



Взаимодействие с данными. Подходы

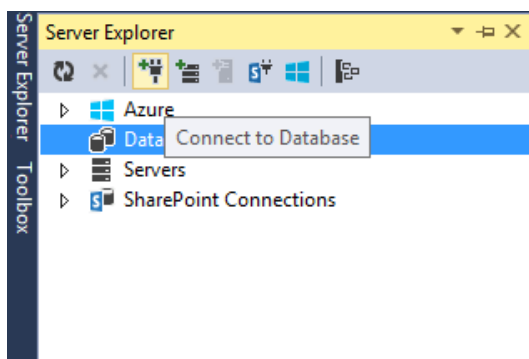
Code First к существующей базе данных

Последнее обновление: 31.10.2015



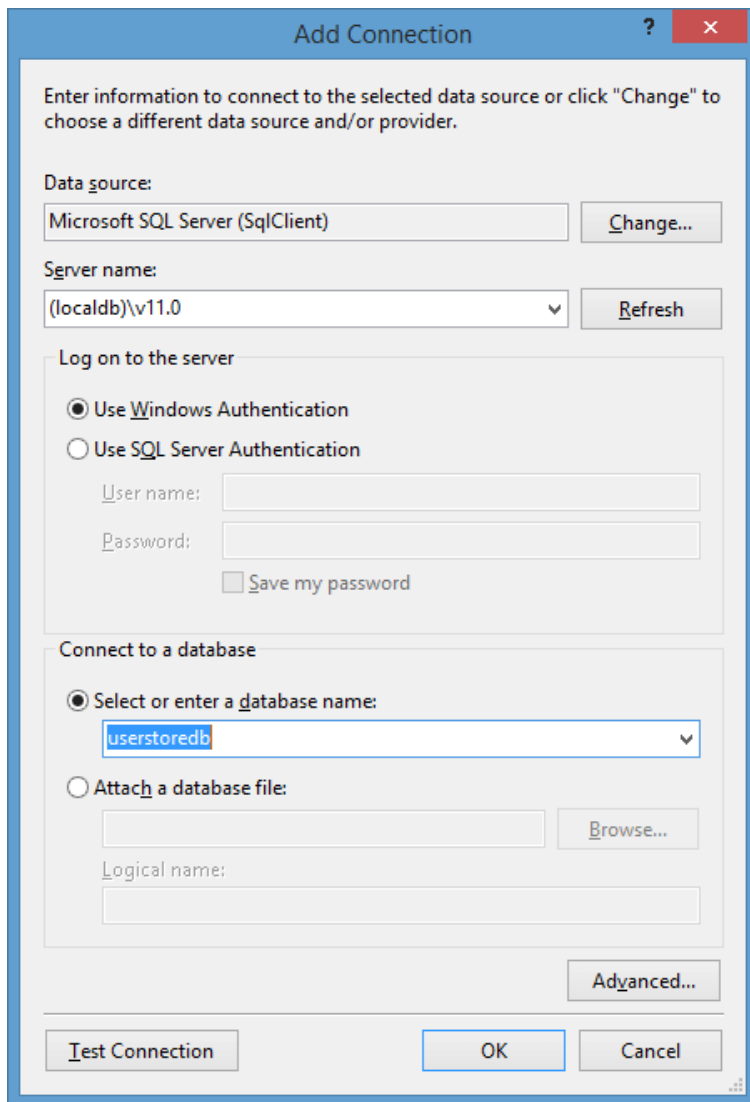
В первой главе при создании первого приложения с Entity Framework мы использовали подход **Code First**. Этот подход очень прост и удобен. Но он также и очень гибкий. Так, вполне часто распространена ситуация, когда база данных уже имеется. И здесь опять же поможет Code First. Иногда программисты называют данный подход **Code Second**. Посмотрим на примере.

Вначале создадим новый проект. Затем создадим тестовую базу данных. В Visual Studio выберем в меню пункт **View->Server Explorer**. В открывшемся окне Server Explorer подключимся к новой базе данных, выбрав Connect to Database.



В окне создания подключения в качестве сервера выберем **(localdb)\v11.0**. В данном случае мы выбираем движок localdb для работы с MS SQL Server, который предназначен специально для целей разработки. Хотя мы также могли выбрать, как в предыдущей теме полноценный SQL Server Express.

А в качестве имени базы данных введем userstoredb:



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
(localdb)\v11.0 Refresh

Log on to the server

☒ Use Windows Authentication
☐ Use SQL Server Authentication

User name:
Password:
☐ Save my password

Connect to a database

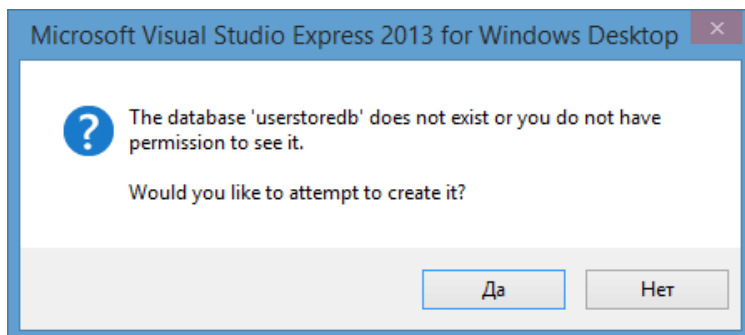
☒ Select or enter a database name:
userstoredb

☐ Attach a database file:
 Browse...
Logical name:

Advanced...

Test Connection OK Cancel

И если база данных не существует, нам отобразится окно с подтверждением ее создания:

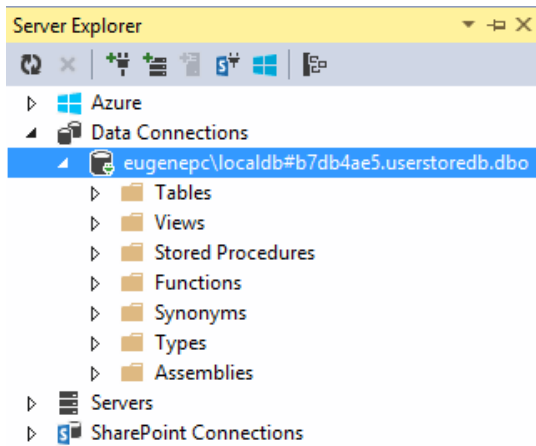


Microsoft Visual Studio Express 2013 for Windows Desktop

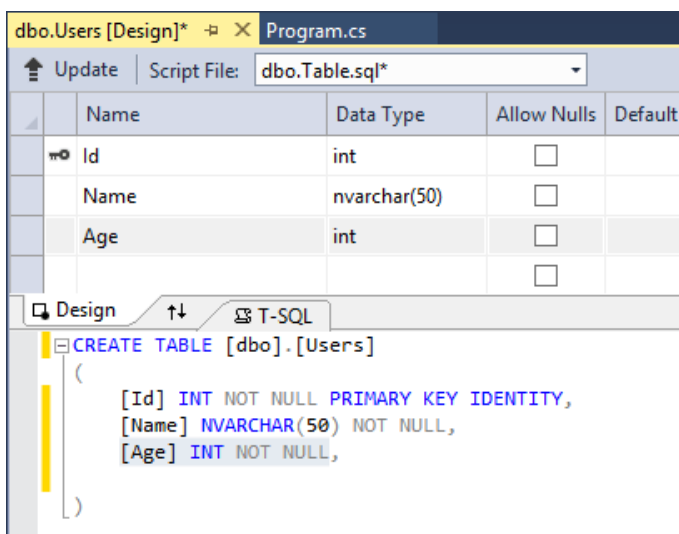
? The database 'userstoredb' does not exist or you do not have permission to see it.
Would you like to attempt to create it?

Да Нет

Нажмем 'Да'. И после этого в окне Server Explorer отобразится созданная база данных:

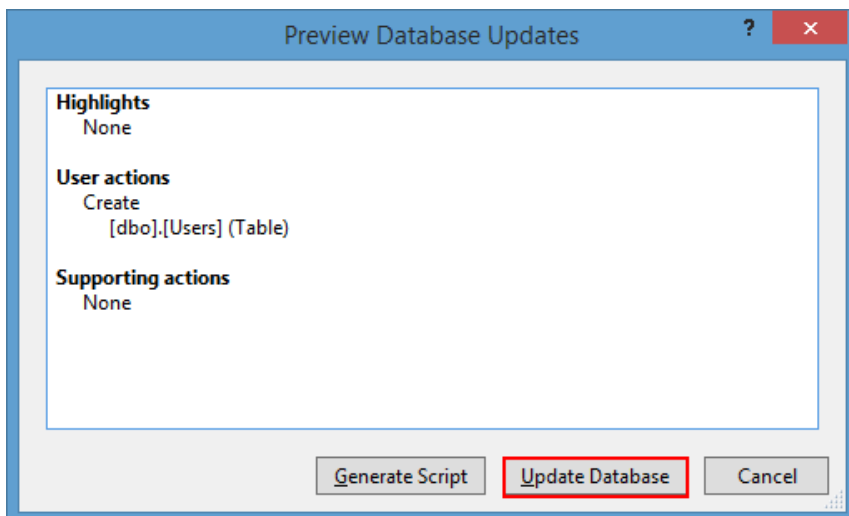


Она еще пуста, поэтому добавим в нее таблицу. Наждем правой кнопкой мыши на узел Tables и в появившемся контекстном меню выберем Add New Table. Затем в центральном поле в режиме дизайнера создадим следующее определение таблицы:



В поле T-SQL (или графически) определим структуру и имя таблицы, типы столбцов и после этого нажмем в верхнем левом углу на кнопку Update.

В новом окне нам будет выдана некоторая информация об изменениях, производимых в бд:



Нажмем на кнопку Update Database. И после этого будет создана таблица Users. Обновив окно Server Explorer и открыв узел Tables, вы сможете увидеть новую таблицу Users.

Мы можем добавить некоторые данные в таблицу. Для этого нажмем на таблицу в окне Server Explorer правой кнопкой мыши и выберем пункт **Show Table Data** (Показать данные таблицы). У нас откроется форма для работы с данными, в

которую введем пару строк:

dbo.Users [Data] Program.cs			
Max Rows: 1000			
	Id	Name	Age
▶	1	Tom	33
	2	Sam	24
*	NULL	NULL	NULL

База данных готова. Теперь нам надо добавить подключение в файл конфигурации приложения. В Solution Explorer найдем файл App.config и откроем его. Перед закрывающим тегом </configuration> добавим новую секцию

connectionStrings:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <!--остальное содержимое-->
  <connectionStrings>
    <add name="UserDB" connectionString="data source=(localdb)\v11.0;Initial
Catalog=userstoredb;Integrated Security=True;"
    providerName="System.Data.SqlClient"/>
  </connectionStrings>
</configuration>
```

Теперь определим классы модели данных и контекста. Добавим класс модели User:

```
public class User
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
}
```

И также добавим класс контекста данных:

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

namespace CodeSecond
{
    class UserContext : DbContext
    {
        public UserContext():
            base("UserDB")
        { }

        public DbSet<User> Users { get; set; }
    }
}
```

В конструкторе контекста данных мы передаем в конструктор базового класса имя строки подключения из файла конфигурации App.config. Так как мы определили там строку подключения UserDB (<add name="UserDB"/>), то именно это значение и используется в конструкторе.

Однако, как вариант, мы могли не использовать конструктор в классе контекста данных, а определить в качестве имени строки подключения название этого класса, например: <add name="UserContext" connectionString="....

И для получения данных определим следующий код в консольном приложении:

```
using(UserContext db = new UserContext())
{
    var users = db.Users;
    foreach(User u in users)
```

```
{  
    Console.WriteLine("{0}.{1} - {2}", u.Id, u.Name, u.Age);  
}  
}
```

[Назад](#) [Содержание](#) [Вперед](#)



[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2017. Все права защищены.