Основы Entity Framework

Основные операции с данными

Последнее обновление: 27.09.2016

Большинство операций с данными представляют собой CRUD-операции (Create, Read, Update, Delete), то есть получение данных, создание, обновление и удаление. Entity Framework позволяет легко производить данные операции.

Для примеров с операциями возьмем простенькую модель Phone:

```
public class Phone
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
}
```

И следующий класс контекста данных:

```
public class PhoneContext : DbContext
{
    public PhoneContext() : base("DefaultConnection")
    { }

    public DbSet<Phone> Phones { get; set; }
}
```

Добавление

Для добавления применяется метод `Add()` у объекта DbSet:

```
using (PhoneContext db = new PhoneContext())
{
    Phone p1 = new Phone { Name = "Samsung Galaxy S7", Price = 20000 };
    Phone p2 = new Phone { Name = "iPhone 7", Price = 28000 };

        // добавление
    db.Phones.Add(p1);
    db.Phones.Add(p2);
    db.SaveChanges();   // сохранение изменений

    var phones = db.Phones.ToList();
    foreach (var p in phones)
        Console.WriteLine("{0} - {1} - {2}", p.Id, p.Name, p.Price);
}
```

После добавления надо сохранить все изменения с помощью метода `SaveChanges()`.

Консольный вывод:

```
1 - Samsung Galaxy S7 - 20000
1 - iPhone 7 - 28000
```

Редактирование

Контекст данных способен отслеживать изменения объектов, поэтому для редактирования объекта достаточно изменить его свойства и после этого вызвать метод SaveChanges():

```
using (PhoneContext db = new PhoneContext())
{
        // получаем первый объект
    Phone p1 = db.Phones.FirstOrDefault();

    p1.Price = 30000;
    db.SaveChanges();   // сохраняем изменения
}
```

Но рассмотрим другую ситуацию:

```
Phone p1;
using (PhoneContext db = new PhoneContext())
{
    p1 = db.Phones.FirstOrDefault();
}

using (PhoneContext db = new PhoneContext())
{
    if(p1!=null)
    {
        p1.Price = 60000;
        db.SaveChanges();
    }
}
```

Так как объект Phone получен в одном контексте, а изменяется для другого контекста, который его не отслеживает. В итоге изменения не сохранятся. Чтобы изменения сохранились, нам явным образом надо установить для его состояния значение **EntityState.Modified**:

```
using (PhoneContext db = new PhoneContext())
{
    if(p1!=null)
    {
        p1.Price = 60000;
                db.Entry(p1).State = EntityState.Modified;
        db.SaveChanges();
    }
}
```

Удаление

Для удаления объекта применяется метод **Remove()** объекта DbSet:

```
using (PhoneContext db = new PhoneContext())
{
        Phone p1 = db.Phones.FirstOrDefault();
    if(p1!=null)
    {
                db.Phones.Remove(p1);
        db.SaveChanges();
    }
}
```

Но как и в случае с обновлением здесь мы можем столкнуться с похожей проблемой, когда объект получаем из базы данных в пределах одного контекста, а пытаемся удалить в другом контексте. И в этом случае нам надо установить вручную у объекта состояние **EntityState.Deleted**:

```
using (PhoneContext db = new PhoneContext())
{
    if(p1!=null)
    {
                db.Entry(p1).State = EntityState.Deleted;
        db.SaveChanges();
    }
}
```

Метод Attach

Если объект получен в одном контексте, а сохраняется в другом, то мы можем устанавливать у него вручную состояния EntityState.Updated или EntityState.Deleted. Но есть еще один способ: с помощью метода **Attach** у объекта DbSet мы можем прикрепить объект к текущему контексту данных:

```
Phone p1;
using (PhoneContext db = new PhoneContext())
{
    p1 = db.Phones.FirstOrDefault();
}
// редактирование
using (PhoneContext db = new PhoneContext())
{
    if(p1!=null)
    {
        db.Phones.Attach(p1);
        p1.Price = 999;
        db.SaveChanges();
    }
}
// удаление
using (PhoneContext db = new PhoneContext())
{
    if(p1!=null)
    {
        db.Phones.Attach(p1);
        db.Remove(p1);
        db.SaveChanges();
    }
}
```

Назад **Содержание** **Вперед**