Бесплатная электронная книга

УЧУСЬ

Entity Framework Core

Free unaffiliated eBook created from
Stack Overflow contributors.

#entity-framework-core

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: entity-framework-core

It is an unofficial and free Entity Framework Core ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Entity Framework Core.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с ядром Entity Framework

замечания

Ядро Entity Framework (EF) Core - это легкая и расширяемая версия популярной технологии доступа к данным Entity Framework.

EF Core - объектно-реляционный картограф (O / RM), который позволяет разработчикам .NET работать с базой данных с использованием объектов .NET. Это устраняет необходимость в большей части кода доступа к данным, который разработчикам обычно приходится писать.

Examples

Добавление пакетов в проект

Чтобы добавить EntityFrameworkCore в ваш проект, обновите файл `project.json` (добавьте новые строки в разделы `dependencies` и `tools`):

```
"dependencies": {
    ...
    "Microsoft.EntityFrameworkCore.SqlServer": "1.0.0",
    "Microsoft.EntityFrameworkCore.SqlServer.Design": "1.0.0",
    "Microsoft.EntityFrameworkCore.Design": {
      "version": "1.0.0",
      "type": "build"
    },
},
"tools": {
    ...
    "Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final"
}
```

Не забудьте запустить `dotnet restore` чтобы загрузить эти пакеты из Интернета.

Если вы используете СУРБД, `Npgsql.EntityFrameworkCore.PostgreSQL` от Microsoft SQLServer, замените `Microsoft.EntityFrameworkCore.SqlServer` с правильной версией (`Microsoft.EntityFrameworkCore.Sqlite` , `Npgsql.EntityFrameworkCore.PostgreSQL` или другой - обратитесь к документации по РСУБД для рекомендуемого пакета).

База данных First in Entity Framework Core с библиотекой классов и SQL Server

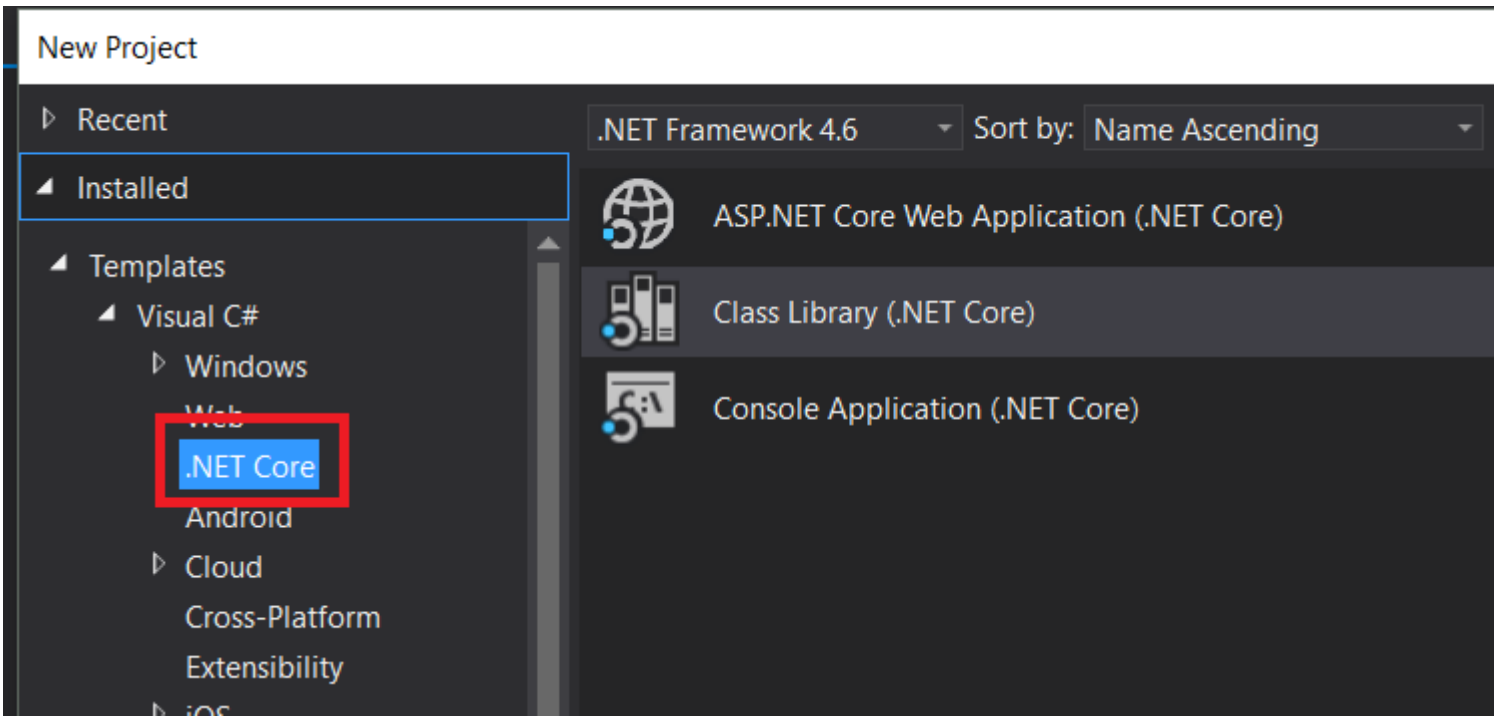
Хорошо, мне потребовался день, чтобы понять это, поэтому здесь я публикую шаги,

которые я выполнил, чтобы получить мою базу данных, впервые работающую в `Class Project (.NET Core)` , с .NET Core Web App.

Шаг 1 - Установите .NET Core

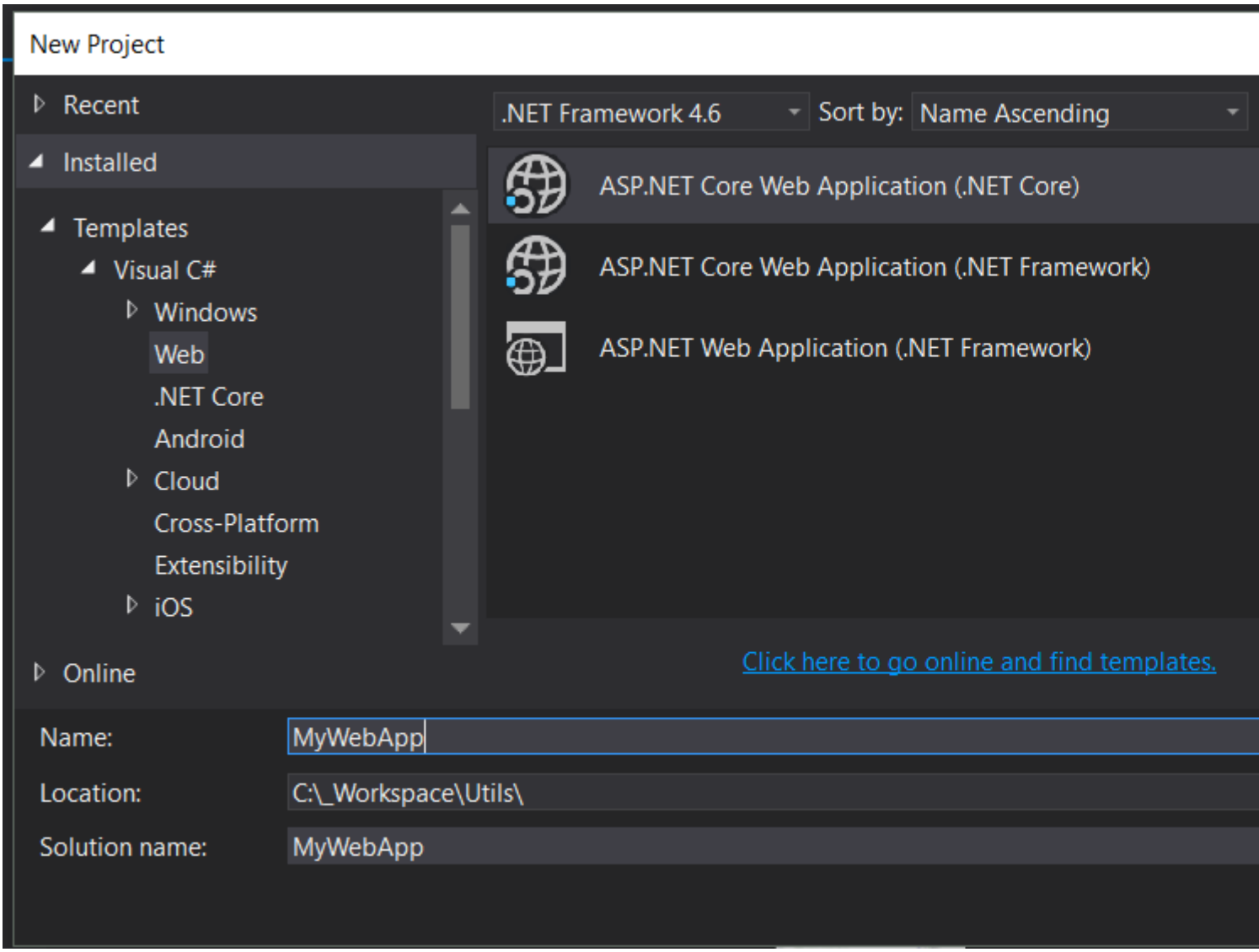
Убедитесь, что вы используете .NET Core, а не DNX `(Hint: You should be able to see the .NET Core option when creating a New Project)` - если NOT Download from Here

Если у вас возникли проблемы с установкой .NET Core (ошибка - это что-то вроде обновления Visual Studio 2015 Update 3 не установлена правильно). Вы можете запустить установку с помощью команды: [`DotNetCore.1.0.0-VS2015Tools.Preview2.exe SKIP_VSU_CHECK=1`] - Это предотвратит установку, выполняемую Visual Studio Check Github Issue

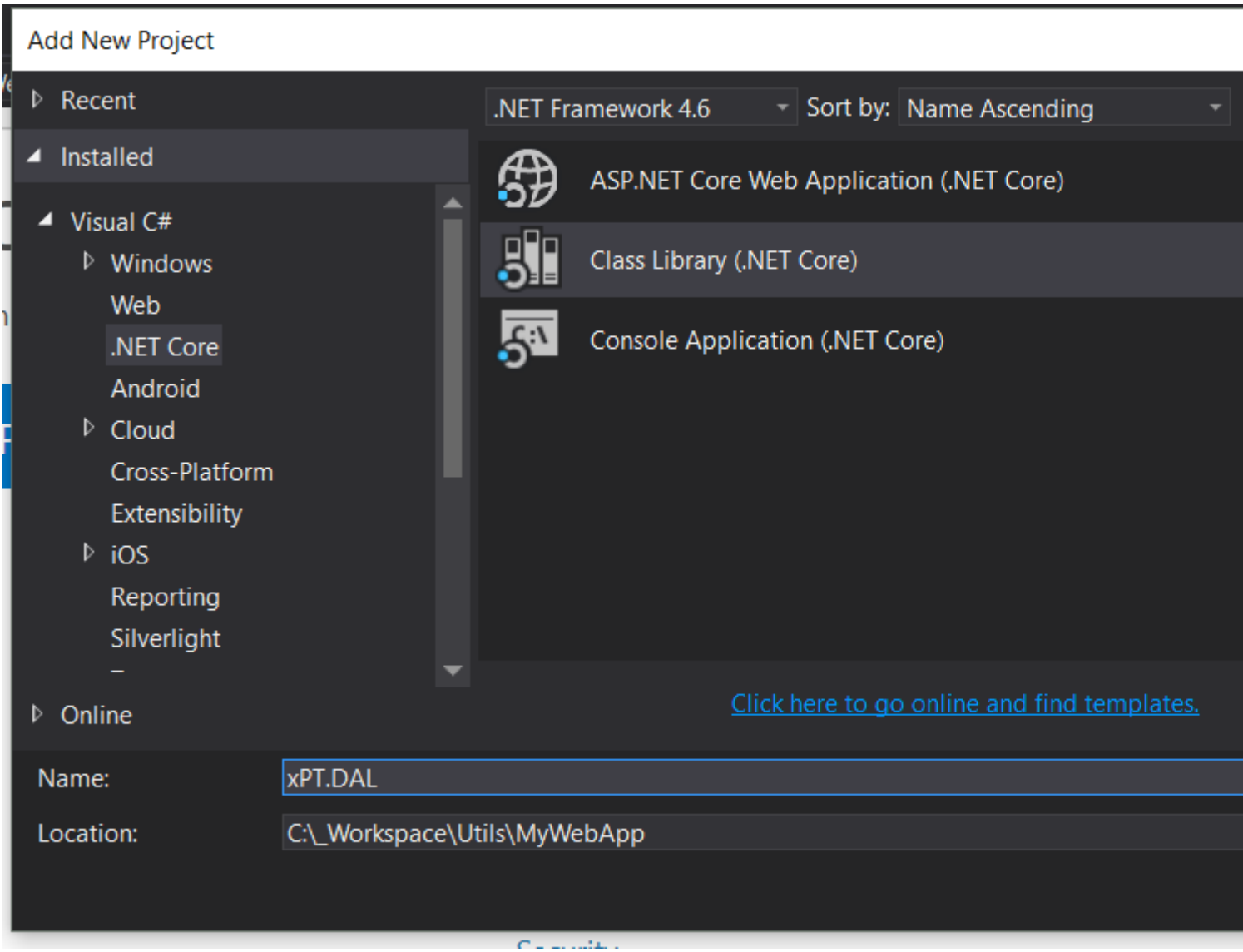


Шаг 2 - Создание проектов

Создайте новое веб-приложение ASP.NET Core -> Затем выберите веб-приложение на следующем экране



Добавить проект `Class Library (.NET Core)`



Шаг 3 - Установка пакетов EF

Откройте файл `project.json` библиотеки классов и вставьте следующее, затем сохраните файл:

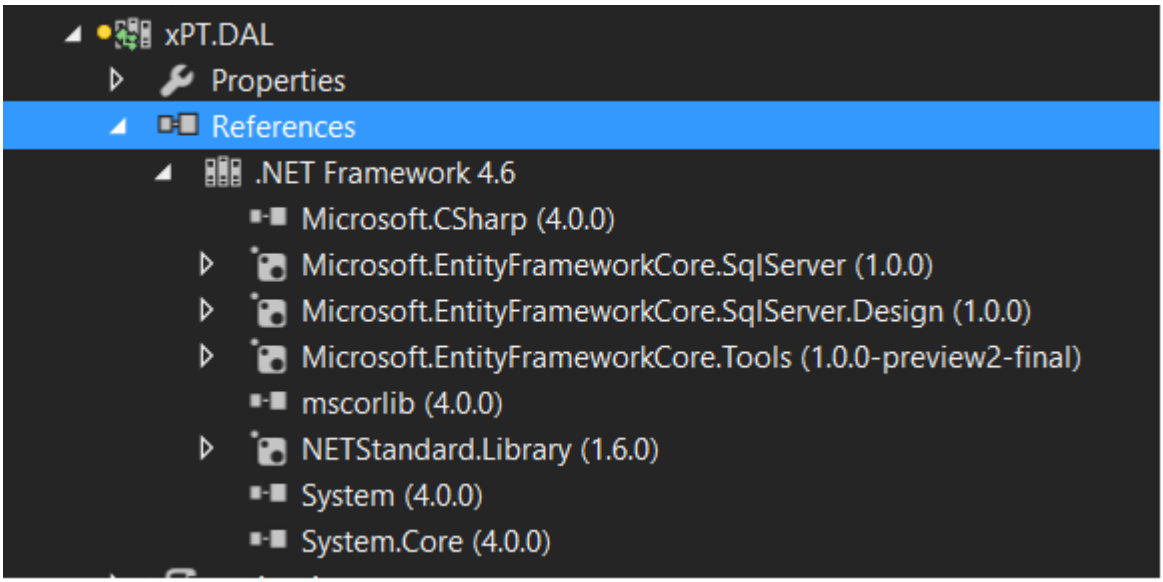
```
{
  "version": "1.0.0-*",

  "dependencies": {
    "Microsoft.EntityFrameworkCore.SqlServer": "1.0.0",
    "Microsoft.EntityFrameworkCore.SqlServer.Design": "1.0.0",
    "Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final",
    "NETStandard.Library": "1.6.0"
  },
  "tools": {
    "Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final"
  },

  "frameworks": {
    "net46": {
    },
```

```
    "netcoreapp1.0": {
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.0.0-*"
        }
      }
    }
  }
}
```

Это должно восстановить пакеты в разделе « `References`



---------- ИЛИ ЖЕ

Вы можете установить их с помощью Nuget Package Manager, выполнив следующие команды в консоли диспетчера пакетов

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.Tools -Pre

Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design
```

Примечание. Установите один пакет за раз - если вы получили сообщение об ошибке после установки

```
Microsoft.EntityFrameworkCore.Tools
```

Затем измените содержимое раздела `project.json` frameworks на следующее:

```
  "frameworks": {
    "net46": {
    },
    "netcoreapp1.0": {
```

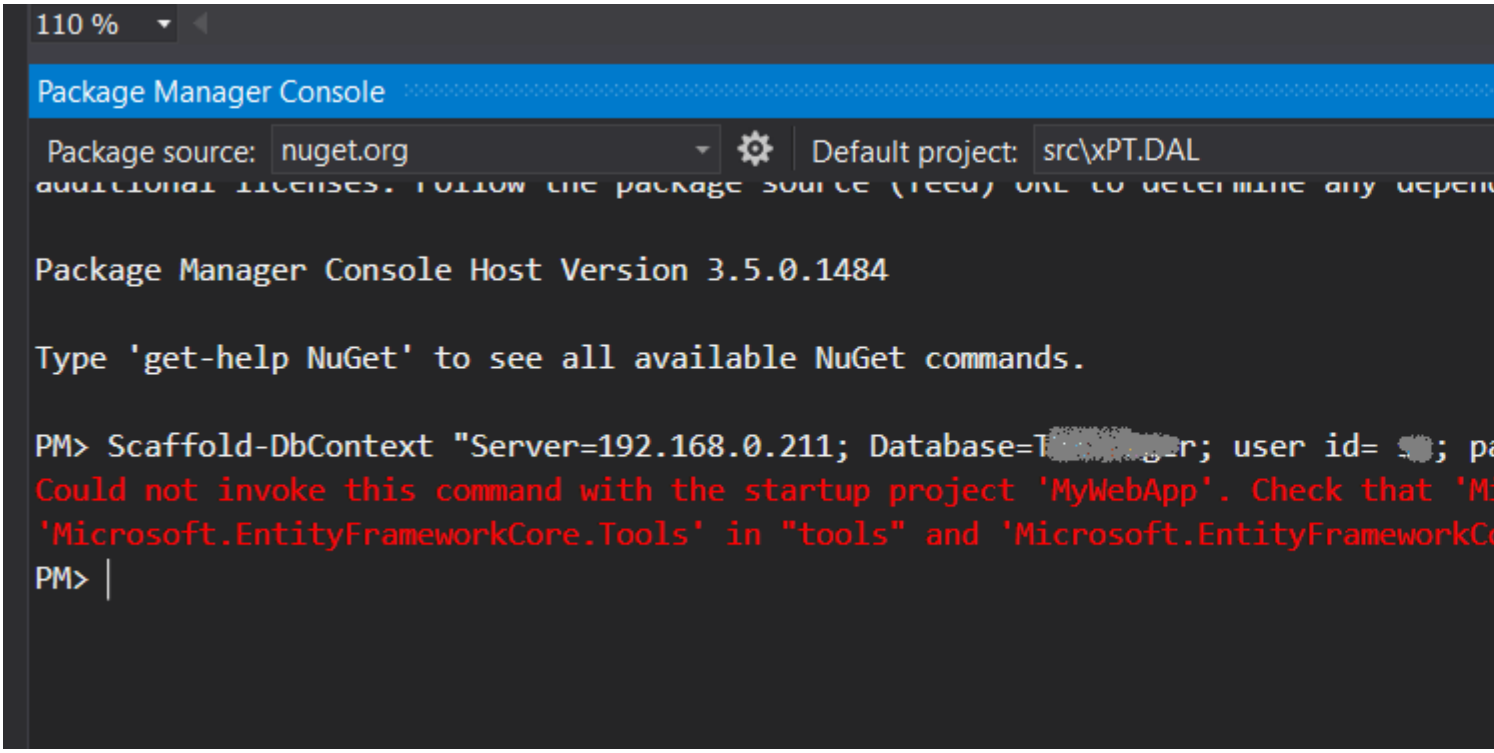
```
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.0.0-*"
        }
      }
    }
  }
```

Шаг 4 - Создание модели базы данных

Теперь для создания базы данных выполните следующую команду в `Package Manager Console` (НЕ забывайте изменить строку подключения в своей базе данных)

```
Scaffold-DbContext "Server=. ; Database=DATABASE; user id= USER ; password = PASSWORD;"
Microsoft.EntityFrameworkCore.SqlServer
```

Это даст вам ошибку о стартовом проекте:



Для этого вам нужно добавить те же ссылки, которые вы добавили в Class Library, в .NET Web App

Поэтому откройте свой `project.json` для веб-приложения,

В `dependencies` добавьте:

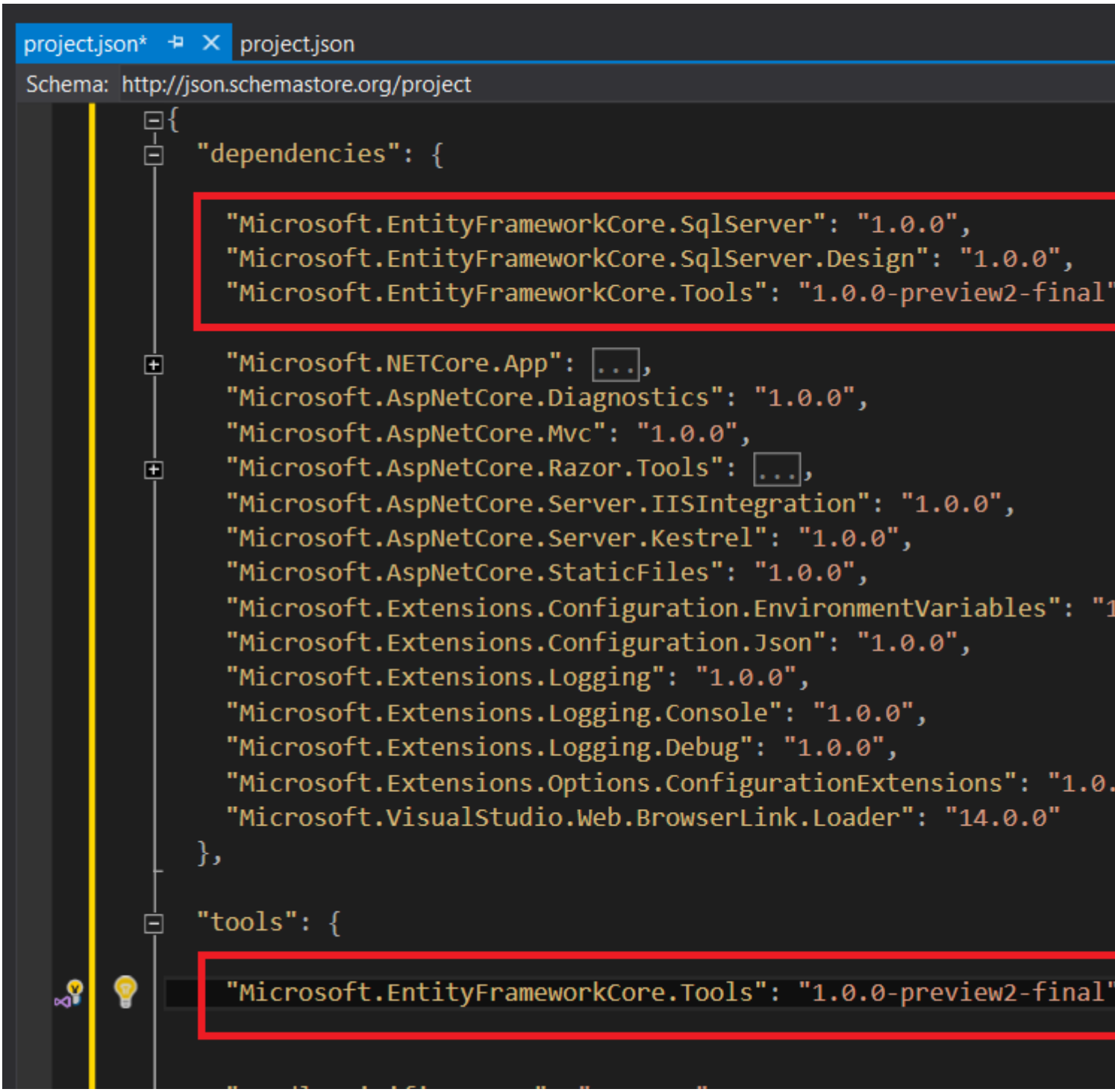
```
"Microsoft.EntityFrameworkCore.SqlServer": "1.0.0",
"Microsoft.EntityFrameworkCore.SqlServer.Design": "1.0.0",
"Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final",
```

и под `tools` добавить:

```
"Microsoft.EntityFrameworkCore.Tools": "1.0.0-preview2-final",
```

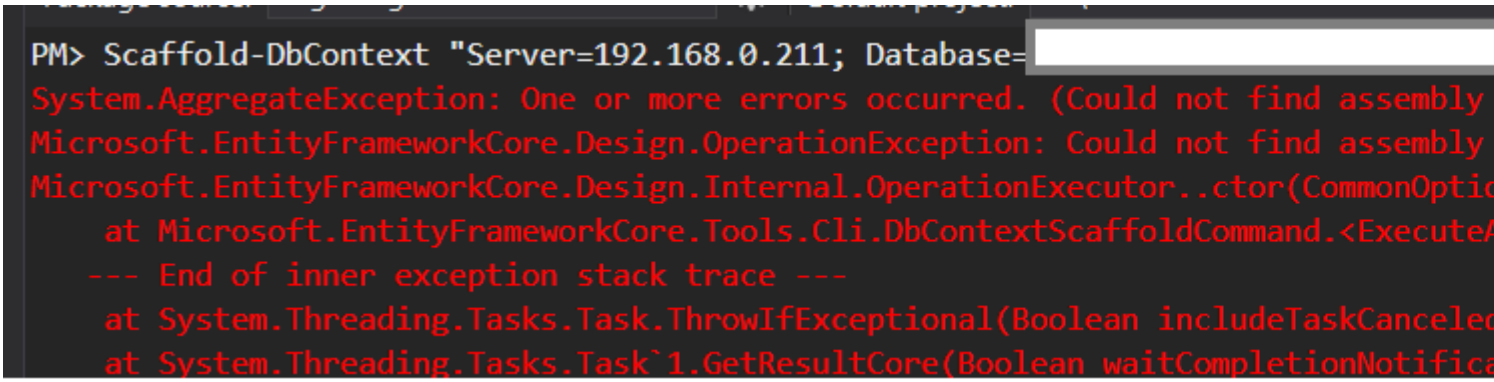
После внесения изменений Сохраните файл.

Это то, что выглядит мой project.json

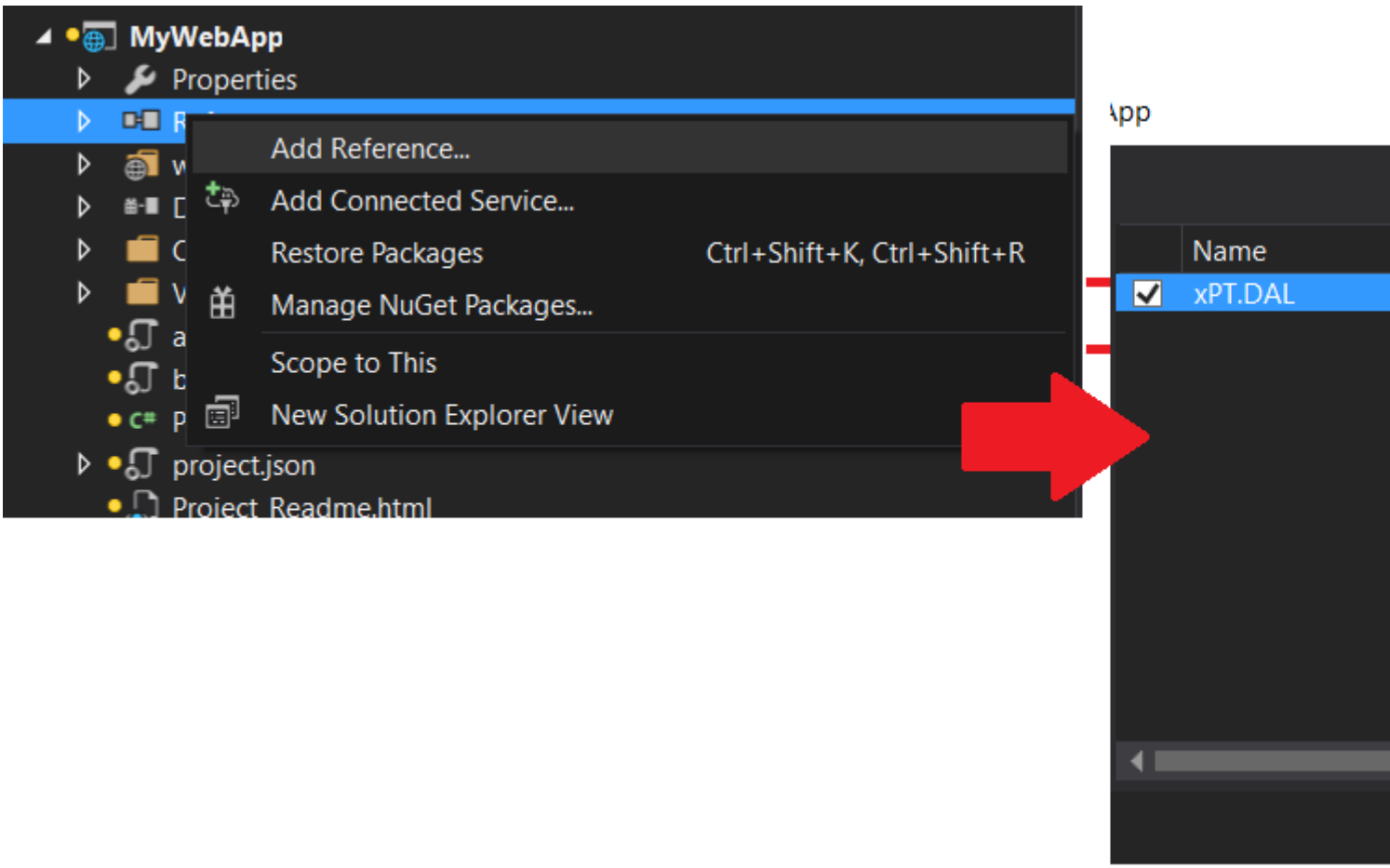


Затем снова запустите команду в консоли диспетчера пакетов против библиотеки классов:

Если вы еще не добавили ссылку на свою библиотеку классов в веб-приложение, вы получите эту ошибку:



для решения этой дополнительной ссылки вашей библиотеки классов на ваше веб-приложение:

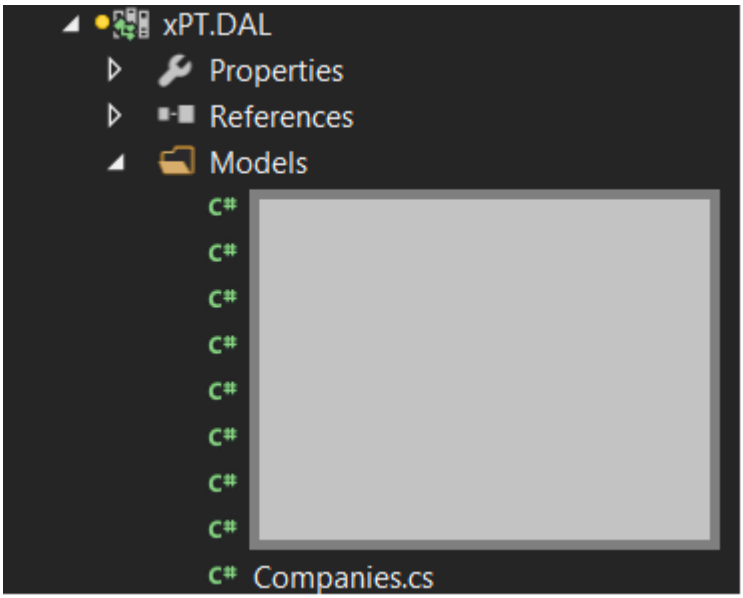


в заключение

Запустите команду еще раз - в `Package Manager Console` :

```
Scaffold-DbContext "Server=. ; Database=DATABASE; user id= USER ; password = PASSWORD;"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

Это должно создать объекты в папке моделей, в библиотеке классов



Передача строки соединения

В моем случае здесь у нас есть приложение Multi Tenant, в котором каждый клиент имеет свою собственную базу данных, например Client_1, Client_2, Client_3. Поэтому строка подключения должна быть динамической.

Таким образом, мы добавили свойство строки соединения в конструктор и передали его в контекст в методе `OnConfiguring`

```
public partial class ClientContext
{
    private readonly string _connectionString;

    public ClientContext(string connectionString) : base()
    {
        _connectionString = connectionString;
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(_connectionString);
    }
}
```

и использовал его так:

```
    public void TestConnection()
    {
        var clientId = 1;

        var connectionString = string.Format("Server=192.168.0.211; Database=Client_{0}; user
id= USER; password = PWD;", clientId);

        using (var clientContext = new ClientContext(connectionString))
        {
```

```
            var assets = clientContext.Users.Where(s => s.UserId == 1);
        }
    }
```

Модель, запрос и сохранение данных

модель

С EF Core доступ к данным осуществляется с использованием модели. Модель состоит из классов сущностей и производного контекста, представляющего сеанс с базой данных, позволяющий запрашивать и сохранять данные.

Вы можете создать модель из существующей базы данных, указать код модели для вашей базы данных или использовать EF Migrations для создания базы данных из вашей модели (и ее эволюцию по мере изменения вашей модели с течением времени).

```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;

namespace Intro
{
    public class BloggingContext : DbContext
    {
        public DbSet<Blog> Blogs { get; set; }
        public DbSet<Post> Posts { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=MyDatabase;Trusted_Connection=True

        }
    }

    public class Blog
    {
        public int BlogId { get; set; }
        public string Url { get; set; }

        public List<Post> Posts { get; set; }
    }

    public class Post
    {
        public int PostId { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }

        public int BlogId { get; set; }
        public Blog Blog { get; set; }
    }
}
```

Запрос

Экземпляры классов сущностей извлекаются из базы данных с помощью Language Integrated Query (LINQ).

```
using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

Сохранение данных

Данные создаются, удаляются и изменяются в базе данных с использованием экземпляров ваших классов сущностей.

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

Удаление данных

Экземпляры классов сущностей извлекаются из базы данных с помощью Language Integrated Query (LINQ).

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Attach(blog);
    db.Blogs.Remove(blog);
    db.SaveChanges();
}
```

Обновление данных

Данные обновляются в базе данных с использованием экземпляров классов сущностей.

```
using (var db = new BloggingContext())
```

```
{
    var blog = new Blog { Url = "http://sample.com" };
    var entity = db.Blogs.Find(blog);
    entity.Url = "http://sample2.com";
    db.SaveChanges();
}
```

Прочитайте Начало работы с ядром Entity Framework онлайн: https://riptutorial.com/ru/entity-framework-core/topic/3796/начало-работы-с-ядром-entity-framework

глава 2: EF Core vs EF6.x

замечания

Для получения последних обновлений обратитесь к: Сравнение функций

Examples

Сравнение бок о бок

Следующая таблица сравнивает доступные функции (1) в EF Core и EF6.x.

Он предназначен для сопоставления на высоком уровне и не перечисляет каждую функцию, или пытается дать подробную информацию о возможных различиях между тем, как работает одна и та же функция.

Создание модели	EF6.x	EF Core 1.0.0
Основное моделирование (классы, свойства и т. Д.)	да	да
Условные обозначения	да	да
Пользовательские соглашения	да	частичный
Аннотации данных	да	да
Свободный API	да	да
Наследование: таблица на иерархию (TPH)	да	да
Наследование: таблица для каждого типа (TPT)	да	
Наследование: таблица для конкретного класса (TPC)	да	
Свойства состояния тени		да
Альтернативные клавиши		да
Много-ко-многим: с объединением	да	да
«Многие ко многим»: без объединения	да	
Генерация ключей: база данных	да	да

Создание модели	EF6.x	EF Core 1.0.0
Генерация ключей: клиент		да
Сложные / стоимостные типы	да	
Пространственные данные	да	
Графическая визуализация модели	да	
Графический редактор перетаскивания	да	
Формат модели: код	да	да
Формат модели: EDMX (XML)	да	
Обратная инженерная модель из базы данных: Командная строка		да
Обратная инженерная модель из базы данных: мастер VS	да	
Инкрементальная модель обновления из базы данных	да	

Запрос данных	EF6.x	EF Core 1.0.0
LINQ: простые запросы	стабильный	стабильный
LINQ: Умеренные запросы	стабильный	стабилизирующий
LINQ: сложные запросы	стабильный	В ходе выполнения
LINQ: запросы с использованием свойств навигации	стабильный	В ходе выполнения
«Довольно» генерация SQL	Бедные	да
Смешанная оценка клиент / сервер		да
Загрузка связанных данных: Eager	да	да
Загрузка связанных данных: Lazy	да	
Загрузка связанных данных: Явная	да	
Необработанные SQL-запросы: типы моделей	да	да
Необработанные SQL-запросы: не отображаемые	да	

Запрос данных	EF6.x	EF Core 1.0.0
типы		
Необработанные SQL-запросы: Сопоставление с LINQ		да

Сохранение данных	EF6.x	EF Core 1.0.0
Сохранить изменения	да	да
Отслеживание изменений: снимок	да	да
Отслеживание изменений: уведомление	да	да
Доступ к отслеживаемому состоянию	да	частичный
Оптимистический параллелизм	да	да
операции	да	да
Вычисление заявлений		да
Хранимая процедура	да	
Поддержка отдельного графика (N-Tier): API уровня низкого уровня	Бедные	да
Поддержка отдельного графика (N-Tier): от конца до конца		Бедные

Другие преимущества	EF6.x	EF Core 1.0.0
Миграции	да	да
API-интерфейсы создания / удаления базы данных	да	да
Данные семян	да	
Устойчивость соединения	да	
Перехваты жизненного цикла (события, командный перехват, ...)	да	

Поставщики баз данных	EF6.x	EF Core 1.0.0
SQL Server	да	да

Поставщики баз данных	EF6.x	EF Core 1.0.0
MySQL	да	Платный только, неоплачиваемый в ближайшее время (2)
PostgreSQL	да	да
оракул	да	Платный только, неоплачиваемый в ближайшее время (2)
SQLite	да	да
SQL Compact	да	да
DB2	да	да
InMemory (для тестирования)		да
Лазерное настольное хранилище		Прототип
Redis		Прототип

Модели приложений	EF6.x	EF Core 1.0.0
WinForms	да	да
WPF	да	да
Приставка	да	да
ASP.NET	да	да
Ядро ASP.NET		да
Xamarin		Скоро (3)
UWP		да

Примечания:

(1): По состоянию на 2016/10/18

(2): Платные провайдеры доступны, работают неоплачиваемые поставщики. Команды, работающие с неоплачиваемыми поставщиками, не предоставили публичные сведения о сроках и т. Д.

(3): EF Core построен для работы с Xamarin, когда поддержка .NET Standard включена в Xamarin.

Прочитайте EF Core vs EF6.x онлайн: https://riptutorial.com/ru/entity-framework-core/topic/7513/ef-core-vs-ef6-x

глава 3: Обновление отношения «Множество ко многим»

Вступление

Как обновить отношения Many to Many в EF Core:

Examples

Пример редактирования MVC POST

Скажем, у нас есть класс Product с несколькими цветами, который может быть на многих Продуктах.

```
public class Product
{
    public int ProductId { get; set; }
    public ICollection<ColorProduct> ColorProducts { get; set; }
}

public class ColorProduct
{
    public int ProductId { get; set; }
    public int ColorId { get; set; }

    public virtual Color Color { get; set; }
    public virtual Product Product { get; set; }
}

public class Color
{
    public int ColorId { get; set; }
    public ICollection<ColorProduct> ColorProducts { get; set; }
}
```

Используя это расширение, чтобы упростить его:

```
public static class Extensions
{
    public static void TryUpdateManyToMany<T, TKey>(this DbContext db, IEnumerable<T>
currentItems, IEnumerable<T> newItems, Func<T, TKey> getKey) where T : class
    {
        db.Set<T>().RemoveRange(currentItems.Except(newItems, getKey));
        db.Set<T>().AddRange(newItems.Except(currentItems, getKey));
    }

    public static IEnumerable<T> Except<T, TKey>(this IEnumerable<T> items, IEnumerable<T>
other, Func<T, TKey> getKeyFunc)
    {
        return items
```

```
            .GroupJoin(other, getKeyFunc, getKeyFunc, (item, tempItems) => new { item,
tempItems })
            .SelectMany(t => t.tempItems.DefaultIfEmpty(), (t, temp) => new { t, temp })
            .Where(t => ReferenceEquals(null, t.temp) || t.temp.Equals(default(T)))
            .Select(t => t.t.item);
    }
}
```

Обновление цветов продукта будет выглядеть так (MVC Edit POST Method)

```
[HttpPost]
public IActionResult Edit(ProductVm vm)
{
if (ModelState.IsValid)
    {
        var model = db.Products
            .Include(x => x.ColorProducts)
            .FirstOrDefault(x => x.ProductId == vm.Product.ProductId);

        db.TryUpdateManyToMany(model.ColorProducts, vm.ColorsSelected
            .Select(x => new ColorProduct
            {
                ColorId = x,
                ProductId = vm.Product.ProductId
            }), x => x.ColorId);

        db.SaveChanges();

        return RedirectToAction("Index");
    }
   return View(vm);
}


public class ProductVm
{
    public Product Product { get; set; }

    public IEnumerable<int> ColorsSelected { get; set; }
}
```

Код был упрощен настолько, насколько я могу, никаких дополнительных свойств для каких-либо классов.

Прочитайте Обновление отношения «Множество ко многим» онлайн: https://riptutorial.com/ru/entity-framework-core/topic/9527/обновление-отношения--множество-ко-многим-

кредиты

S. No	Главы	Contributors
1	Начало работы с ядром Entity Framework	Community, Dawood Awan, Dmitry, hasan, natemcmaster, NovaDev, tmg, uTeisT
2	EF Core vs EF6.x	Frédéric, Ruud Lenders, uTeisT
3	Обновление отношения «Множество ко многим»	Paw Ormstrup Madsen