Связь один-к-одному

Последнее обновление: 31.10.2015

Строго говоря в Entity Framework нет как таковой связи один-к-одному, так как ожидается, что обработчик будет использовать связь один-ко-многим. Но все же нередко возникает потребность в наличие подобной связи между объектами в приложении, и в Entity Framework мы можем настроить данный тип отношений.

Рассмотрим стандартный пример подобных отношений: есть класс пользователя User, который хранит логин и пароль, то есть данные учетных записей. А все данные профиля, такие как имя, возраст и так далее, выделяются в класс профиля UserProfile.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
//..............................

public class User
{
    public int Id { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }

    public UserProfile Profile { get; set; }
}

public class UserProfile
{
    [Key]
    [ForeignKey("User")]
    public int Id { get; set; }

    public string Name { get; set; }
    public int Age { get; set; }

    public User User { get; set; }
}
```

В этой связи между классами класс UserProfile является дочерним или подчиненным по отношению к классу User. И чтобы установить связь одни к одному, у подчиненного класса устанавливается свойство идентификатора, которое называется также, как и идентификатор в основном классе. То есть в классе User свойство называется Id, то и в UserProfile также свойство называется Id. Если бы в классе User свойство называлось бы UserId, то такое же название должно было быть и в UserProfile.

И в классе UserProfile над этим свойством Id устанавливаются два атрибута: `[Key]`, который показывает, то это первичный ключ, и `[ForeignKey]`, который показывает, что это также и внешний ключ. Причем внешний ключ к таблице объектов User.

Соответственно классы User и UserProfile имеют ссылки друг на друга.

В классе контекста определяются свойства для взаимодействия с талицами в бд:

```
public class UserContext : DbContext
{
    public DbSet<User> Users { get; set; }
    public DbSet<UserProfile> UserProfiles { get; set; }
}
```

Для этих классов контекст данных будет создавать следующую таблицу UserProfiles:

```
CREATE TABLE [dbo].[UserProfiles] (
    [Id]   INT            NOT NULL,
    [Name] NVARCHAR (MAX) NULL,
    [Age]  INT            NOT NULL,
    CONSTRAINT [PK_dbo.UserProfiles] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.UserProfiles_dbo.Users_Id] FOREIGN KEY ([Id]) REFERENCES [dbo].[Users] ([Id])
);
```

Посмотрим, как работать с моделями с такой связью. Добавление и получение:

```
using(UserContext db = new UserContext())
{
    User user1 = new User { Login = "login1", Password = "pass1234" };
    User user2 = new User { Login = "login2", Password = "5678word2" };
    db.Users.AddRange(new List<User> { user1, user2 });
    db.SaveChanges();
    UserProfile profile1 = new UserProfile { Id = user1.Id, Age = 22, Name = "Tom" };
    UserProfile profile2 = new UserProfile { Id = user2.Id, Age = 27, Name = "Alice" };
    db.UserProfiles.AddRange(new List<UserProfile> { profile1, profile2 });
    db.SaveChanges();

    foreach(User user in db.Users.Include("Profile").ToList())
        Console.WriteLine("Name: {0}  Age: {1}  Login: {2}  Password: {3}",
                user.Profile.Name, user.Profile.Age, user.Login, user.Password);
}
```

Редактирование:

```
using (UserContext db = new UserContext())
{
    User user1 = db.Users.FirstOrDefault();
    if(user1!=null)
    {
        user1.Password = "dsfvbggg";
        db.Entry(user1).State = EntityState.Modified;
        db.SaveChanges();
    }

    UserProfile profile2 = db.UserProfiles.FirstOrDefault(p => p.User.Login == "login2");
    if(profile2!=null)
    {
        profile2.Name = "Alice II";
        db.Entry(profile2).State = EntityState.Modified;
        db.SaveChanges();
    }
}
```

При удалении надо учитывать следующее: так как объект UserProfile требует наличие объекта User и зависит от этого объекта, то при удалении связанного объекта User надо будет удалить и связанный с ним объект UserProfile. Поскольку по молчанию у нас не предусмотрено каскадное даление при данной связи. Если же будет удален объект UserProfile, на объект User это никак не повлияет:

```
using (UserContext db = new UserContext())
{
    User user1 = db.Users.Include("Profile").FirstOrDefault();
```

```
    if(user1!=null)
    {
        db.UserProfiles.Remove(user1.Profile);
        db.Users.Remove(user1);
        db.SaveChanges();
    }

    UserProfile profile2 = db.UserProfiles.FirstOrDefault(p => p.User.Login == "login2");
    if(profile2!=null)
    {
        db.UserProfiles.Remove(profile2);
        db.SaveChanges();
    }
}
```

Назад **Содержание** **Вперед**

Контакты для связи: metanit22@mail.ru