

Упражнение №4 по ПС - Част 2

Entity Framework

*Microsoft SQL Server
ADO.Net
Entity Framework
CRUD операции с Entity Framework*

Целта на това упражнение:

Разглеждани на различни методи за свързване на приложение с база от данни и работа с нея.

Необходими познания:

От предметите дотук и предното упражнение трябва да можете:

- Да може да сглобите основно визуално приложение
- Да може да опишете основен интерфейс със XAML
- Да познавате основните контроли в WPF
- Да извеждате данни на интерфейса чрез свързване

Легенда:

Най-важното

- Схематично и накратко

Обяснение. Уточнение.

Пример: или **Задача:** оградено трябва да изпълните за да е синхронизиран проекта ви

- Подходящо място да стартирате изпълнение за да проверите дали работи вярно.

Задачите в упражнението изграждат:

Университетска информационна система.

В това упражнение: Преправяме съществуващите проекти свързани с университетската информационна система да работят с база данни:

- създаваме база от данни в Microsoft SQL Server
- създаваме Connection String настройка за връзка с базата
- (-изпълняваме SQL заявки в кода с ADO.NET)
- преправяме съществуващите Student и User класове във формат за Entity класове (Code first)
- създаваме контекстни класове
- създаваме функции, които копират тестовите обекти в базата
- преправяме приложението да работи изцяло с базата данни

В края на упражнението:

Университетска информационна система чете данни от и работи с база данни

За синхронизация на упражненията:

Добавете в проекта **StudentRepository**.

- На страницата/прозореца с контроли за извеждане на данни за студент:
 - Контролите за извеждане на данни за студент да са свързани към един обект от тип студент.

1. Microsoft SQL Server

Microsoft SQL Server е предназначен да поддържа големи бази данни в различни области, включително такива свързани с обработка на онлайн транзакции (OLTP), съхранение на данни и електронна търговия. За тези функции SQL Server осигурява набор от инструменти, като инструменти от команден ред (например bcp.exe, който копира данни между SQL Server и файлове на операционната система), и Management Studio (сложен графичен инструмент за управление на множество бази данни на SQL Server). Ще използваме предимно Management Studio.

Microsoft SQL Server има безплатна Express версия.

SQL Server Management Studio е безплатно и включено в инсталационния пакет на Microsoft SQL Server Express.

- За да продължите с упражнението трябва да инсталирате Microsoft SQL Server

На компютрите в лабораторията е наличен Microsoft SQL Server Express 2014.

При инсталация изберете Windows Authentication Mode, за да използвате сървъра без допълнително потребителско име и парола:

Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

License Terms
Global Rules
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Feature Configuration Rules
Installation Progress
Complete

Server Configuration | Data Directories | User Instances | FILESTREAM

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

- ☒ Windows authentication mode
- ☐ Mixed Mode (SQL Server authentication and Windows authentication)

Specify the password for the SQL Server system administrator (sa) account.

Enter password:

Confirm password:

Specify SQL Server administrators

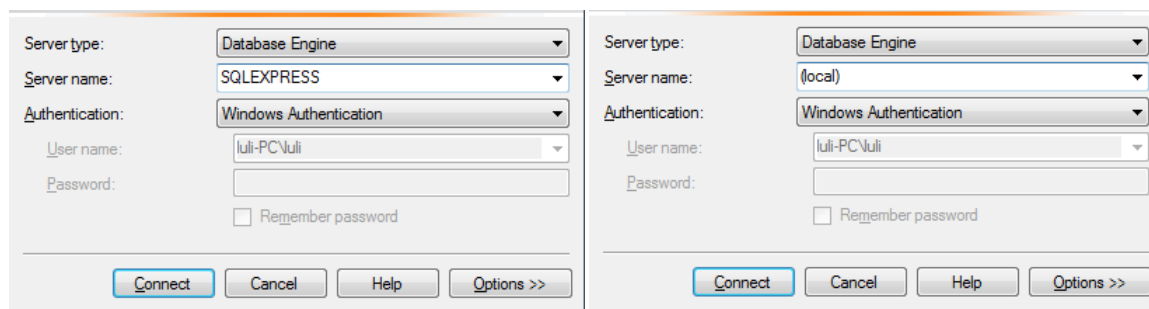
Add Current User | Add... | Remove



SQL Server administrators have unrestricted access to the Database Engine.

1.1. SQL Management Studio

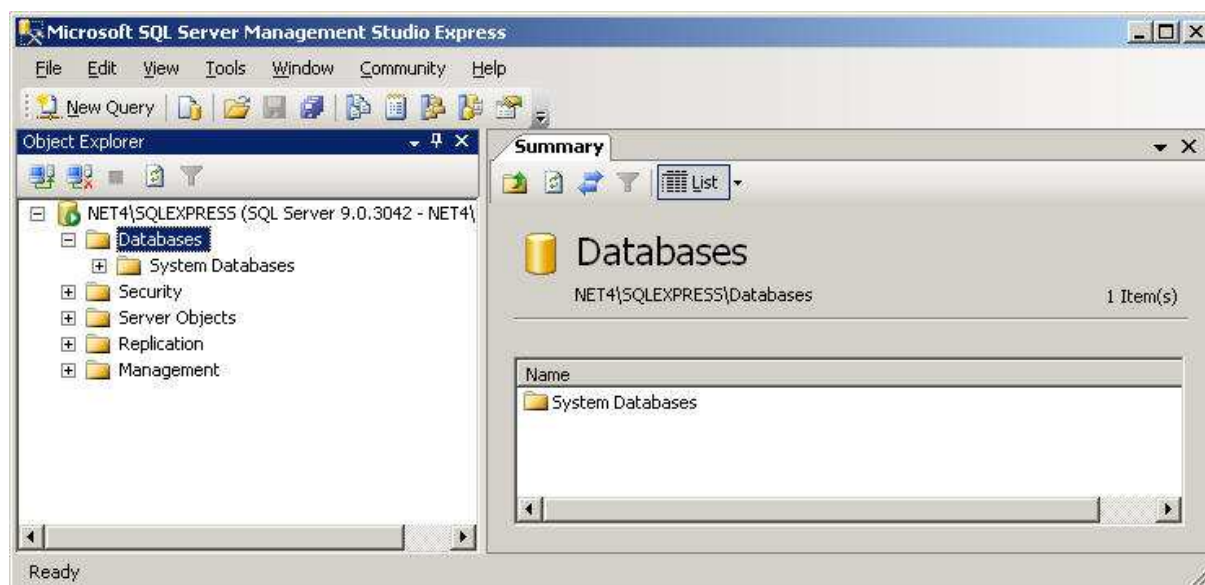
SQL Management Studio предоставя инструменти, които са необходим иза администриране на SQL Server и за създаване и поддържане на бази данни.

При стартиране е необходимо да се изгради връзка до сървъра. Сървърът може да се открие поименно – SQLEXPRESS (или ".\SQLEXPRESS") – ако не е променено името по подразбиране при инсталация. Алтернативно ако се намира на същата машина на която стартираме SQL Management Studio можем да се свържем с (local).



След свързване прозорецът е разделен на два панела: Object Explorer (дърво с обектите) в ляво и Details Panel (панел с детайли) вдясно. Позициите в дърво с обектите са подредени йерархично като списък от папки в Microsoft Windows Explorer. Можете да разширите подчинени позиции в йерархичен ред, като кликнете върху иконата  или ги свиете, като кликнете върху иконата .

Обектите, които се появява в дървото, когато стартирате SQL Management Studio, може да не съответстват на тези, показано в примерите. Например, името на сървър NET4/SQLEXPRESS. Вашият сървър ще бъде на някое друго име. Не се тревожете - това означава само, че вашия системен администратор е конфигурирал SQL Server по някакъв друг начин.






1.2. Управление на SQL Server

Преди да можете да използвате SQL Management Studio за да създадете нова база данни или за достъп до данни от съществуваща база данни, първо трябва да се идентифицирате в SQL Management Studio, да се уверите, че сървърът е пуснат, и да се свърже с базата данни, с която ще работите.

SQL Server поддържа два различни начина на удостоверяване на влизане, което гарантира, че само оторизирани потребители ще имат достъп до съответните данни - Windows Authentication и SQL Server Authentication. Microsoft препоръчва модел на Windows Authentication, който "прозрачно" позволява на потребителите на Microsoft Windows всистемата с техните потребителски имена и пароли в операционната система. Ако използвате удостоверяване SQL Server Authentication, то самия сървър управлява идентификацията на потребителя и потребителите трябва да въведете потребителско име и парола при свързване към базата данни.

Предполагаме, че вашия сървър ще използва препоръчителната конфигурация: Windows удостоверяване. Ако вашият сървър използва за удостоверяване на SQL Server Authentication, след това всеки път, когато се регистрирате на сървър или база данни, връзката ще трябва да въведете името и паролата за достъп. В този случай, въведете името и паролата, предоставена от администратора.

Преди да можете да се свържете с инстанция на SQL Server, услугата трябва да работи. Можете да намерите дали услугата се изпълнява или не, като погледнете в иконата на сървъра в дървото на конзолата в SQL Management Studio. Смысла на иконите е показан в следната таблица:

Означение	Смисъл
	Работещ сървър
	Преустановен сътвър
	Спрян сървър

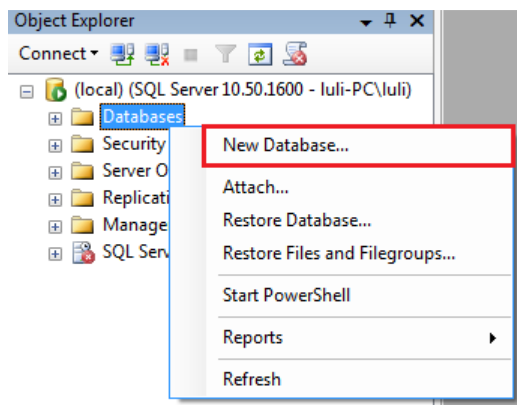
За стартиране на сървъра - щракнете с десния бутон на сървъра. Щракнете върху Start (Старт) в контекстното меню.

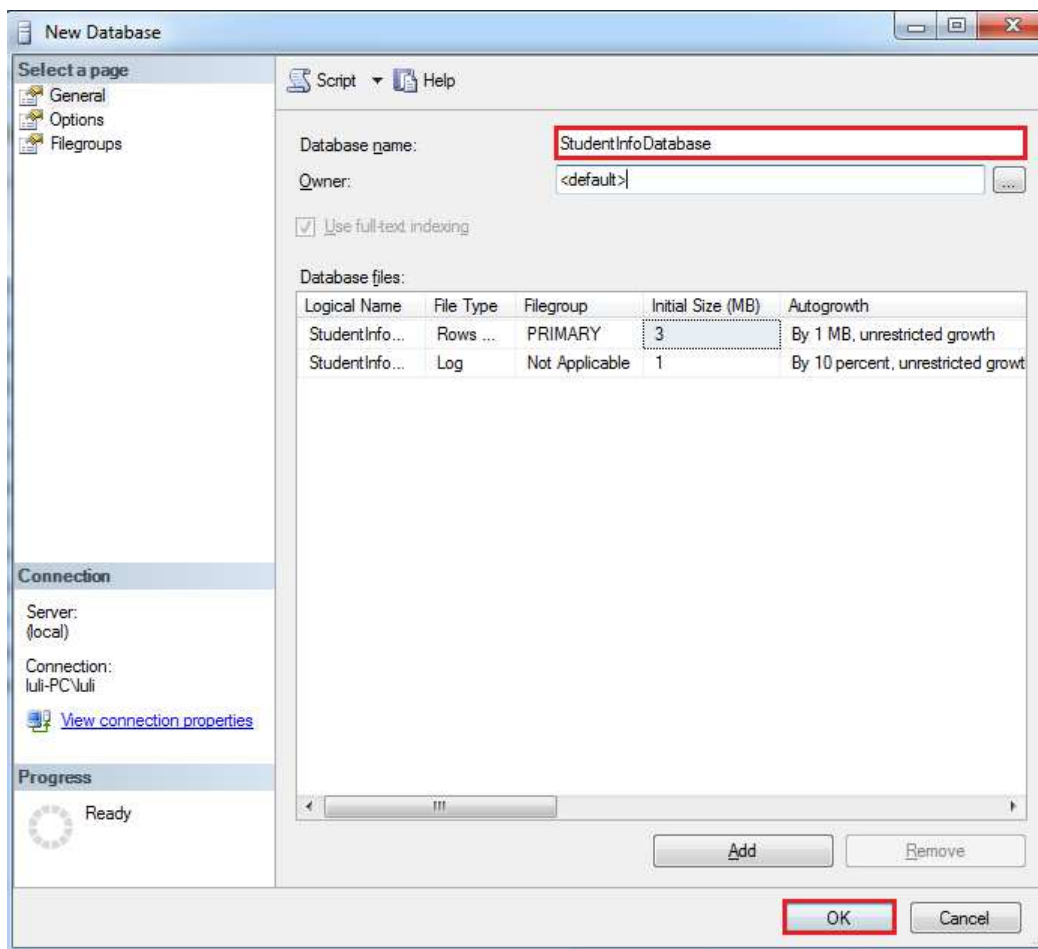
За спиране на сървъра - щракнете с десния бутон на сървъра. Натиснете Pause (Пауза) от контекстното меню.

За спиране на сървъра - щракнете с десния бутон на сървъра. Кликнете Stop (Стоп) от контекстното меню.

Преди да продължим да работим по упражнението, уверете се, че сървъра е стартиран.

1.3. Създаване на база данни





Задача:

Създайте нова база данни с име **StudentInfoDatabase**.

1.4. Копие и възстановяване на база

Независимо от това колко сигурна е основната технология, трябва да се има предвид възможността от провал на компютърния хардуер, грешки и пропуски в софтуера, както и погрешните действия на потребителя. Най-добрият начин да се предпазите е редовно архивиране на вашите данни и да се съхраняват в някое сигурно място. Този процес се нарича "създаване на резервно копие." Ако нещо се случи (а то нещо обезателно ще се случи), можете да използвате архивира за да се възстанови състоянието на база данни, както е била преди възникването на проблема.

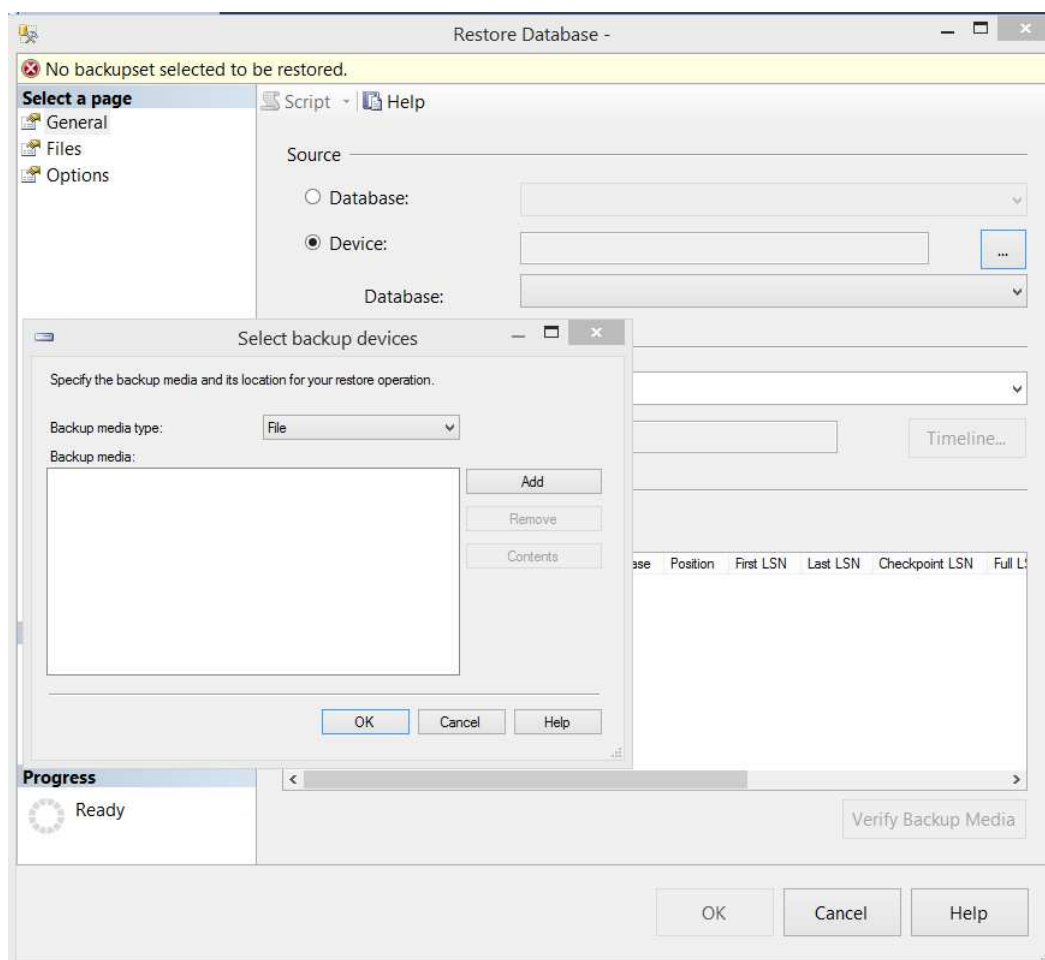
1.4.1. Архив на база

1. Стартирайте SQL Management Studio.
2. **Object Explorer --> Databases --> Избор на база (StudentInfoDatabase) → Десен бутон --> Tasks → Back Up...**
3. В появилия се диалог, в Destination изберете Add.. за да посочите файл, а за BackUp Type изберете Full.

Ако не променяте името на файла, то в един и същи файл ще се натрупват множество BackUp-и на базата.

1.4.2. Възстановяване на база

1. Стартирайте SQL Management Studio.
адссад
2. **Object Explorer --> Databases --> Десен бутон --> Restore Database**
3. В появилия се диалог избираме че ще възстановяваме от **Device --> File**




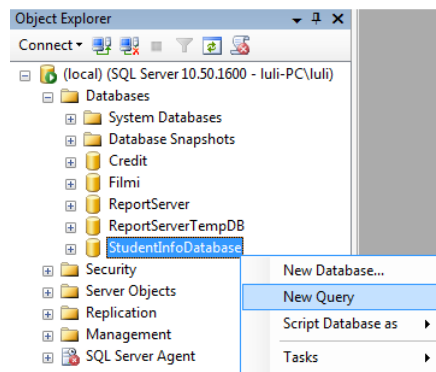
- a. От бутона **Add** изберете файла с архива на базата.
- b. В полето **Database** посочете съдържащата се в архива база.
- c. Натиснете **OK** и изчакайте приключването на операцията

По този начин може да си пренасяте базата (ако използвате компютрите в лабораторията).

1.5. Работа с база данни

1. Стартирайте SQL Management Studio.
адссад

2. **Object Explorer --> Databases --> StudentInfoDatabase → Десен бутон --> New Query**
3. В появилия се прозорец в дясно се въвеждат SQL заявки и се изпълняват с бутона **Execute**  **Execute**.



Задача:

1. Създайте нова таблица с име StudStatus.

```
CREATE TABLE StudStatus
(
    Id INT PRIMARY KEY,
    StatusDescr VARCHAR(100)
);
```

2. Добавете данни в новата таблица

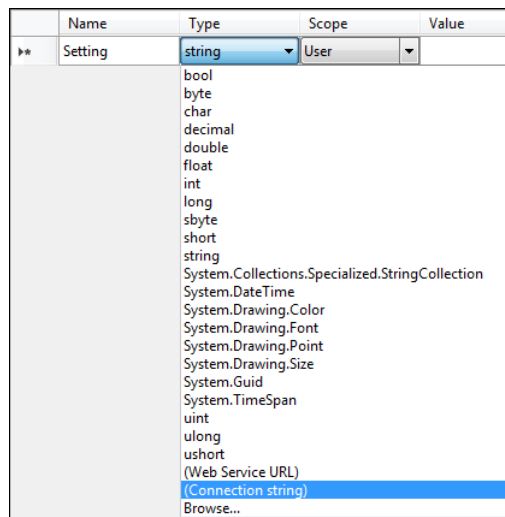
```
INSERT INTO StudStatus
    (Id, StatusDescr)
VALUES
    (1, 'Редовен'),
    (2, 'Самост. подготовка'),
    (3, 'Задочен'),
    (4, 'Прекъснал по успех'),
    (5, 'Прекъснал по болест'),
    (6, 'Прекъснал по майчинство');
```

2. Работа с база данни с ADO.Net

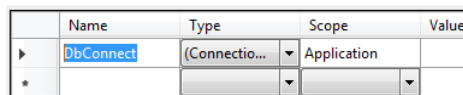
2.1. Връзка към база данни


В проекта **StudentInfoSystem**:

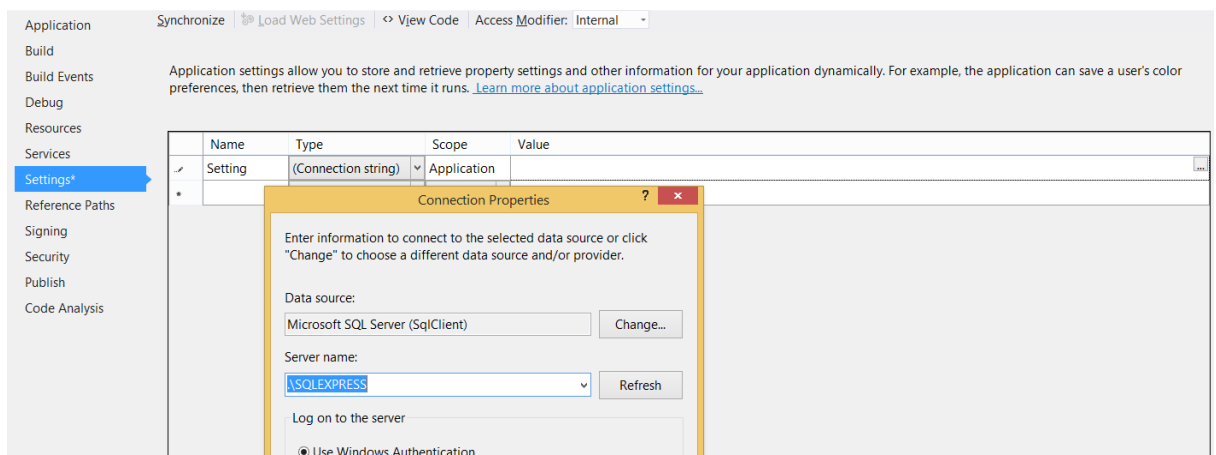
1. Стартирайте MS Visual Studio.
2. За съзране с базата данни настройваме проекта по следния начин:
 - 2.1. От настройките на проекта (**Project --> Properties --> Settings**) добавяме нов ред от тип **Connection string**



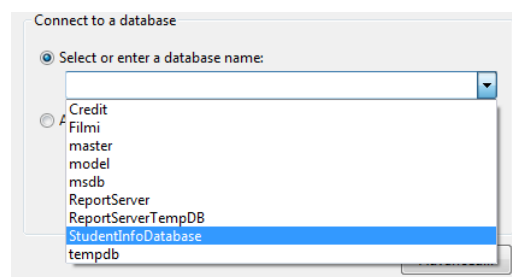
2.2. Задайте име на нстройката DbConnect;



2.3. Настройте връзката от диалога на бутона (...) . Попълва се името на сървъра (напр (local)):



2.4. Попълва се базата с която ще работим. Ако е попълнено вярно името на сървъра, диалогът ще предложи наличните бази:



2.5. Избирате OK и Save (File → Save Selected Items).

Така създадената настройка се съхранява във файла app.config (или файла web.config).
Ако се отвори в него се вижда следното:

```
<connectionStrings>
  <add name="StudentInfoSystem.Properties.Settings.DbConnect"
        connectionString="Data Source=(local);
        Initial Catalog=StudentInfoDatabase;Integrated Security=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

2.2. Заявка към база данни

Добавете код към класа `MainWindow`:

Клаузи към файла:

```
using System.Data;
using System.Data.SqlClient;
```

Свойство на формата/класа:

```
public List<string> StudStatusChoices { get; set; }
```

Функция:

```
private void FillStudStatusChoices()
{
}
```

Функционалност:

```
StudStatusChoices = new List<string>();
```

```
using (IDbConnection connection = new
SqlConnection(Properties.Settings.Default.DbConnect))
{
    string sqlquery =
        @"SELECT StatusDescr
        FROM StudStatus";

    IDbCommand command = new SqlCommand();
    command.Connection = connection;
    connection.Open();

    command.CommandText = sqlquery;
    IDataReader reader = command.ExecuteReader();

    bool notEndOfResult;

    notEndOfResult = reader.Read();

    while (notEndOfResult)
```

```

{
    string s = reader.GetString(0);

    StudStatusChoices.Add(s);

    notEndOfResult = reader.Read();
}
}

```

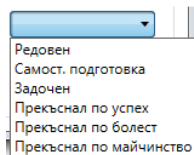
Със създаването на нов обект от тип `IDbConnection` осъществяваме връзка с базата данни, възползвайки се от съществуващия стандартен клас `SqlConnection`, на който подаваме настройката за връзка. След това подаваме заявката, която искаме да бъде изпълнена, възползвайки се стандартния клас `SqlCommand`, на който подаваме SQL заявка. След това отваряме връзката и подаваме заявката. Възнатия резултат, ако има, изчитаме в обект от тип `IDataReader`.

Получените данни изчитаме по редове, докато не свършат редовете (`notEndOfResult`). Колоните за всеки ред се конвертират към изходния тип. В случая има само една колона, за това подаваме само параметър 0 (`GetString(0)`).

Остава да:

- Извикате функцията `FillStudStatusChoices()` в конструктора.
- Подмените контролата за Статус със `ListBox`
- Свързвате `ItemsSource` към новото поле `StudStatusChoices`
`ItemsSource="{Binding StudStatusChoices}"`
- Задавате `DataContext` - `this.DataContext = this;`

Така съдържанието на интерфейса е извлечено от базата данни.



Задача: Добавете още стойности към таблицата в SQL Server Management Studio и вижте дали се появяват на интерфейса.

В проектите **StudentRepository** и **UserLogin**:

Задача: Създайте настройки за връзка към базата данни със същото име `DbConnect` и в проектите `StudentRepository` и `UserLogin`.

Конзолните проекти по подразбиране нямат клас за настройки, но това не пречи да се създаде:

[This project does not contain a default settings file. Click here to create one.](#)

3. ORM Entity Framework

Entity Framework представлява технология за достъп до данни на компанията Microsoft. Технологията се основава на Object Relation Mapping (ORM). Това не е единствената такава технология - съществуват множество такива технологии под .Net - и всички работят на един и същ принцип. Ето защо запознаване с Entity Framework ще помогне при необходимост за работа с друга ORM технология.

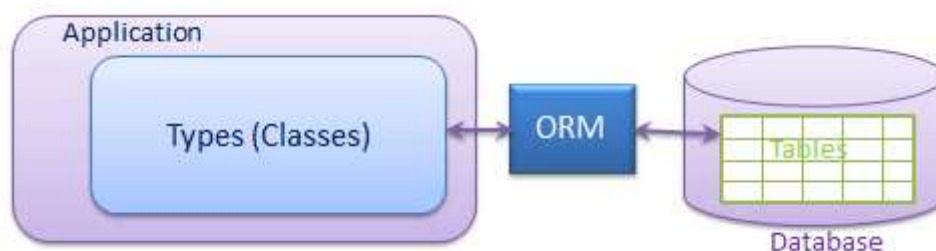
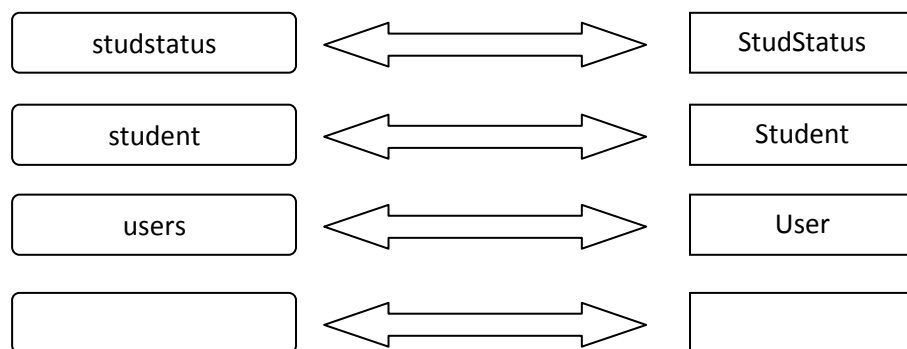
Реализация на ниско ниво посредством EDO.Net има следните минуси:

- използване на SQL заявки директно в C# кода. Те не подлежат на проверка от компилатора и е лесно да се допусне грешка.
- ръчно управление на връзката с базата данни - отваряне, затваряне.
- Работа с нетипизирани данни. След ExecuteReader() върнатия резултат може да е произволен. Трябва предварително да разполагаме с допълнителна информация за типовете на всяка върната конла (целочислен, низ, т.н.)
- Трябва да се грижим за съхраняване на върнатите резултати. В най-добрия случай трябва да напишем клас, описващ данните, и с ръчно обхождане на reader да ги съхраним в обект на този клас.

Например за таблица StudStatus подходящ клас е следния:

```
public class StudStatusProxy
{
    public int Id, { get; set; }
    public string StatusDescr { get; set; }
}
```

Entity Framework в основата си също използва EDO.Net, но предлага по-удобен стил на работа. Тъй като кодът за представяне на таблица като клас е еднотипен, то е необходим инструмент, който автоматично да генерира класове от съществуващи таблици, или пък да генерира таблици в база данни от съществуващи класове. Такава задача изпълняват ORM технологиите, в частност Entity Framework.



3.1. Сценарии при използване на Entity Framework

Entity framework (EF) е удобен в следните три случая:

1. Database First – Ако вече имате съществуваща база данни или смятате да проектирате базата преди останалите части на приложението.
2. Code First – Ако искате да се фокусирате на основните класове и след това от тях да генерирате базата данни.
3. Model First – Ако искате да проектирате схемата на базата данни с визуален дизайнер и след това да създадете самата база и класовете от схемата.

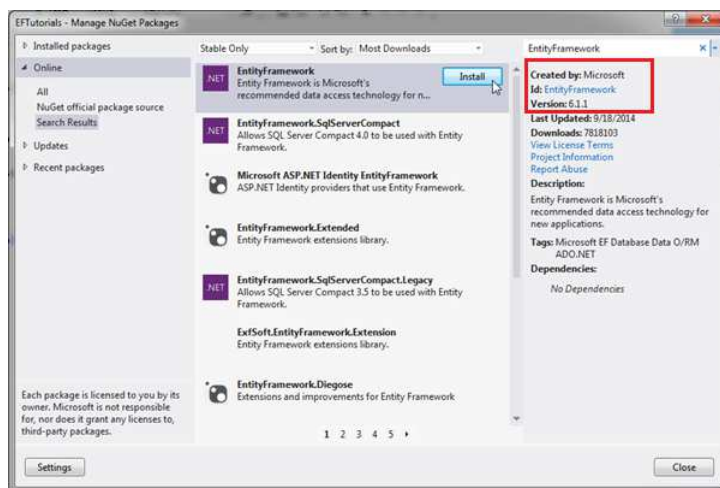
Ние ще използваме Code First.

3.2. Добавяне на Entity Framework към проекта

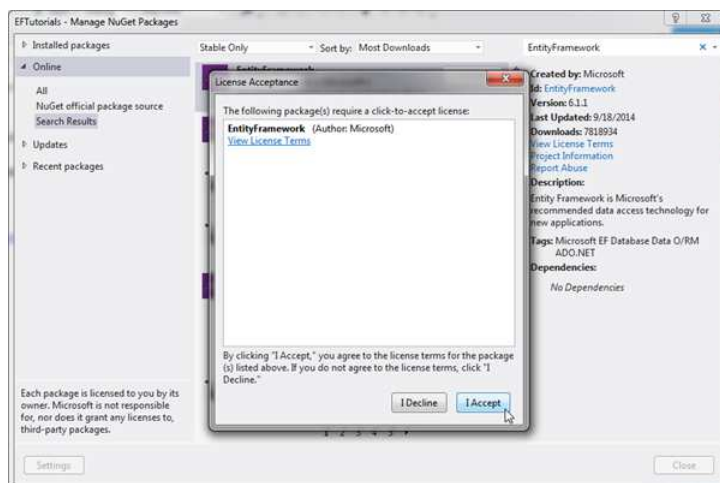
В новия си проект трябва да добавим референция към библиотеките за Entity Framework. Някой от стъпките надолу могат да варират в зависимост от нивото на инсталация на компютъра, на който седите.

В проекта **StudentInfoSystem**:

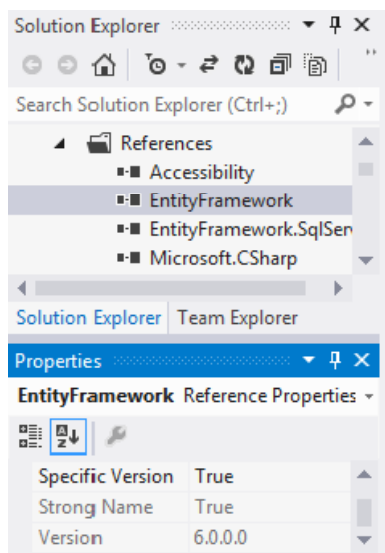
1. Натиснете с десния бутон на мишката върху името на вашия проект в solution explorer и от контекстното меню изберете **Manage NuGet Packages...** Това ще отвори за вас диалоговия прозорец **Manage NuGet packages**.
2. В ляво изберете „Online“
3. Намерете EntityFramework пакета и натиснете Install



4. Натиснете на бутона **I Accept** за потвърждение на лиценза. Това ще стартира инсталацията.



След като тя завърши успешно, в проекта ви ще бъде добавен автоматично EntityFramework.dll файла (не се вижда в Solution Explorer, а само в реалните файлове). Под категория References трябва да виждате нещо подобно.



След тази стъпка сме готови да ползваме Entity Framework в конкретния проект!

В проектите **StudentRepository** и **UserLogin**:

Задача: Добавете пакета Entity Framework и в проектите StudentRepository и UserLogin.

3.3. Класове описващи данните

За всички данни, които искаме да се съхраняват в базата (в таблица) трябва да опишем клас (Entity Class).

3.3.1. Описване на данни за съхранение

В проекта **StudentInfoSystem**:

Вече имаме два класа, подходящи за запазване: User и Student. Полетата, които искаме да запазваме трябва да преправим на свойства.

Приведете вашите класове в подобен формат:

<pre>public class User { public System.String Username { get; set; } public System.String Password { get; set; } public System.String FakNum { get; set; } public System.Int32 Role { get; set; } public System.DateTime Created { get; set; } public System.DateTime ActiveTo { get; set; } }</pre>	<pre>public class Student { public string Name {get; set;} public string Surname {get; set;} public string FamilyName {get; set;} public byte[] Photo { get; set; } ... }</pre>
--	--

3.3.2. Идентификатор

Класовете, които участват в база (Entity Classes) трябва да имат идентификатор, който да служи за първичен ключ.

Добавете следниве свойства:

<pre>public System.Int32 UserId { get; set; }</pre>	<pre>public int StudentId { get; set; }</pre>
---	---

3.3.3. Разширения на класовете

Класовете, които участват в база (Entity Classes), могат да имат и други членове (методи, конструктори, ...)

Следният примерен клас не е проблем:

```
public class Student
{
    public Student()
    {
    }

    public Student(string name, string surname, string familyname)
    {
        Name = name;
        Surname = surname;
        FamilyName = familyname;
        ...
    }
}
```



```

        public string Name { get; set; }
        public string Surname { get; set; }
        public string FamilyName { get; set; }
        public byte[] Photo { get; set; }
        ...

        public override string ToString() { return this.Name; }
    }

```

3.3.4. Незадължителни полета

Данните, които могат да останат непълнени трябва да са от null-разрешен тип (nullable). string по дефиниция може да е null. За останлите типове има предефиниран null-разрешен вариант с "?" (въпросителен).

Например за User датата ActiveTo може да не е посочена, което значи че потребителят е активен. За да е възможно трябва да декларираме свойството така:

```

public System.DateTime? ActiveTo { get; set; }

```

Преправете и останалите свойства, където е необходимо да може да съхраняват null.

3.4. Клас, описващ работата с базата данни

След като приключим с описването на класовете, трябва да напишем и контекстен клас, който наследява DbContext. Това се изисква от Code-First подхода.

1. Създайте нов клас `StudentInfoContext`.
2. Наследете новия клас от `DbContext`.

```

class StudentInfoContext : DbContext

```

За целта трябва да използвате `using System.Data.Entity;`

3. Създайте конструктор на новия клас.
4. Извикайте към конструктора базовия конструктор на класа `DbContext`, от който наследяваме:

```

        public StudentInfoContext()
            : base()
        { }

```

5. Базовият конструктор може да приема конкретна връзка към база. Подайте вече настроената връзка `DbConnect`:

```

        base(Properties.Settings.Default.DbConnect)

```

Ако не подадем параметър EF ще създаде нова база на локалния сървър вграден във Visual Studio. Посочвайки известна база на самостоятелен SQL Server, ще може да следим какво се случва в нея.

6. Добавете свойства от типа `DbSet` за класовете, които искаме да са част от модела (в нашия случай `Student` и `User` класовете)

```

public DbSet<Student> Students { get; set; }
public DbSet<User> Users { get; set; }

```

DbSet е колекция. Колекцията е от ентити класове (множество обекти от посочените класове Student и User), затова на свойствата от този тип даваме имена в множествено число (Students и Users).

DbSet наследява IEnumerable.

В проектите **StudentRepository** и **UserLogin**:

Задача: Създайте класове наследяващи **DbContext** и в проектите StudentRepository и UserLogin:

- Кръстете ги съответно:
 - в проекта StudentRepository - **StudentContext**
 - в проекта UserLogin - **UserContext**
-

4. CRUD операции с Entity Framework

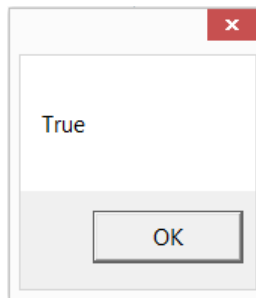
CRUD = Create, Read, Update, Delete

4.1. Read

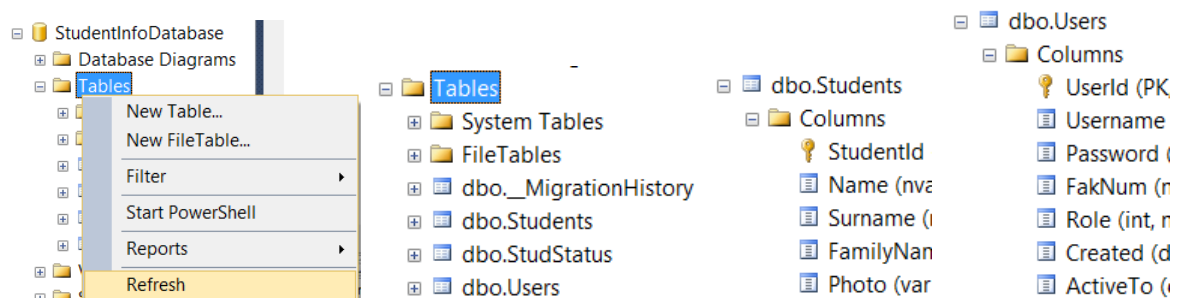
Да направим функция, която проверява колко записа за студенти има. Тъй като DbSet наследява IEnumerable, можем да използваме функциите на LINQ, в случая Count().

В проекта **StudentInfoSystem**:

1. Създайте функция TestStudentsIfEmpty(), която връща булева стойност.
2. Създайте инстанция от контекста за работа с базата данни, която ще използваме:
`StudentInfoContext context = new StudentInfoContext();`
3. Създайте инстанция на колекция, с която да се обръщаме към множеството Students достъпно през контекста context:
`IEnumerable<Student> queryStudents = context.Students;`
4. Създайте променлива, в която да съхраним бройката на множеството Students:
`int countStudents = queryStudents.Count();`
5. Добавете проверка queryStudents дали е нула.
 - 5.1. Ако е: да се връща true. Иначе false.
6. Извикайте функцията TestStudentsIfEmpty() от някъде (временен тестови бутон) и проверете резултата.



7. Проверете базата в SQL Server Management Studio. Таблиците описани в `StudentInfoContext` трябва да са създадени.



4.2. Create

Чрез Entity Framework могат да се извършват промени в базата данни.

Да направим метод, който да запише тестовите данни за студенти `TestStudents` в базата данни.

1. Създайте функция `CopyTestStudents()`.
2. Създайте инстанция от контекста за работа с базата данни, която ще използваме:

```
StudentInfoContext context = new StudentInfoContext();
```
3. Добавете към множеството `Students` достъпно през контекста `context` обектите на `TestStudents`:

```
foreach (Student st in StudentData.TestStudents)
{
    context.Students.Add(st);
}
```

4. Посочете на контекста да запази посочените промени в базата:

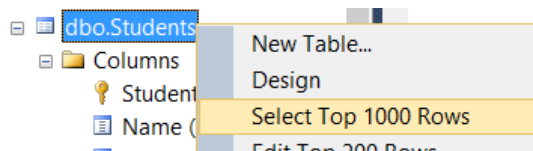
```
context.SaveChanges();
```

Промените ще се запишат единствено при извикване на `SaveChanges`.

5. Организирайте извикване на метода когато таблицата `Students` е празна.

```
if (TestStudentsIsEmpty())
    CopyTestStudents();
```

6. Проверете базата в SQL Server Management Studio. В таблицата `Students` трябва да има записи.



Задача: Добавете метод, който да записва тестовите данни за потребители TestUsers (или _testUsers) в базата данни и го извиквайте по подходящ начин (само веднъж).

4.3. Read (отново)

Чрез Entity Framework, код за извличане на данни от таблицата Students ще изглежда по следния начин:

```
private static List<Student> GetRegions()
{
    StudentContext context = new StudentContext();
    List<Student> students = context.Students.ToList();
    return students;
}
```

Пример: Нека подменим използването на TestStudents в метода IsThereStudent в класа StudentData с обръщение към базата данни:

Старата заявка:

```
Student result =
(from st in TestStudents
 where st.FacultyNumber == facNum
 select st).First();
return result;
```

Новата заявка от базата данни:

```
StudentContext context =
    new StudentContext();

Student result =
(from st in context.Students
 where st.FacultyNumber == facNum
 select st).First();
return result;
```

LINQ заявка върху Entity клас ще формира SQL заявка към базата данни. Резултатът ще е Entity обекти.

Задача: Подменете използването на TestUsers в метода IsUserPassCorrect в класа UserData с обръщение към базата данни.

Използвайте контекста на съответния проект, в който се намира кода.

4.4. Update

Пример: Нека подменим използването на TestStudents в метода AssignUserRole в класа UserData с обръщение към базата данни:

```
static public void AssignUserRole(int userid, UserRoles newRole)
```

Старата логика:

```
TestUsers[userid].Role = newRole;
```

Новата логика със заявка от базата данни:

```
UserContext context =  
    new UserContext();  
  
User usr =  
(from u in UserData.TestUsers  
 where u.UserId == userid  
 select u).First();  
  
usr.Role = newRole;  
  
context.SaveChanges();
```

Вашият метод може да изглежда различно.

Задача: Подменете използването на TestUsers в навсякъде с обръщение към базата данни.

Използвайте контекста на съответния проект, в който се намира кода.

4.5. Delete

След като намерим обектът, който искаме да изтрием със заявка от базата, можем да го подадем за изтриване на колекцията към контекста.

Например изтриване на конкретен студент delObj би изглеждало така:

```
StudentContext context = new StudentContext();  
  
Student delObj =  
    (from st in context.Students  
     where st.FacultyNumber == facNum  
     select st).First();  
  
context.Students.Remove(delObj);  
  
context.SaveChanges();
```

Самостоятелна задача: (подобна на тази за работа в час)

Създайте Entity клас Logs и го използвайте в класа Logger, за да се записват логовете и в базата, освен във файл.

Запишете си проекта! USB, DropBox, GoogleDrive, e-mail, SmartPhone, където и да е.
До края на упражненията ще работите върху този проект.

Вкъщи също ще работите върху този проект.
Накрая трябва да го представите завършен.