ADO.NET

(Entity Framework Core: Queries and Updates)

Андрей Голяков

Вставка простых объектов

```
var customer = new Customer { Name = "Andrei" };
using (var context = new OnlineStoreContext())
{
    context.Add(customer);
}
```

Теперь контекст *отслеживает* объект customer.

Однако самой вставки пока не происходит.

Вставка простых объектов

```
var customer = new Customer { Name = "Andrei" };
using (var context = new OnlineStoreContext())
{
    context.Add(customer);
    context.SaveChanges();
}
```

Чтобы произошла вставка записи в БД, необходимо вызвать метод SaveChanges.

Что происходит внутри SaveChanges?

При выполнении метода SaveChanges контекст выполняет следующие шаги:

1. Проверяются все отслеживаемые контекстом объекты сущностей. Поскольку мы добавляли клиента (с помощью метода Add), контекст хранит информацию о том, что необходимо вставить новую запись в таблицу,
2. Подготавливается необходимый SQL-скрипт,
3. SQL-скрипт отправляется на выполнение в базу данных, причём обёрнутый в транзакцию.

Совместная работа

Демонстрация создания новой сущности Customer.

Самостоятельная работа

Создаём несколько новых сущностей Product используя методы Add и AddRange соответствующего DbSet-а.

Упражняемся в специально отведённом для этого месте — методе InsertProducts.

Самостоятельная работа (решение)

```
private static void InsertProducts()
{
     var product = new Product { Name = "Fenix 5 Plus Sapphire", Price = 73989.99M };
     using (var context = new OnlineStoreContext())
     {
          context.Products.Add(product);
          context.SaveChanges();
     }

     var products = new[]
     {
          new Product { Name = "Forerunner 645 Music", Price = 42199.99M },
          new Product { Name = "MARQ Aviator", Price = 208400 }
     };
     using (var context = new OnlineStoreContext())
     {
          context.AddRange(products);
          context.SaveChanges();
     }
}
```

Выборка данных с помощью EF Core

Особенность работы с Entity Framework заключается в использовании запросов LINQ (Language Integrated Query) для выборки данных из БД:

```
using (var context = new OnlineStoreContext())
{
    // select all the entities of type 'Customer'
    var customers = context.Customers.ToList();
}
```

С помощью LINQ строятся похожие на SQL-запросы обращения к БД для извлечения данных в виде объектов.

Два способа писать LINQ-запросы

LINQ-методы:

```
var allCustomers = context
      .Customers
      .ToList();
```

```
var severalCustomers = context
      .Customers
      .Where(c => c.Name == "Andrei")
      .ToList();
```

Синтаксис LINQ-запросов:

```
var allCustomersLinqSyntax = (
      from c
      in context.Customers
      select c
).ToList();
```

```
var severalCustomersLinqSyntax = (
      from c
      in context.Customers
      where c.Name == "Andrei"
      select c
).ToList();
```

Фильтрация объектов при выборке из БД

Если искомое значение вставляется константой прямо в лямбда-выражение:

```
...Where(c => c.Name == "Andrei")
```

… в SQL-команду параметр НЕ добавляется:

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE [c].[Name] = 'Andrei'
```

Если искомее значение передаются в виде параметра:

```
var name = "Andrei";

...Where(c => c.Name == name)
```

… в SQL-команду добавляется параметр:

```
@parameter = 'Andrei'

SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE [c].[Name] = @parameter
```

LINQ to Entities Execution Methods

```
ToList()
First()   FirstOrDefault()
Single()  SingleOrDefault()
Last()*   LastOrDefault()*
Count()   LongCount()
Min()  Max()  Average()
```

Не LINQ-метод, однако DbSet его выполнит: `Find(keyValue)`

* Методы `Last` требуют, чтобы в запросе был `OrderBy`-метод, иначе из БД сначала вычитаются все данные, а потом вернётся последний элемент

Обновление объектов в БД

Для отслеживаемых контекстом объектов обновление произойдёт автоматически при вызове метода SaveChanges:

```
var customer = _context.Customers.First();
customer.Name = "Mr. " + customer.Name;
_context.SaveChanges();
```

Для обновления неотслеживаемых объектов необходимо сначала вызвать метод Update:

```
var product = _context.Products.First();
product.Price *= 0.1M;

using (var newContextInstance = new OnlineStoreContext())
{
     newContextInstance.Products.Update(product);
     newContextInstance.SaveChanges();
}
```

Удаление объектов из БД

Для удаления необходимо иметь объект целиком, одного идентификатора недостаточно (несмотря на то, что в самом SQL-запросе ничего кроме идентификатора не фигурирует)!

```
var customer = _context.Customers.FirstOrDefault(c => c.Id == 1);
if (customer != null)
{
     _context.Customers.Remove(customer);
     _context.SaveChanges();
}
```

Если очень хочется сделать удаление оптимальнее, можно вызвать необходимую SQL-команду напрямую:

```
_context.Database.ExecuteSqlCommand(
     "EXEC DELETE FROM [dbo].[Customers] WHERE Id = {0}", 1);
```

Домашняя работа

Попробовать самостоятельно реализовать библиотеку слоя Reminder.Storage.SqlServer.EF данных для чат-бота (по сути интерфейс `IReminderStorage`) на базе SQL Server не через примитивы ADO.NET Core, а через Entity Framework Core.

Для самопроверки рекомендуется также написать библиотеку с тестами Reminder.Storage.SqlServer.EF.Tests.

Спасибо за внимание.