# Конфигурация моделей

**Последнее обновление: 26.09.2016**

С помощью Fluent API мы можем задать многочисленные настройки для моделей и их свойств. Однако, если настроек очень много, то они могут утяжелять класс контекста. И для вынесения их во вне можно использовать конфигурации, которые представлены классом **EntityTypeConfiguration<T>**.

К примеру, пусть у нас есть следующий класс контекста:

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

    public DbSet<Company> Companies { get; set; }
    public DbSet<Phone> Phones { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Phone>()
            .ToTable("Mobiles").HasKey(p => p.Ident);
        modelBuilder.Entity<Phone>()
            .Property(p => p.Name).IsRequired().HasMaxLength(30);

        modelBuilder.Entity<Company>().ToTable("Manufacturers")
            .Property(c => c.Name).IsRequired().HasMaxLength(30);
    }
}
public class Company
{
    public int Id { get; set; }
    public string Name { get; set; }

    public ICollection<Phone> Phones { get; set; }
    public Company()
    {
        Phones = new List<Phone>();
    }
}

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

    public int CompanyId { get; set; }
```

```
    public Company Company { get; set; }
}
```

Вся конфигурация здесь определена в методе `OnModelCreating()`. В принципе он не содержит много кода, однако при наличии гораздо большего количества сущностей и более изощренной их конфигурации с помощью Fluen API данный метод мог бы сильно раздуться в размерах. И теперь изменим определение контекста, применив конфигурации:

```
using System.Data.Entity.ModelConfiguration;


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

    public DbSet<Company> Companies { get; set; }
    public DbSet<Phone> Phones { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Configurations.Add(new PhoneConfiguration());
        modelBuilder.Configurations.Add(new CompanyConfiguration());
    }
}
public class PhoneConfiguration : EntityTypeConfiguration<Phone>
{
    public PhoneConfiguration()
    {
        ToTable("Mobiles").HasKey(p => p.Ident);
        Property(p => p.Name).IsRequired().HasMaxLength(30);
    }
}
public class CompanyConfiguration : EntityTypeConfiguration<Company>
{
    public CompanyConfiguration()
    {
        ToTable("Manufacturers").Property(c => c.Name).IsRequired().HasMaxLength(30);
    }
}
```

Теперь конфигурация моделей вынесена в отдельные классы. А для добавления конкретных конфигураций в контекст используется метод `modelBuilder.Configurations.Add()`, которому передается нужный объект конфигурации. В итоге по своему действию первый и второй варианты контекста будут идентичны.

**Назад** **Содержание** **Вперед**