم. نکته آخر اینکه با استفاده از زبان پی اچ پی هم می‌توان تراکنشی را شروع و تمام کرد و بین این دو دستورات مورد نظر را نوشت و همیشه وجود پروسیجر الزامی نیست.