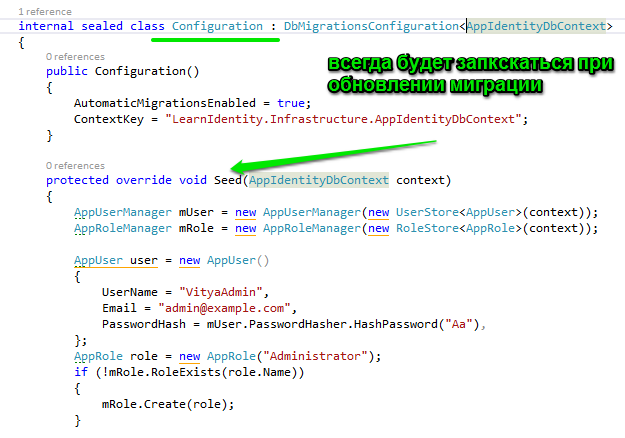
Действия если обновляются таблицы



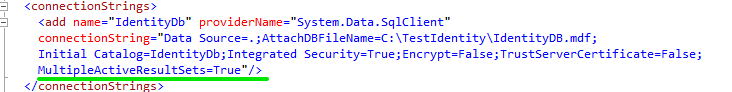






Когда из одной таблице делаем выборку и переключаемся на другую таблицу





При ошибке cannot attach the file as database

На master – drop database имя

<http://msdn.microsoft.com/ru-ru/data/jj591583.aspx> - атрибуты

возможно при создании может быть ошибка с путями

AppDomain.CurrentDomain.SetData("DataDirectory", System.IO.Directory.GetCurrentDirectory());

<connectionStrings>

<add name="myconnetc"

connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\SchoolDB.mdf;

initial catalog=School;Integrated Security=true"

providerName="System.Data.SqlClient"/>

</connectionStrings>

приведение к типу

Console.WriteLine("Found entity of type {0} with state {1}",

ObjectContext.GetObjectType(entry.Entity.GetType()).Name,

entry.State);

Вложенный подзапрос и объединение в одну строку

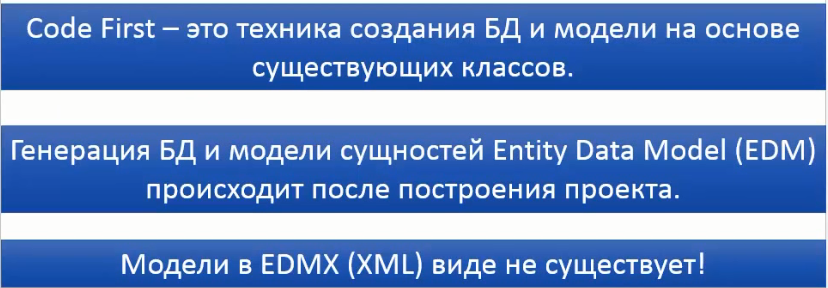
IEnumerable<string> t = context.GetValidationErrors().

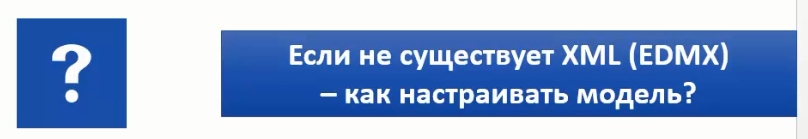
SelectMany(d => d.ValidationErrors).Select(r => r.ErrorMessage);

tranc.Rollback();

ReloadLocalData();

throw new Exception(t.Aggregate(new StringBuilder(), (n, i) => n.AppendLine(i)).ToString());

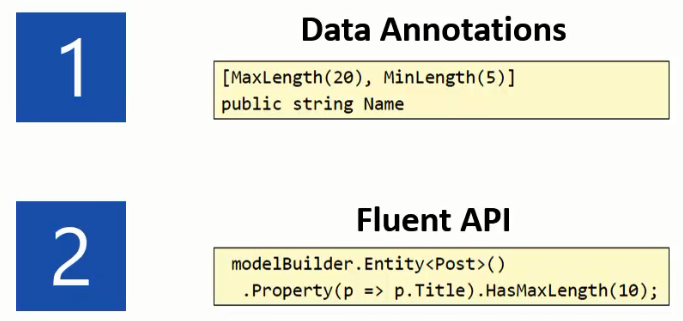


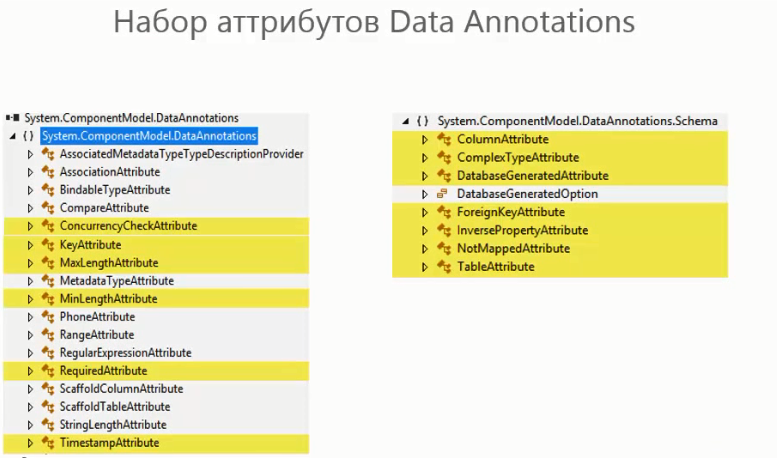


Классы должны следовать определенным соглашениям

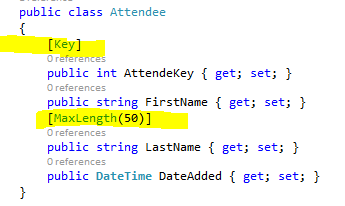
С помощью атрибутов

С помощью Fluent API



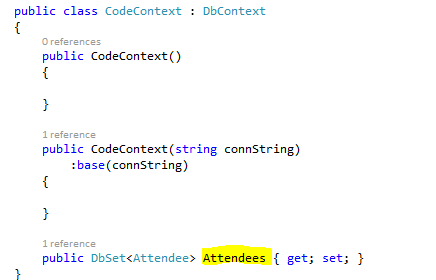


Таблица



Контекст содержащий таблицы

DbSet<Attendee>

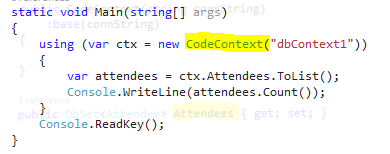


Данный пример создает БД с одной таблицей Attendee.

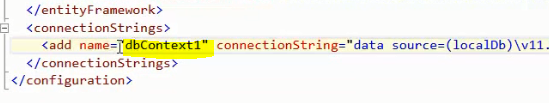
Но не пересоздает ее, когда мы меняем структуру таблицы.

В случае, если мы будем изменять тип поля, или добавлять еще одну таблицу -

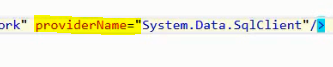
программа нам выдаст ошибку. Чтобы этого избежать нужно удалить старую БД и запустить программу.

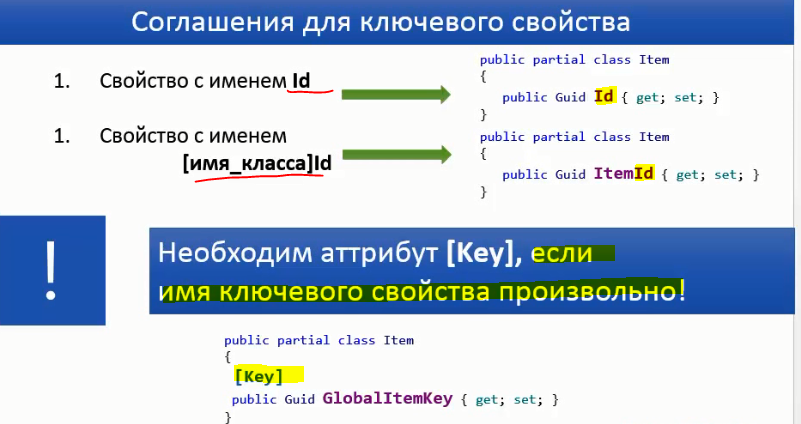


В конструктор передаем имя строки подключения ее отличие от Model First и DB First в том что нет ссылок на XML файлы в которых описывается модель

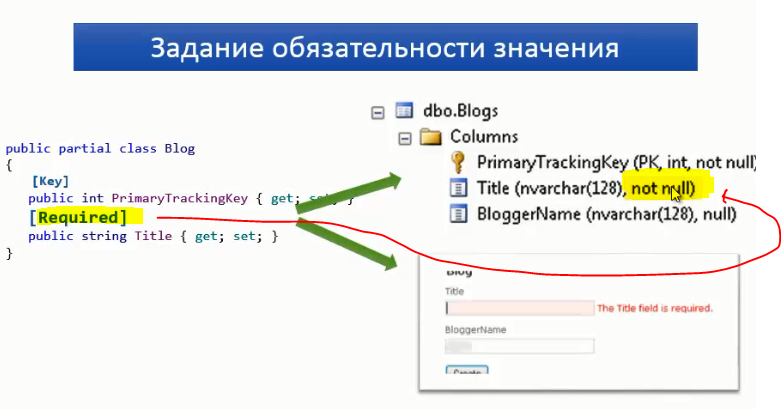


Указывает каким провайдером данных будем пользоваться

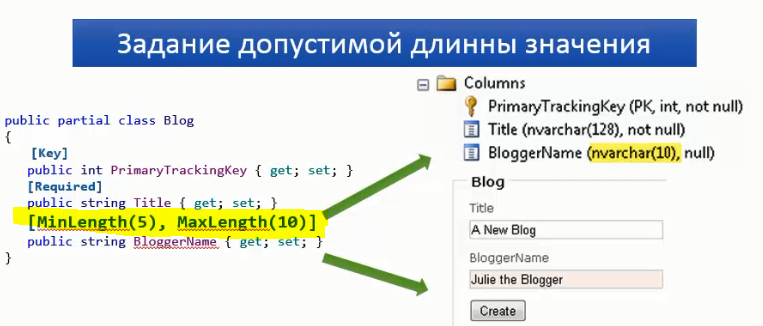


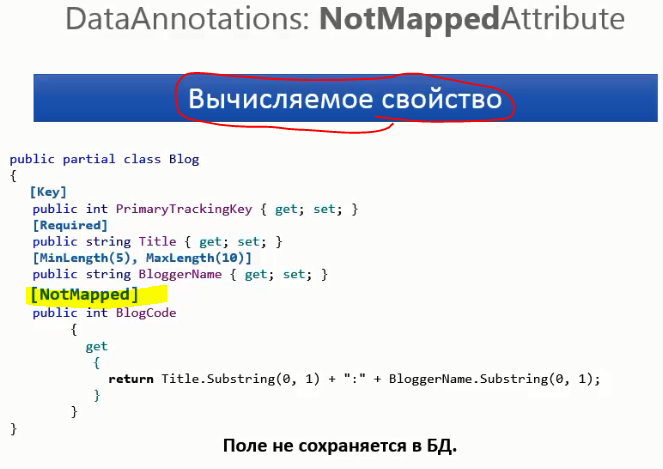


Задание обязательности значения (если null не пройдет валидацию на стороне клиента)



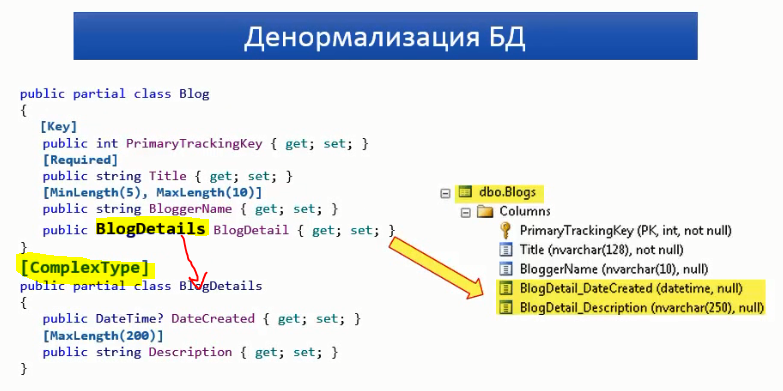
Задание длины (если меньше не пройдет валидацию на стороне клиента)



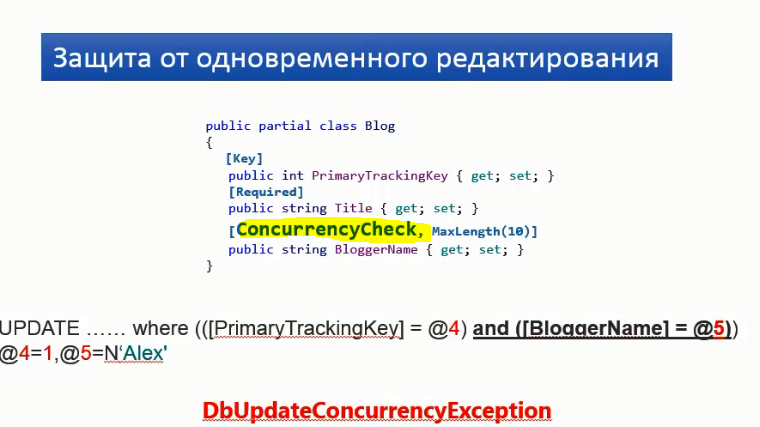


Вынесение отдельных сущностей в отдельные классы

Но таблица ОДНА

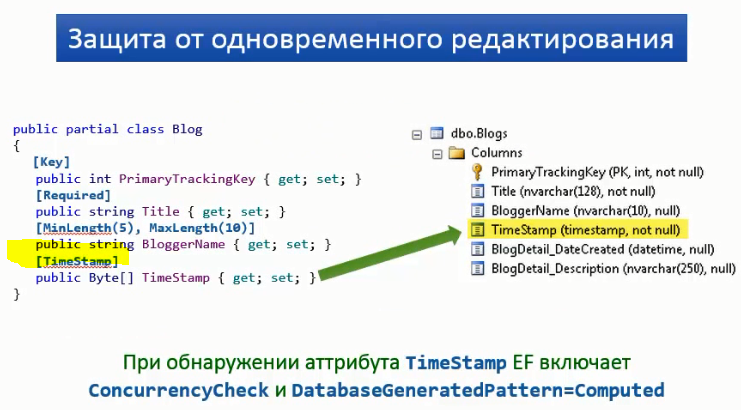


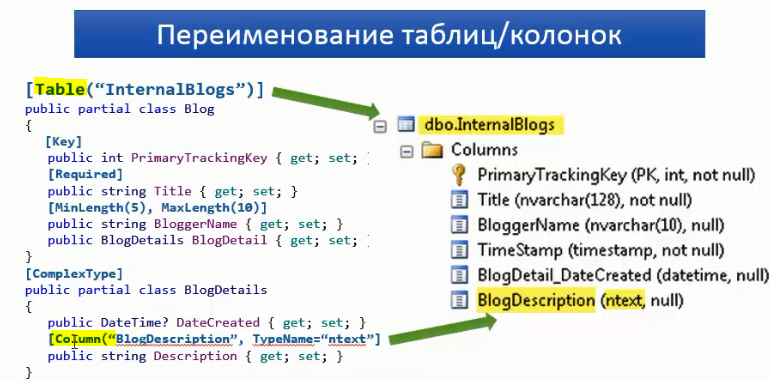
Защита сущности от одновременного редактирования несколькими людьми или программами (что то наподобие LOCK)



Защита любого поля сущности от одновременного редактирования

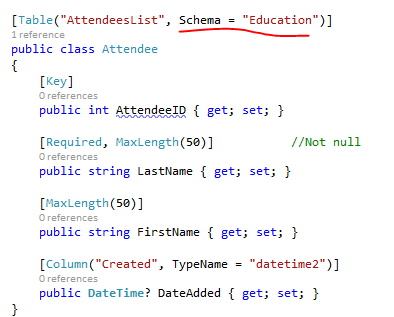
Назначается версия кортеджу и после изменения версия меняется и если мы локально изменили кортедж и попытаемся сохранить то получим исключение, в связи с этим нужно будет обновлять локальные данные





Стратегии инициализации БД

Указываем имя и схему таблицы



Database.SetInitializer

Database.SetInitializer(new System.Data.Entity.DropCreateDatabaseIfModelChanges<CodeContext>()); - если модель изменилась и она не соответствует схеме бд уничтожить бд (со всеми полями) и создать новую

Database.SetInitializer(new System.Data.Entity.DropCreateDatabaseAlways<CodeContext>()); - всегда удалять старую бд и создавать новую

Database.SetInitializer(new System.Data.Entity.CreateDatabaseIfNotExists<CodeContext>()); - создать если бд не существует (по умолчанию) при изменении схемы класса таблицы (MAX или тип) то будет ОШИБКА

////Используется CodeFirst Migrations

Database.SetInitializer(new System.Data.Entity.MigrateDatabaseToLatestVersion<CodeContext, CF.DataAccess.Migrations.Configuration>()); - выполнить миграцию бд,

-если бд нет то она будет создана,

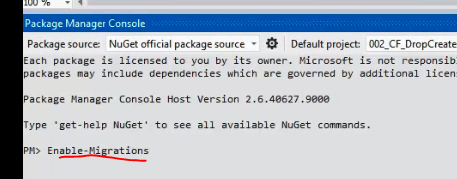
-если есть бд и схема соответствует модели, то бд будет использована

-если схемы не соответствуют то будет сгенерирован скрипт и преобразует схему бд от модели

Чтобы включить миграцию

TOOLS->Package Manager->





Создать класс самостоятельно

namespace CF.DataAccess.Migrations

{

using System;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

public sealed class Configuration : DbMigrationsConfiguration<CF.DataAccess.CodeContext>

{

public Configuration()

{

AutomaticMigrationsEnabled = false;

AutomaticMigrationDataLossAllowed = false;

}

protected override void Seed(CF.DataAccess.CodeContext context)

{

// This method will be called after migrating to the latest version.

// You can use the DbSet<T>.AddOrUpdate() helper extension method

// to avoid creating duplicate seed data. E.g.

//

// context.People.AddOrUpdate(

// p => p.FullName,

// new Person { FullName = "Andrew Peters" },

// new Person { FullName = "Brice Lambson" },

// new Person { FullName = "Rowan Miller" }

// );

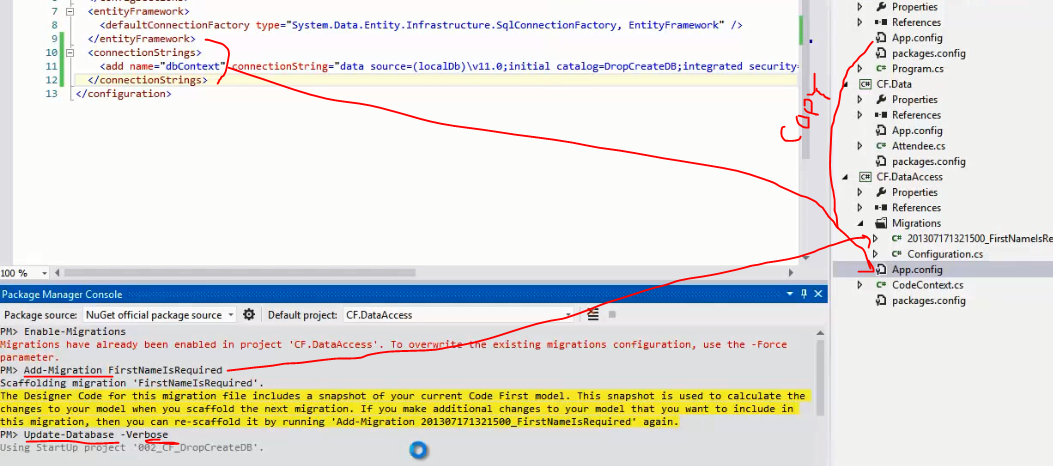
//

}

}

}

Verbose отображает изменения выполненные на базе данных





Чтобы разрешить автоизменения схемы бд относительно классов

Чтобы разрешить уменьшать размер полей (но можно потертерять данные (урежится слово или что то другое))



Самостоятельная Замена типов и значений схемы БД относительно кода

Необходимо создать класс наследующий интерфейс IDatabaseInitializer<КОНТЕКСТ> и реализовать метод InitializeDatabase(ManContext context)

public class ManInitializer : IDatabaseInitializer<ManContext>

{

public void InitializeDatabase(ManContext context)

{

if (context.Database.Exists())

context.Database.Delete();

context.Database.Create();

context.Men.Add(new Man { ManID = 1, Name = "Alex" });

context.Men.Add(new Man { ManID = 2, Name = "Dima" });

context.Men.Add(new Man { ManID = 3, Name = "Aleksey" });

context.SaveChanges();

}

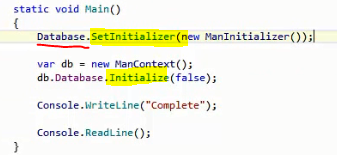
}

Устанавливаем инициализацию

db.Database.Initialize(false); - инициализируем БД Запускает интерфейс [IDatabaseInitializer<TContext>](http://msdn.microsoft.com/ru-ru/library/gg696323(v=vs.113).aspx), зарегистрированный в данном контексте. Если параметр «force» имеет значение true, инициализатор запускается независимо от того, запускался ли он уже прежде. Это может оказаться полезным, если база данных была удалена во время выполнения приложения и должна быть повторно инициализирована. Если параметр «force» имеет значение false, инициализатор выполняется, только если не выполнялся ранее для этого контекста

Принудительная иициализация,

* если была проинициализирована и false то при запросе инициализироваться не будет
* если была проинициализирована и true то при запросе заново будет инициализироваться

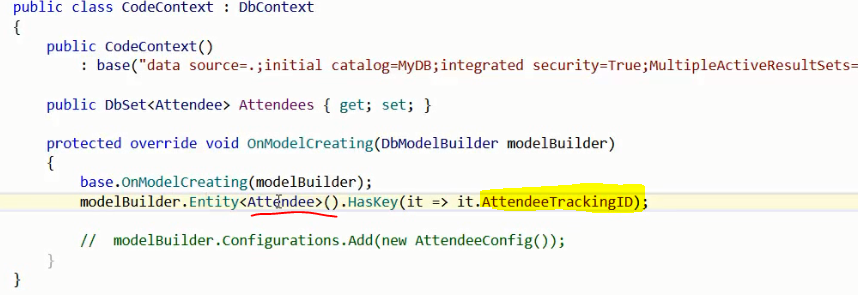


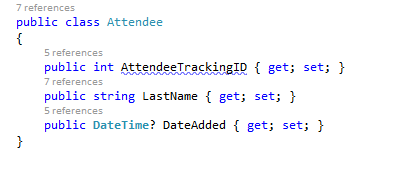
Fluent API

Первый способ, минус заключается в том что если имеется множество таблиц, то все инициализация проходит в теле одного метода, а это большое количество строк, и после разобраться будет очень проблематично

Назначили первичный ключ сущности (таблице)

Аналогично атрибуту [Key]



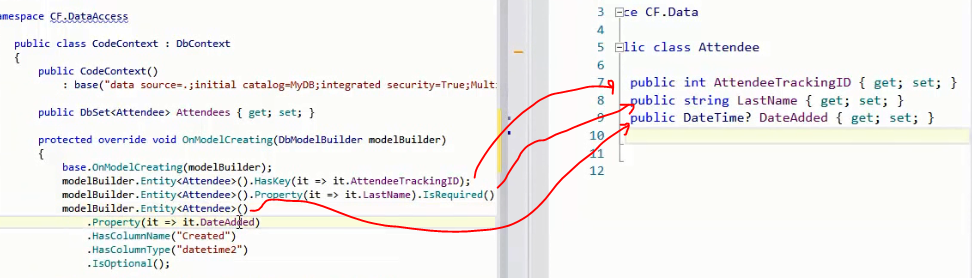


Первичный ключ,

Обязательное и макс длина 100,

IsConcurrancyToken – разрешает конфликты одновременного доступа

Название столбца, тип, не обязательное заполнение (опциональное допускает значение null)



Задать имя таблице

1 – имя таблице

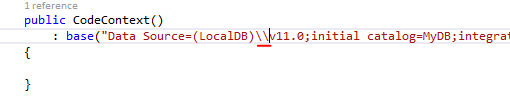
2 - имя схемы

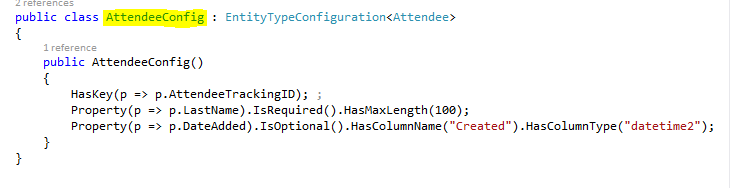
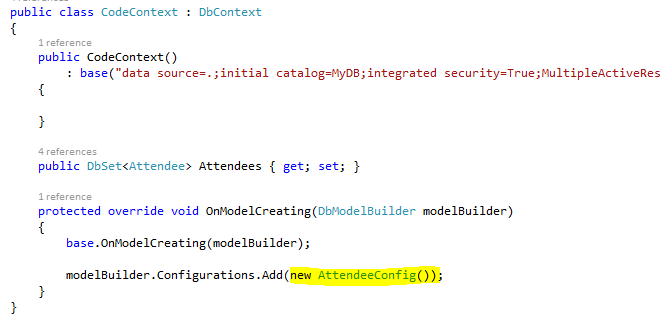


Второй способ

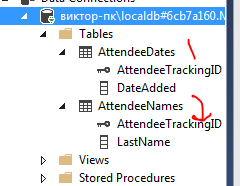
Создание конфигураций и выделение их в отдельные классы

экранировать локальную базу

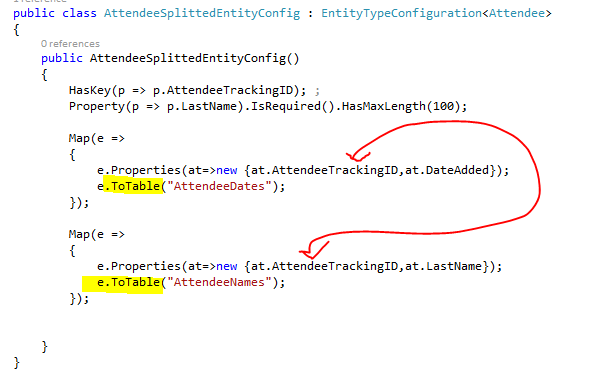




Данный способ конфигурации разделяет сущность на две таблицы



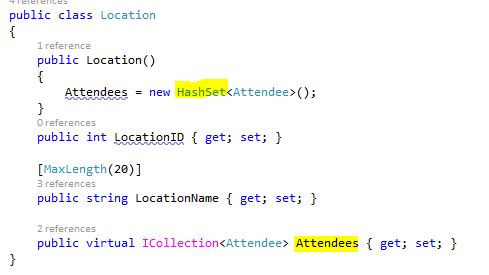
При данном подходе все поля сущности должны быть обязательно в той или иной таблице реализованы

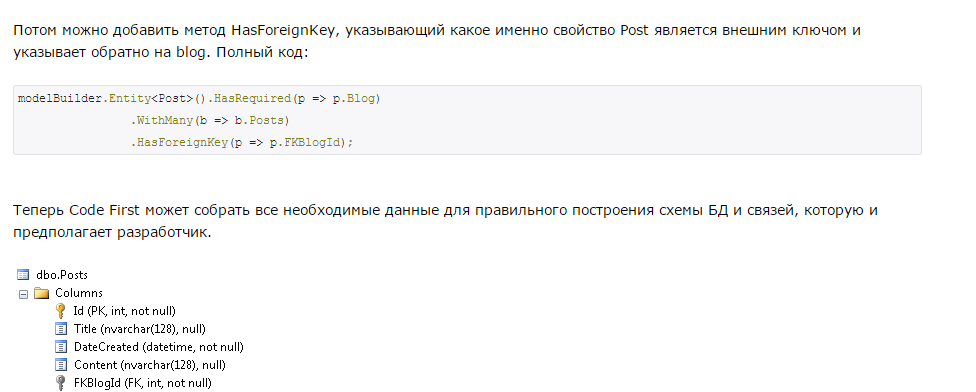


Создание внешнего ключа



Вторая сущность так же может содержать в себе коллекцию, которая будет знать кто на нее ссылается

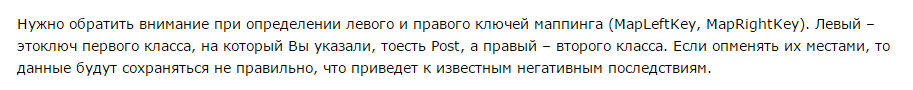




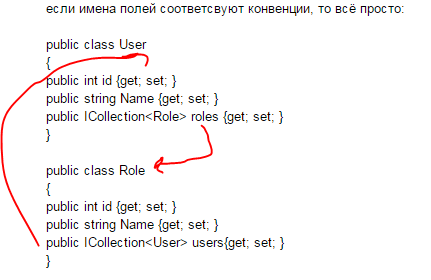
Осуществляется связь путем получения ссылки



Многие ко многим

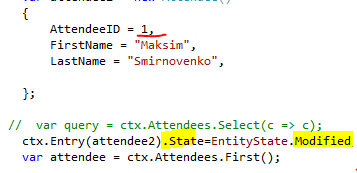
 

или

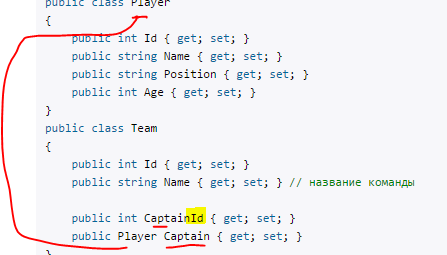


Обновление записи

Обязательно указывать ID записи



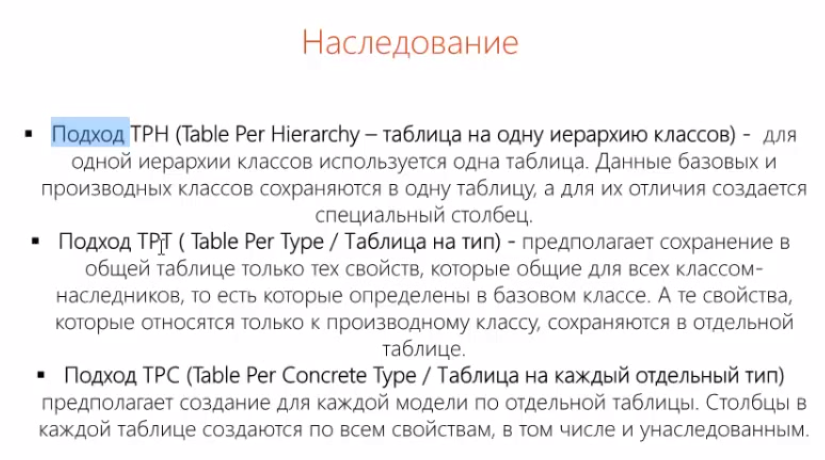
Один к одному



Подключение базы данных



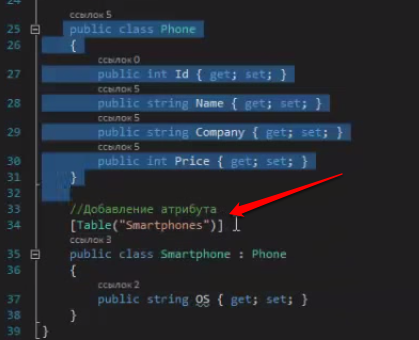
Наследование

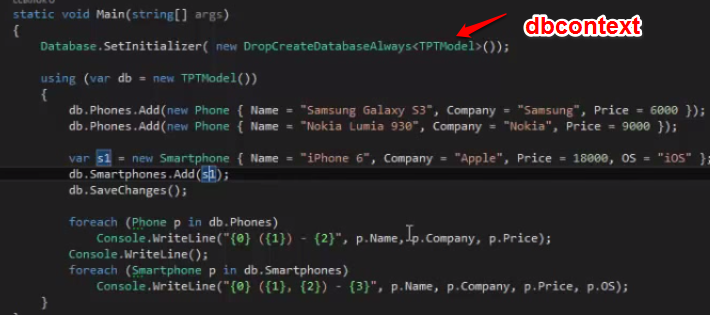


При первом случаи обычное наследование, в таблице базы данных просто создается столбец

Description где различие происходит по имени класса Именем класса

Во втором случаи при наследовании класс оборачиваем в атрибут

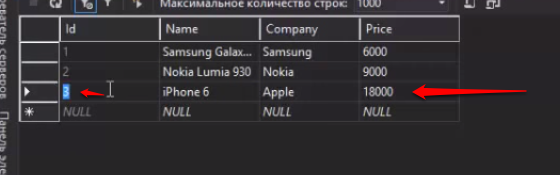




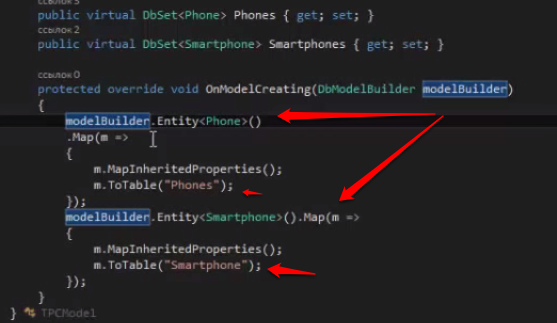
Создала еще одну таблицу



Связь по Id

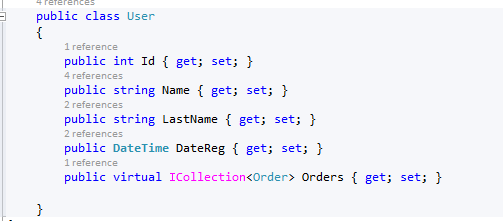


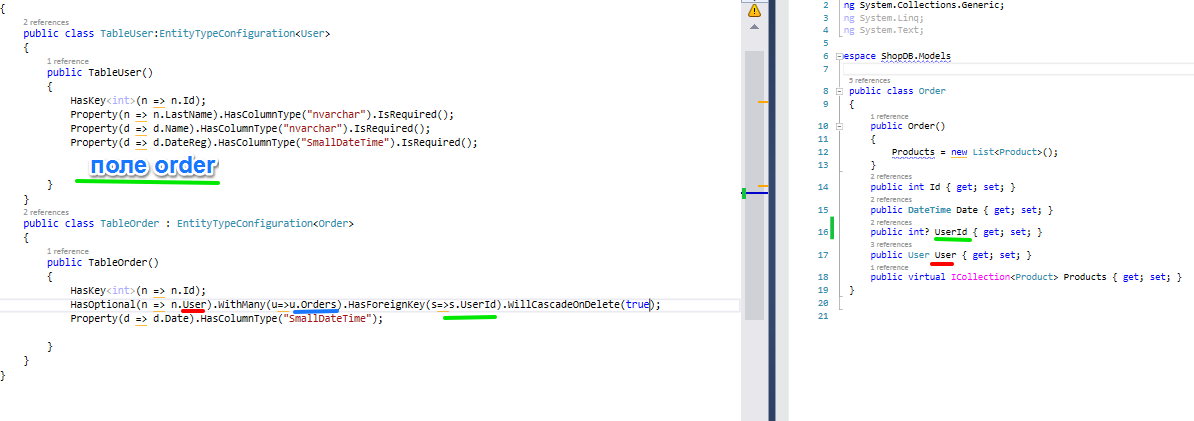
Третий способ создание отдельной таблицы для каждого класса

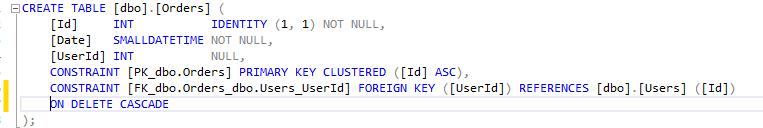


ОДИН КО МНОГИМ

С каскадным удалением

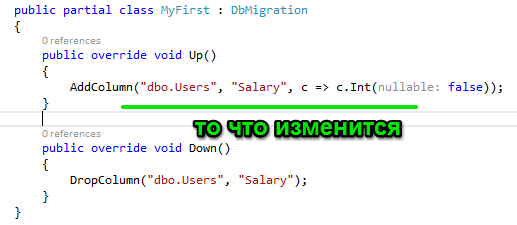






Добавить миграцию

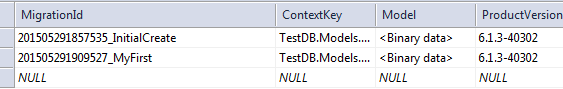


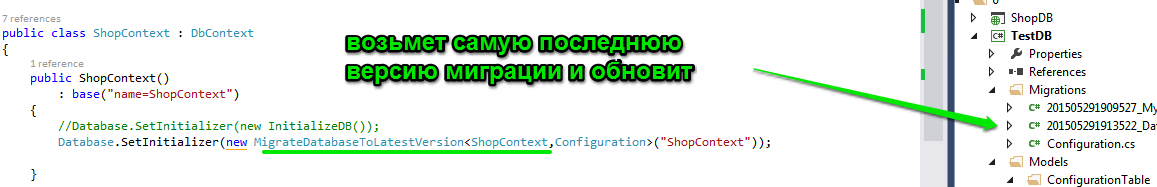


После изменяем бД Update-Database –Verbose

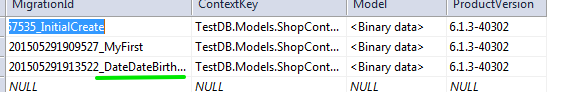
Вернуться к версии миграции Update-Database –TargetMigration:MyFirst

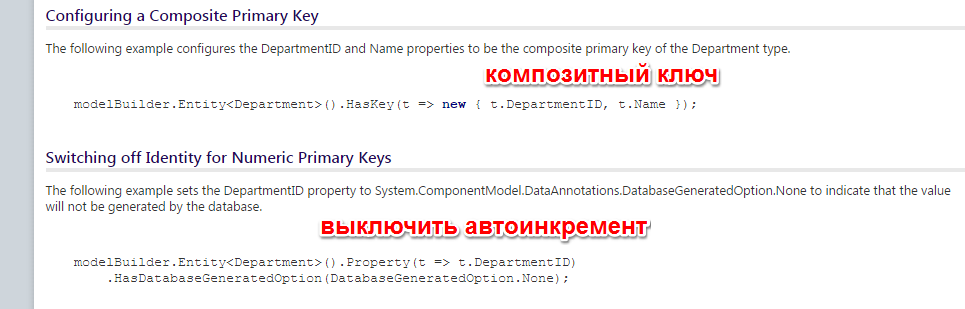
если нужно вернуться к полностью пустой базе данных, можно воспользоваться командой **Update-Database –TargetMigration: $InitialDatabase**





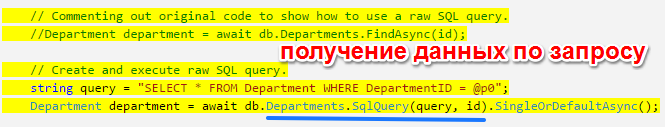
Результат



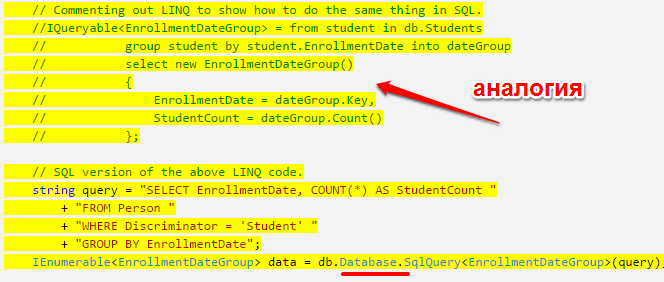


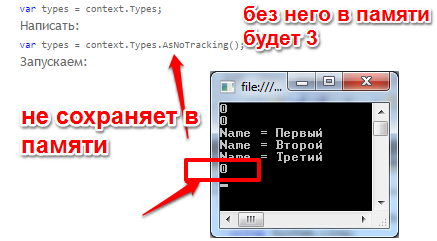


Извлекаем данные для класса Department SqlQuery



Можно и для БД





TimeStamp

