

یکی از مزایای مهم استفاده از Entity framework، خواص راهبری (navigation properties) آن هستند که امکان تهیه کوئری‌های بین جداول را به سادگی و به نحوی منطقی فراهم می‌کنند. برای مثال دو جدول شهرها و افراد را در نظر بگیرید. مقصود از تعریف جدول شهرها در اینجا، مشخص سازی محل تولد افراد است:

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

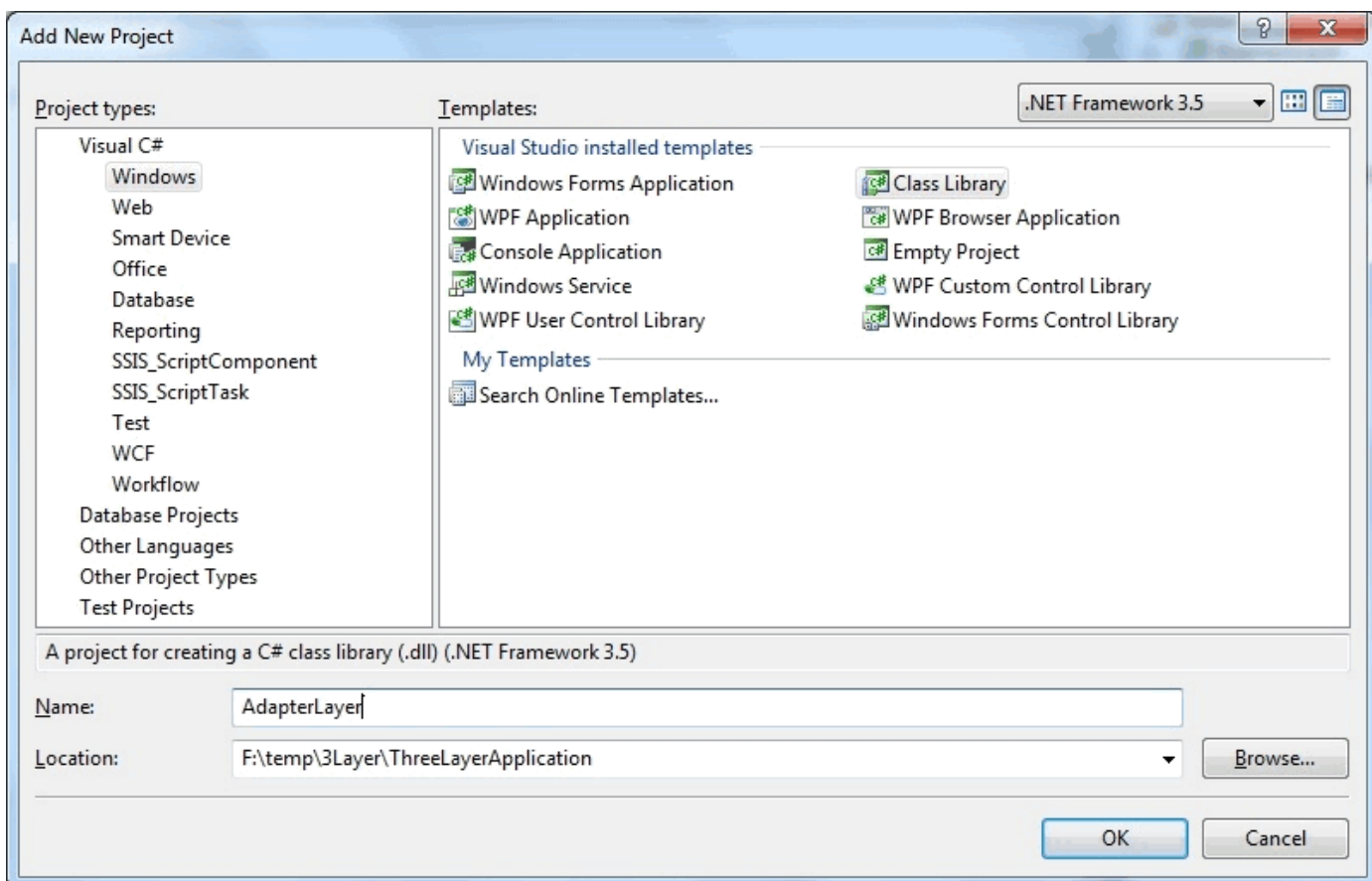
    [ForeignKey("BornInCityId")]
    public virtual City BornInCity { get; set; }
    public int BornInCityId { get; set; }
}

public class City
{
    public int Id { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Person> People { get; set; }
}
```

در ادامه این کلاس‌ها را در معرض دید EF Code first قرار داده:

```
public class MyContext : DbContext
{
    public DbSet<City> Cities { get; set; }
    public DbSet<Person> People { get; set; }
}
```



و همچنین تعدادی رکورد آغازین را نیز به جداول مرتبط اضافه می‌کنیم:

```
public class Configuration : DbMigrationsConfiguration<MyContext>
{
    public Configuration()
    {
        AutomaticMigrationsEnabled = true;
        AutomaticMigrationDataLossAllowed = true;
    }

    protected override void Seed(MyContext context)
    {
        var city1 = new City { Name = "city-1" };
        var city2 = new City { Name = "city-2" };
        context.Cities.Add(city1);
        context.Cities.Add(city2);

        var person1 = new Person { Name = "user-1", BornInCity = city1 };
        var person2 = new Person { Name = "user-2", BornInCity = city1 };
        context.People.Add(person1);
        context.People.Add(person2);

        base.Seed(context);
    }
}
```

در این حالت برای نمایش لیست نام افراد به همراه محل تولد آن‌ها، بنابر روال سابق SQL نویسی، نوشتن کوئری LINQ زیر بسیار متداول است:

```
public static class Test
{
    public static void RunTests()
    {
        Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());
    }
}
```

```
using (var context = new MyContext())
{
    var peopleAndCitiesList = from person in context.People
                              join city in context.Cities
                              on person.BornInCityId equals city.Id
                              select new
                              {
                                  PersonName = person.Name,
                                  CityName = city.Name
                              };

    foreach (var item in peopleAndCitiesList)
    {
        Console.WriteLine("{0}:{1}", item.PersonName, item.CityName);
    }
}
}
```

که حاصل آن اجرای کوئری ذیل بر روی بانک اطلاعاتی خواهد بود:

```
SELECT
    [Extent1].[BornInCityId] AS [BornInCityId],
    [Extent1].[Name] AS [Name],
    [Extent2].[Name] AS [Name1]
FROM    [dbo].[People] AS [Extent1]
INNER JOIN [dbo].[Cities] AS [Extent2] ON [Extent1].[BornInCityId] = [Extent2].[Id]
```

این نوع کوئری‌های join دار را به نحو ساده‌تری نیز می‌توان در EF با استفاده از خواص راهبری و بدون join نویسی مستقیم تهیه کرد:

```
var peopleAndCitiesList = context.People
                              .Select(person => new
                              {
                                  PersonName = person.Name,
                                  CityName = person.BornInCity.Name
                              });
```

که دقیقاً همان خروجی SQL یاد شده را تولید می‌کند.

## مثال دوم:

می‌خواهیم لیست شهرها را بر اساس تعداد کاربر متناظر به صورت نزولی مرتب کنیم:

```
var citiesList = context.Cities.OrderByDescending(x => x.People.Count());
foreach (var item in citiesList)
{
    Console.WriteLine("{0}", item.Name);
}
```

همانطور که مشاهده می‌کنید از خواص راهبری در قسمت order by هم می‌شود استفاده کرد. خروجی SQL کوئری فوق به صورت زیر است:

```
SELECT
    [Project1].[Id] AS [Id],
    [Project1].[Name] AS [Name]
FROM ( SELECT
        [Extent1].[Id] AS [Id],
        [Extent1].[Name] AS [Name],
        (SELECT
            COUNT(1) AS [A1]
            FROM [dbo].[People] AS [Extent2]
            WHERE [Extent1].[Id] = [Extent2].[BornInCityId]) AS [C1]
        FROM [dbo].[Cities] AS [Extent1]
    ) AS [Project1]
ORDER BY [Project1].[C1] DESC
```

### مثال سوم:

در ادامه قصد داریم لیست شهرها را به همراه تعداد نفرات متناظر با آنها نمایش دهیم:

```
var peopleAndCitiesList = context.Cities
    .Select(city => new
    {
        InUseCount = city.People.Count(),
        CityName = city.Name
    });

foreach (var item in peopleAndCitiesList)
{
    Console.WriteLine("{0}:{1}", item.CityName, item.InUseCount);
}
```

در اینجا از خاصیت راهبری People برای شمارش تعداد اعضای متناظر با هر شهر استفاده شده است.  
خروجی SQL کوئری فوق به نحو ذیل است:

```
SELECT
[Extent1].[Id] AS [Id],
(SELECT
    COUNT(1) AS [A1]
    FROM [dbo].[People] AS [Extent2]
    WHERE [Extent1].[Id] = [Extent2].[BornInCityId]) AS [C1],
[Extent1].[Name] AS [Name]
FROM [dbo].[Cities] AS [Extent1]
```

## نظرات خوانندگان

نویسنده: ایلیا

تاریخ: ۱۳۹۱/۰۷/۰۸ ۲۰:۳۲

مختصر و مفید. عالی. سپاس.

نویسنده: میرزایی

تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۲:۵۸

با سلام

به موضوع جالب و کاربردی ای اشاره فرمودید.

لطفا روش کار در هنگامی که ارتباط دو جدول به صورت یک به چند باشد و قصد بازیابی رکوردهایی را از جدول اول در حالتی که حداقل یک رکورد در جدول دوم با شرط ما وجود داشته باشد را بیان فرمایید.

مثلا جدول کارمندان یک شرکت با شرکت هایی که هر فرد قبلا در آن سابقه کار داشته است. می‌خواهیم کارمندانی را که در شرکت x کار کرده اند را به دست آوریم.

با تشکر از مطالب مفید شما.

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۳:۰۹

معادل سؤال شما با توجه به مدل‌های فوق به صورت زیر است:  
می‌خواهیم لیست افرادی را بدست بیاوریم که در شهر x متولد شده‌اند.  
روش اول: اگر شماره شهر را داریم:

```
var cityId = 1;  
var list = context.People.Where(x => x.BornInCityId == cityId).ToList();
```

روش دوم: اگر نام شهر را داریم:

```
var cityName = "city-1";  
var list2 = context.People.Where(x => x.BornInCity.Name == cityName).ToList();
```

در روش اول از [نکته تعریف کلید خارجی](#) استفاده شده.  
در روش دوم از نکته استفاده از خواص راهبری، استفاده شده.

نویسنده: سید مهران موسوی

تاریخ: ۱۳۹۱/۰۹/۱۶ ۱:۵۷

ممنون از مطلب مفیدتون . جالب اینه که بدون هیچ دردسری از خواص راهبری میشه برای به روز رسانی و افزودن رکوردهای مرتبط در صورت وجود رابطه‌های صحیح و نرمال سازی دقیق پایگاه داده بهترین استفاده رو کرد ... واقعا ORM ها برنامه نویسانو از شر کد نویسی تکراری و خسته کننده‌ی بانک اطلاعاتی تا حد زیادی راحت کردن ...

نویسنده: debugger

تاریخ: ۱۳۹۲/۰۴/۲۳ ۰:۲۹

با سلام؛ اگه حالتی که برای کاربر میرزایی پاسخ دادید برعکس بشه کوئری به چه صورت میشه ، یعنی اگر بخوایم فهرست شهرهایی که در اون فردی به اسم خاصی متولد شده رو بدست بیاریم (خروجی کوئری از جنس لیستی از شهر باشه ) کوئری رو به صورت زیر نوشتم اگه راهنمایی کنید در صورتی که بخوایم از طریق cities به خروجی مورد نظر برسیم ممنون میشم .

```
string personName = "user-1";
var result = context.People.Where(p => p.Name == personName).Select(c =>
c.BornInCity).ToList();
```

ممنون

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۲/۰۴/۲۳ ۰:۵۷

از Any استفاده کنید:

```
var citiesContainPerson = context.Cities.Where(city => city.People.Any(person => person.Name == "user-1")).ToList();
```

با این خروجی SQL:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name]
FROM [dbo].[Cities] AS [Extent1]
WHERE EXISTS (SELECT
1 AS [C1]
FROM [dbo].[People] AS [Extent2]
WHERE ([Extent1].[Id] = [Extent2].[BornInCityId]) AND (N'user-1' = [Extent2].[Name]))
)
```

نویسنده: میثم  
تاریخ: ۱۳۹۲/۰۹/۱۱ ۱۹:۵۴

سلام واقعا ممنون بابت این مطالب کلی مسائل جدید یاد گرفتم از سایتتون.  
یه سوال داشتم اگر براتون مقدوره راهنمایی کنید ممنون : تو این خاصیت راهبری کوئری ایجاد شده به صورت inner join درمیا حالا ما اگه بخوایم left - right outer join یا حتی full join بشه کوئریمون به چه صورت باید عمل کنیم؟ اصلا با خاصیت راهبری EF میشه همچین کاری رو انجام داد؟

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۲/۰۹/۱۱ ۲۱:۰۶

اینجا بحث شده: « [شبیه سازی outer Join در entity framework](#) »

نویسنده: saeed  
تاریخ: ۱۳۹۲/۱۰/۰۹ ۱۷:۵۲

سلام؛ میشه منظورتون رو در مورد خواص راهبری بگید ؟ یعنی به چی میگن خواص راهبری ؟

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۲/۱۰/۰۹ ۱۷:۵۹

خاصیتی که یک entity را به entity دیگر وصل می‌کند: [navigation property](#)  
در مثال بالا خاصیت‌هایی که به صورت virtual تعریف شدند، از این دست هستند.

نویسنده: سعید  
تاریخ: ۱۴:۴ ۱۳۹۲/۱۱/۲۰

سلام؛ در یک رابطه many-to-many چطور میشه اطلاعات رو واکنشی کرد.

نویسنده: وحید نصیری  
تاریخ: ۱۴:۸ ۱۳۹۲/۱۱/۲۰

« بررسی تفصیلی رابطه Many-to-Many در EF Code first »

نویسنده: فخاری  
تاریخ: ۱۸:۴۸ ۱۳۹۳/۰۴/۰۵

با سلام  
اگه یک کلاس مخاطب با کد زیر باشه:

```
public class Contact
{
    public int ContactId { get; set; }
    public string FName { get; set; }
    public string LName { get; set; }
    public string FatherName { get; set; }
    public string Email { get; set; }
    public virtual ICollection<Phone> Phones { get; set; }
}
```

و یک کلاس هم برای شماره تلفن‌ها با کد زیر:

```
public class Phone
{
    public int PhoneId { get; set; }
    public string PhoneNumber { get; set; }
    public string PhoneNote { get; set; }
    public string PhoneAddress { get; set; }
    public int PhoneTypeId { get; set; }
    public virtual PhoneType PhoneType { get; set; }

    [ForeignKey("ContactId")]
    public virtual Contact Contact { get; set; }
    public int ContactId { get; set; }
}
```

حالا در زمان جستجو من از کد زیر استفاده نموده ام :

```
var listContacts = db.Contacts.Include(p => p.Phones).AsQueryable();
if (searchContact.ByNumber)
    listContacts = listContacts.Where(c => c.LName.Contains(searchContact.Name));
if (searchContact.ByName)
{
    listContacts = listContacts.Where(c=>c.);
}
var phonelistmodel = await
    listContacts.OrderBy(p => p.ContactId)
        .Skip(page * count)
        .Take(count)
        .Select(c => new ListPhoneNumberViewmodel()
        {
            ContactId = c.ContactId,
            Email = c.Email,
            Name = c.FName + " " + c.LName,
            Phones = c.Phones
        }).ToListAsync();
```

ولی اصلا به اطلاعات جدول phone دسترسی ندارم؟

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۳/۰۴/۰۵ ۲۰:۴

- از Any استفاده کنید، برای رسیدن به لیست اشخاص:

```
listContacts = listContacts.Where(c => c.Phones.Any(x => x.PhoneNumber == "1234....."));
```

- قبل از where یک SelectMany قرار دهید، برای رسیدن به لیست تلفن‌ها:

```
listContacts.SelectMany(c=>c.Phones).Where(c=>c.PhoneNumber=="123....")
```