عنوان: **آشنایی با NHibernate - قسمت سوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۱۹ ۱۱:۲۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

در ادامه، تعاریف سایر موجودیت‌های سیستم ثبت سفارشات و نگاشت آن‌ها را بررسی خواهیم کرد.

کلاس Product تعریف شده در فایل جدید Product.cs در پوشه domain برنامه:

```
namespace NHSample1.Domain
{
   public class Product
   {
       public int Id { get; set; }
       public string Name { get; set; }
       public decimal UnitPrice { get; set; }
       public bool Discontinued { get; set; }
   }
}
```

کلاس ProductMapping تعریف شده در فایل جدید ProductMapping.cs (توصیه شده است که به ازای هر کلاس یک فایل جداگانه در نظر گرفته شود)، در پوشه Mappings برنامه:

```
using FluentNHibernate.Mapping;
using NHSample1.Domain;

namespace NHSample1.Mappings
{
   public class ProductMapping : ClassMap<Product>
   {
       public ProductMapping()
       {
           Not.LazyLoad();
           Id(p => p.Id).GeneratedBy.HiLo("1000");
           Map(p => p.Name).Length(50).Not.Nullable();
           Map(p => p.UnitPrice).Not.Nullable();
           Map(p => p.Discontinued).Not.Nullable();
       }
   }
}
```

همانطور که ملاحظه می‌کنید، روش تعریف آن‌ها همانند شیء Customer است که در قسمت‌های قبل بررسی شد و نکته جدیدی ندارد.

آزمون واحد بررسی این نگاشت نیز همانند مثال قبلی است.

کلاس ProductMapping_Fixture را در فایل جدید ProductMapping_Fixture.cs به پروژه UnitTests خود (که ارجاعات آن‌را در قسمت قبل مشخص کردیم) خواهیم افزود:

```
using NUnit.Framework;
using FluentNHibernate.Testing;
using NHSample1.Domain;

namespace UnitTests
{
   [TestFixture]
   public class ProductMapping_Fixture : FixtureBase
   {
       [Test]
       public void can_correctly_map_product()
       {
           new PersistenceSpecification<Product>(Session)
                   .CheckProperty(p => p.Id, 1001)
                   .CheckProperty(p => p.Name, "Apples")
                   .CheckProperty(p => p.UnitPrice, 10.45m)
```

```
                    .CheckProperty(p => p.Discontinued, true)
                    .VerifyTheMappings();
        }
    }
}
```

و پس از اجرای این آزمون واحد، عبارات SQL ایی که به صورت خودکار توسط این ORM جهت بررسی عملیات نگاشت صورت خواهند گرفت به صورت زیر می‌باشند:

```
ProductMapping_Fixture.can_correctly_map_product : Passed
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 2, @p1 = 1
NHibernate: INSERT INTO "Product" (Name, UnitPrice, Discontinued, Id) VALUES (@p0, @p1, @p2, @p3);@p0 =
'Apples', @p1 = 10.45, @p2 = True, @p3 = 1001
NHibernate: SELECT product0_.Id as Id1_0_, product0_.Name as Name1_0_, product0_.UnitPrice as
UnitPrice1_0_, product0_.Discontinued as Disconti4_1_0_ FROM "Product" product0_ WHERE
product0_.Id=@p0;@p0 = 1001
```

در ادامه تعریف کلاس کارمند، نگاشت و آزمون واحد آن به صورت زیر خواهند بود:

```
using System;
namespace NHSample1.Domain
{
    public class Employee
    {
        public int Id { set; get; }
        public string LastName { get; set; }
        public string FirstName { get; set; }
    }
}
```

```
using NHSample1.Domain;
using FluentNHibernate.Mapping;

namespace NHSample1.Mappings
{
    public class EmployeeMapping : ClassMap<Employee>
    {
        public EmployeeMapping()
        {
            Not.LazyLoad();
            Id(e => e.Id).GeneratedBy.Assigned();
            Map(e => e.LastName).Length(50);
            Map(e => e.FirstName).Length(50);
        }
    }
}
```

```
using NUnit.Framework;
using NHSample1.Domain;
using FluentNHibernate.Testing;

namespace UnitTests
{
    [TestFixture]
    public class EmployeeMapping_Fixture : FixtureBase
    {
        [Test]
        public void can_correctly_map_employee()
        {
            new PersistenceSpecification<Employee>(Session)
                    .CheckProperty(p => p.Id, 1001)
                    .CheckProperty(p => p.FirstName, "name1")
                    .CheckProperty(p => p.LastName, "lname1")
                    .VerifyTheMappings();
        }
    }
```

```
}
```

خروجی SQL حاصل از موفقیت آزمون واحد آن:

```
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 2, @p1 = 1
NHibernate: INSERT INTO "Employee" (LastName, FirstName, Id) VALUES (@p0, @p1, @p2);@p0 = 'lname1', @p1
= 'name1', @p2 = 1001
NHibernate: SELECT employee0_.Id as Id4_0_, employee0_.LastName as LastName4_0_, employee0_.FirstName
as FirstName4_0_ FROM "Employee" employee0_ WHERE employee0_.Id=@p0;@p0 = 1001
```

همانطور که ملاحظه می‌کنید، این آزمون‌های واحد 4 مرحله را در یک سطر انجام می‌دهند:
الف) ایجاد یک وهله از کلاس Employee
ب) ثبت اطلاعات کارمند در دیتابیس
ج) دریافت اطلاعات کارمند در وهله‌ای جدید از شیء Employee
د) و در پایان بررسی می‌کند که آیا شیء جدید ایجاد شده با شیء اولیه مطابقت دارد یا خیر

اکنون در ادامه پیاده سازی سیستم ثبت سفارشات، به قسمت جالب این مدل می‌رسیم. قسمتی که در آن ارتباطات اشیاء و روابط one-to-many تعریف خواهند شد. تعاریف کلاس‌های OrderItem و OrderItemMapping را به صورت زیر در نظر بگیرید:

کلاس OrderItem تعریف شده در فایل جدید OrderItem.cs واقع شده در پوشه domain پروژه:
که در آن هر سفارش (order) دقیقا از یک محصول (product) تشکیل می‌شود و هر محصول می‌تواند در سفارشات متعدد و مختلفی درخواست شود.

```
namespace NHSample1.Domain
{
   public class OrderItem
   {
       public int Id { get; set; }
       public int Quantity { get; set; }
       public Product Product { get; set; }
   }
}
```

کلاس OrderItemMapping تعریف شده در فایل جدید OrderItemMapping.cs :

```
using FluentNHibernate.Mapping;
using NHSample1.Domain;

namespace NHSample1.Mappings
{
   public class OrderItemMapping : ClassMap<OrderItem>
   {
       public OrderItemMapping()
       {
           Not.LazyLoad();
           Id(oi => oi.Id).GeneratedBy.Assigned();
           Map(oi => oi.Quantity).Not.Nullable();
           References(oi => oi.Product).Not.Nullable();
       }
   }
}
```

نکته جدیدی که در این کلاس نگاشت مطرح شده است، واژه کلیدی References می‌باشد که جهت بیان این ارجاعات و وابستگی‌ها بکار می‌رود. این ارجاع بیانگر یک رابطه many-to-one بین سفارشات و محصولات است. همچنین در ادامه آن Not.Nullable ذکر شده است تا این ارجاع را اجباری نمائید (در غیر اینصورت سفارش غیر معتبر خواهد بود).
نکته‌ی دیگر مهم آن این مورد است که Id در اینجا به صورت یک کلید تعریف نشده است. یک آیتم سفارش داده شده، موجودیت

به حساب نیامده و فقط یک شیء مقداری (value object) است و به خودی خود امکان وجود ندارد. هر وهله از آن تنها توسط یک سفارش قابل تعریف است. بنابراین id در اینجا فقط به عنوان یک index می‌تواند مورد استفاده قرار گیرد و فقط توسط شیء Order زمانیکه یک OrderItem به آن اضافه می‌شود، مقدار دهی خواهد شد.

اگر برای این نگاشت نیز آزمون واحد تهیه کنیم، به صورت زیر خواهد بود:

```
using NUnit.Framework;
using NHSample1.Domain;
using FluentNHibernate.Testing;

namespace UnitTests
{
   [TestFixture]
   public class OrderItemMapping_Fixture : FixtureBase
   {
       [Test]
       public void can_correctly_map_order_item()
       {
           var product = new Product
           {
               Name = "Apples",
               UnitPrice = 4.5m,
               Discontinued = true
           };

           new PersistenceSpecification<OrderItem>(Session)
                       .CheckProperty(p => p.Id, 1)
                       .CheckProperty(p => p.Quantity, 5)
                       .CheckReference(p => p.Product, product)
                       .VerifyTheMappings();
       }
   }
}
```

مشکل! این آزمون واحد با شکست مواجه خواهد شد، زیرا هنوز مشخص نکرده‌ایم که دو شیء Product را که در قسمت CheckReference فوق برای این منظور معرفی کرده‌ایم، چگونه باید با هم مقایسه کرد. در مورد مقایسه نوع‌های اولیه و اصلی مانند int و string و امثال آن مشکلی نیست، اما باید منطق مقایسه سایر اشیاء سفارشی خود را با پیاده سازی اینترفیس IEqualityComparer دقیقا مشخص سازیم:

```
using System.Collections;
using NHSample1.Domain;

namespace UnitTests
{
   public class CustomEqualityComparer : IEqualityComparer
   {
       public bool Equals(object x, object y)
       {
           if (ReferenceEquals(x, y)) return true;
           if (x == null || y == null) return false;

           if (x is Product && y is Product)
               return (x as Product).Id == (y as Product).Id;

           if (x is Customer && y is Customer)
               return (x as Customer).Id == (y as Customer).Id;

           if (x is Employee && y is Employee)
               return (x as Employee).Id == (y as Employee).Id;

           if (x is OrderItem && y is OrderItem)
               return (x as OrderItem).Id == (y as OrderItem).Id;


           return x.Equals(y);
       }

       public int GetHashCode(object obj)
       {
           //شاید وقتی دیگر
```

```
            return obj.GetHashCode();
        }
    }
}
```

در اینجا فقط Id این اشیاء با هم مقایسه شده است. در صورت نیاز تمامی خاصیت‌های این اشیاء را نیز می‌توان با هم مقایسه کرد (یک سری از اشیاء بکار گرفته شده در این کلاس در ادامه بحث معرفی خواهند شد).
سپس برای بکار گیری این کلاس جدید، سطر مربوط به استفاده از PersistenceSpecification به صورت زیر تغییر خواهد کرد:

```
new PersistenceSpecification<OrderItem>(Session, new CustomEqualityComparer())
```

پس از این تغییرات و مشخص سازی نحوه‌ی مقایسه دو شیء سفارشی، آزمون واحد ما پاس شده و خروجی SQL تولید شده آن به صورت زیر می‌باشد:

```
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 2, @p1 = 1
NHibernate: INSERT INTO "Product" (Name, UnitPrice, Discontinued, Id) VALUES (@p0, @p1, @p2, @p3);@p0 =
'Apples', @p1 = 4.5, @p2 = True, @p3 = 1001
NHibernate: INSERT INTO "OrderItem" (Quantity, Product_id, Id) VALUES (@p0, @p1, @p2);@p0 = 5, @p1 =
1001, @p2 = 1
NHibernate: SELECT orderitem0_.Id as Id0_1_, orderitem0_.Quantity as Quantity0_1_,
orderitem0_.Product_id as Product3_0_1_, product1_.Id as Id3_0_, product1_.Name as Name3_0_,
product1_.UnitPrice as UnitPrice3_0_, product1_.Discontinued as Disconti4_3_0_ FROM "OrderItem"
orderitem0_ inner join "Product" product1_ on orderitem0_.Product_id=product1_.Id WHERE
orderitem0_.Id=@p0;@p0 = 1
```

قسمت پایانی کار تعاریف کلاس‌های نگاشت، مربوط به کلاس Order است که در ادامه بررسی خواهد شد.

```
using System;
using System.Collections.Generic;

namespace NHSample1.Domain
{
    public class Order
    {
        public int Id { set; get; }
        public DateTime OrderDate { get; set; }
        public Employee Employee { get; set; }
        public Customer Customer { get; set; }
        public IList<OrderItem> OrderItems { get; set; }
    }
}
```

نکته‌ی مهمی که در این کلاس وجود دارد استفاده از IList جهت معرفی مجموعه‌ای از آیتم‌های سفارشی است (بجای List و یا IEnumerable که در صورت استفاده خطای type cast exception در حین نگاشت حاصل می‌شد).

```
using NHSample1.Domain;
using FluentNHibernate.Mapping;

namespace NHSample1.Mappings
{
    public class OrderMapping : ClassMap<Order>
    {
        public OrderMapping()
        {
            Not.LazyLoad();
            Id(o => o.Id).GeneratedBy.GuidComb();
            Map(o => o.OrderDate).Not.Nullable();
            References(o => o.Employee).Not.Nullable();
            References(o => o.Customer).Not.Nullable();
            HasMany(o => o.OrderItems)
                .AsList(index => index.Column("ListIndex").Type<int>());
        }
```

```
    }
}
```

در تعاریف نگاشت این کلاس نیز دو ارجاع به اشیاء کارمند و مشتری وجود دارد که با References مشخص شده‌اند.
قسمت جدید آن HasMany است که جهت تعریف رابطه one-to-many بکار گرفته شده است. یک سفارش رابطه many-to-one با یک مشتری و همچنین کارمندی که این رکورد را ثبت می‌کند، دارد. در اینجا مجموعه آیتم‌های یک سفارش به صورت یک لیست بازگشت داده می‌شود و ایندکس آن به ستونی به نام ListIndex در یک جدول دیتابیس نگاشت خواهد شد. نوع این ستون، int می‌باشد.

```
using System;
using System.Collections.Generic;
using NUnit.Framework;
using NHSample1.Domain;
using FluentNHibernate.Testing;

namespace UnitTests
{
   [TestFixture]
   public class OrderMapping_Fixture : FixtureBase
   {
       [Test]
       public void can_correctly_map_an_order()
       {
           {
               var product1 =
                   new Product
                   {
                       Name = "Apples",
                       UnitPrice = 4.5m,
                       Discontinued = true
                   };
               var product2 =
                   new Product
                   {
                       Name = "Pears",
                       UnitPrice = 3.5m,
                       Discontinued = false
                   };

               Session.Save(product1);
               Session.Save(product2);

               var items = new List<OrderItem>
                               {
                                   new OrderItem
                                   {
                                       Id = 1,
                                       Quantity = 100,
                                       Product = product1
                                   },
                                   new OrderItem
                                   {
                                       Id = 2,
                                       Quantity = 200,
                                       Product = product2
                                   }
                               };

               var customer = new Customer
               {
                   FirstName = "Vahid",
                   LastName = "Nasiri",
                   AddressLine1 = "Addr1",
                   AddressLine2 = "Addr2",
                   PostalCode = "1234",
                   City = "Tehran",
                   CountryCode = "IR"
               };

               var employee =
                   new Employee
                   {
                       FirstName = "name1",
                       LastName = "lname1"
                   };
```

```
                var order = new Order
                {
                    Customer = customer,
                    Employee = employee,
                    OrderDate = DateTime.Today,
                    OrderItems = items
                };

                new PersistenceSpecification<Order>(Session, new CustomEqualityComparer())
                    .CheckProperty(o => o.OrderDate, order.OrderDate)
                    .CheckReference(o => o.Customer, order.Customer)
                    .CheckReference(o => o.Employee, order.Employee)
                    .CheckList(o => o.OrderItems, order.OrderItems)
                    .VerifyTheMappings();
            }
        }
    }
}
```

همانطور که ملاحظه می‌کنید در این متد آزمون واحد، نیاز به مشخص سازی منطق مقایسه اشیاء سفارش، مشتری و آیتم‌های سفارش داده شده نیز وجود دارد که پیشتر در کلاس CustomEqualityComparer معرفی شدند؛ درغیر اینصورت این آزمون واحد با شکست مواجه می‌شد.
متد آزمون واحد فوق کمی طولانی است؛ زیرا در آن باید تعاریف انواع و اقسام اشیاء مورد استفاده را مشخص نمود (و ارزش کار نیز دقیقا در همینجا مشخص می‌شود که بجای SQL نوشتن، با اشیایی که توسط کامپایلر تحت نظر هستند سر و کار داریم).
تنها نکته جدید آن استفاده از CheckList برای بررسی IList تعریف شده در قسمت قبل است.

خروجی SQL این آزمون واحد پس از اجرا و موفقیت آن به صورت زیر است:

```
OrderMapping_Fixture.can_correctly_map_an_order : Passed
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 2, @p1 = 1
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 3, @p1 = 2
NHibernate: INSERT INTO "Product" (Name, UnitPrice, Discontinued, Id) VALUES (@p0, @p1, @p2, @p3);@p0 =
'Apples', @p1 = 4.5, @p2 = True, @p3 = 1001
NHibernate: INSERT INTO "Product" (Name, UnitPrice, Discontinued, Id) VALUES (@p0, @p1, @p2, @p3);@p0 =
'Pears', @p1 = 3.5, @p2 = False, @p3 = 1002
NHibernate: INSERT INTO "Customer" (FirstName, LastName, AddressLine1, AddressLine2, PostalCode, City,
CountryCode, Id) VALUES (@p0, @p1, @p2, @p3, @p4, @p5, @p6, @p7);@p0 = 'Vahid', @p1 = 'Nasiri', @p2 =
'Addr1', @p3 = 'Addr2', @p4 = '1234', @p5 = 'Tehran', @p6 = 'IR', @p7 = 2002
NHibernate: select next_hi from hibernate_unique_key
NHibernate: update hibernate_unique_key set next_hi = @p0 where next_hi = @p1;@p0 = 4, @p1 = 3
NHibernate: INSERT INTO "Employee" (LastName, FirstName, Id) VALUES (@p0, @p1, @p2);@p0 = 'lname1', @p1
= 'name1', @p2 = 3003
NHibernate: INSERT INTO "OrderItem" (Quantity, Product_id, Id) VALUES (@p0, @p1, @p2);@p0 = 100, @p1 =
1001, @p2 = 1
NHibernate: INSERT INTO "OrderItem" (Quantity, Product_id, Id) VALUES (@p0, @p1, @p2);@p0 = 200, @p1 =
1002, @p2 = 2
NHibernate: INSERT INTO "Order" (OrderDate, Employee_id, Customer_id, Id) VALUES (@p0, @p1, @p2,
@p3);@p0 = 2009/10/10 12:00:00 ق.ظ, @p1 = 3003, @p2 = 2002, @p3 = 0
NHibernate: UPDATE "OrderItem" SET Order_id = @p0, ListIndex = @p1 WHERE Id = @p2;@p0 = 0, @p1 = 0, @p2
= 1
NHibernate: UPDATE "OrderItem" SET Order_id = @p0, ListIndex = @p1 WHERE Id = @p2;@p0 = 0, @p1 = 1, @p2
= 2
NHibernate: SELECT order0_.Id as Id1_2_, order0_.OrderDate as OrderDate1_2_, order0_.Employee_id as
Employee3_1_2_, order0_.Customer_id as Customer4_1_2_, employee1_.Id as Id4_0_, employee1_.LastName as
LastName4_0_, employee1_.FirstName as FirstName4_0_, customer2_.Id as Id2_1_, customer2_.FirstName as
FirstName2_1_, customer2_.LastName as LastName2_1_, customer2_.AddressLine1 as AddressL4_2_1_,
customer2_.AddressLine2 as AddressL5_2_1_, customer2_.PostalCode as PostalCode2_1_, customer2_.City as
City2_1_, customer2_.CountryCode as CountryC8_2_1_ FROM "Order" order0_ inner join "Employee"
employee1_ on order0_.Employee_id=employee1_.Id inner join "Customer" customer2_ on
order0_.Customer_id=customer2_.Id WHERE order0_.Id=@p0;@p0 = 0
NHibernate: SELECT orderitems0_.Order_id as Order4_2_, orderitems0_.Id as Id2_, orderitems0_.ListIndex
as ListIndex2_, orderitems0_.Id as Id0_1_, orderitems0_.Quantity as Quantity0_1_,
orderitems0_.Product_id as Product3_0_1_, product1_.Id as Id3_0_, product1_.Name as Name3_0_,
product1_.UnitPrice as UnitPrice3_0_, product1_.Discontinued as Disconti4_3_0_ FROM "OrderItem"
orderitems0_ inner join "Product" product1_ on orderitems0_.Product_id=product1_.Id WHERE
orderitems0_.Order_id=@p0;@p0 = 0
```

تا اینجای کار تعاریف اشیاء ، نگاشت آن‌ها و همچنین بررسی صحت این نگاشت‌ها به پایان می‌رسد.

نکته:
دیتابیس برنامه را جهت آزمون‌های واحد برنامه، از نوع SQLite ساخته شده در حافظه مشخص کردیم. اگر علاقمند باشید که database schema تولید شده توسط NHibernate را مشاهده نمائید، در متد SetupContext کلاس FixtureBase که در قسمت قبل معرفی شد، سطر آخر را به صورت زیر تغییر دهید، تا اسکریپت دیتابیس نیز به صورت خودکار در خروجی اس کیوال آزمون واحد لحاظ شود (پارامتر دوم آن مشخص می‌کند که schema ساخته شده، نمایش داده شود یا خیر):

```
SessionSource.BuildSchema(Session, true);
```

پس از این تغییر و انجام مجدد آزمون واحد، اسکریپت دیتابیس ما به صورت زیر خواهد بود (که جهت ایجاد یک دیتابیس SQLite می‌تواند مورد استفاده قرار گیرد):

```
drop table if exists "OrderItem"

    drop table if exists "Order"

    drop table if exists "Customer"

    drop table if exists "Product"

    drop table if exists "Employee"

    drop table if exists hibernate_unique_key

    create table "OrderItem" (
        Id INTEGER not null,
       Quantity INTEGER not null,
       Product_id INTEGER not null,
       Order_id INTEGER,
       ListIndex INTEGER,
       primary key (Id)
    )

    create table "Order" (
        Id INTEGER not null,
       OrderDate DATETIME not null,
       Employee_id INTEGER not null,
       Customer_id INTEGER not null,
       primary key (Id)
    )

    create table "Customer" (
        Id INTEGER not null,
       FirstName TEXT not null,
       LastName TEXT not null,
       AddressLine1 TEXT not null,
       AddressLine2 TEXT,
       PostalCode TEXT not null,
       City TEXT not null,
       CountryCode TEXT not null,
       primary key (Id)
    )

    create table "Product" (
        Id INTEGER not null,
       Name TEXT not null,
       UnitPrice NUMERIC not null,
       Discontinued INTEGER not null,
       primary key (Id)
    )

    create table "Employee" (
        Id INTEGER not null,
       LastName TEXT,
       FirstName TEXT,
       primary key (Id)
    )
```

```
    create table hibernate_unique_key (
         next_hi INTEGER
    )
```

البته اگر مستندات SQLite را مطالعه کرده باشید می‌دانید که مفهوم کلید خارجی در این دیتابیس وجود دارد اما اعمال نمی‌شود! (برای اعمال آن باید تریگر نوشت) به همین جهت در این اسکریپت تولیدی خبری از کلید خارجی نیست.

برای اینکه از دیتابیس اس کیوال سرور استفاده کنیم، در همان متد SetupContext کلاس مذکور، سطر اول را به صورت زیر تغییر دهید (نوع دیتابیس اس کیوال سرور 2008 مشخص شده و سپس رشته اتصالی به دیتابیس ذکر گردیده است):

```
var cfg = Fluently.Configure().Database(
              // SQLiteConfiguration.Standard.ShowSql().InMemory
              MsSqlConfiguration
              .MsSql2008
              .ShowSql()
              .ConnectionString("Data Source=(local);Initial Catalog=testdb2009;Integrated Security =
true")
               );
```

اکنون اگر مجددا آزمون واحد را اجرا نمائیم، اسکریپت تولیدی به صورت زیر خواهد بود (در اینجا مفهوم استقلال برنامه از نوع دیتابیس را به خوبی می‌توان درک کرد):

```
if exists (select 1 from sys.objects where object_id = OBJECT_ID(N'[FK3EF88858466CFBF7]') AND
parent_object_id = OBJECT_ID('[OrderItem]'))
alter table [OrderItem]  drop constraint FK3EF88858466CFBF7


    if exists (select 1 from sys.objects where object_id = OBJECT_ID(N'[FK3EF888589F32DE52]') AND
parent_object_id = OBJECT_ID('[OrderItem]'))
alter table [OrderItem]  drop constraint FK3EF888589F32DE52


    if exists (select 1 from sys.objects where object_id = OBJECT_ID(N'[FK3117099B1EBA72BC]') AND
parent_object_id = OBJECT_ID('[Order]'))
alter table [Order]  drop constraint FK3117099B1EBA72BC


    if exists (select 1 from sys.objects where object_id = OBJECT_ID(N'[FK3117099BB2F9593A]') AND
parent_object_id = OBJECT_ID('[Order]'))
alter table [Order]  drop constraint FK3117099BB2F9593A


    if exists (select * from dbo.sysobjects where id = object_id(N'[OrderItem]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1) drop table [OrderItem]

    if exists (select * from dbo.sysobjects where id = object_id(N'[Order]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1) drop table [Order]

    if exists (select * from dbo.sysobjects where id = object_id(N'[Customer]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1) drop table [Customer]

    if exists (select * from dbo.sysobjects where id = object_id(N'[Product]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1) drop table [Product]

    if exists (select * from dbo.sysobjects where id = object_id(N'[Employee]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1) drop table [Employee]

    if exists (select * from dbo.sysobjects where id = object_id(N'hibernate_unique_key') and
OBJECTPROPERTY(id, N'IsUserTable') = 1) drop table hibernate_unique_key

    create table [OrderItem] (
        Id INT not null,
       Quantity INT not null,
       Product_id INT not null,
       Order_id INT null,
       ListIndex INT null,
       primary key (Id)
    )

    create table [Order] (
        Id INT not null,
       OrderDate DATETIME not null,
```

```
    Employee_id INT not null,
    Customer_id INT not null,
    primary key (Id)
)

create table [Customer] (
     Id INT not null,
    FirstName NVARCHAR(50) not null,
    LastName NVARCHAR(50) not null,
    AddressLine1 NVARCHAR(50) not null,
    AddressLine2 NVARCHAR(50) null,
    PostalCode NVARCHAR(10) not null,
    City NVARCHAR(50) not null,
    CountryCode NVARCHAR(2) not null,
    primary key (Id)
)

create table [Product] (
     Id INT not null,
    Name NVARCHAR(50) not null,
    UnitPrice DECIMAL(19,5) not null,
    Discontinued BIT not null,
    primary key (Id)
)

create table [Employee] (
     Id INT not null,
    LastName NVARCHAR(50) null,
    FirstName NVARCHAR(50) null,
    primary key (Id)
)

alter table [OrderItem]
     add constraint FK3EF88858466CFBF7
     foreign key (Product_id)
     references [Product]

alter table [OrderItem]
     add constraint FK3EF888589F32DE52
     foreign key (Order_id)
     references [Order]

alter table [Order]
     add constraint FK3117099B1EBA72BC
     foreign key (Employee_id)
     references [Employee]

alter table [Order]
     add constraint FK3117099BB2F9593A
     foreign key (Customer_id)
     references [Customer]

create table hibernate_unique_key (
      next_hi INT
)
```

که نکات ذیل در مورد آن جالب توجه است:

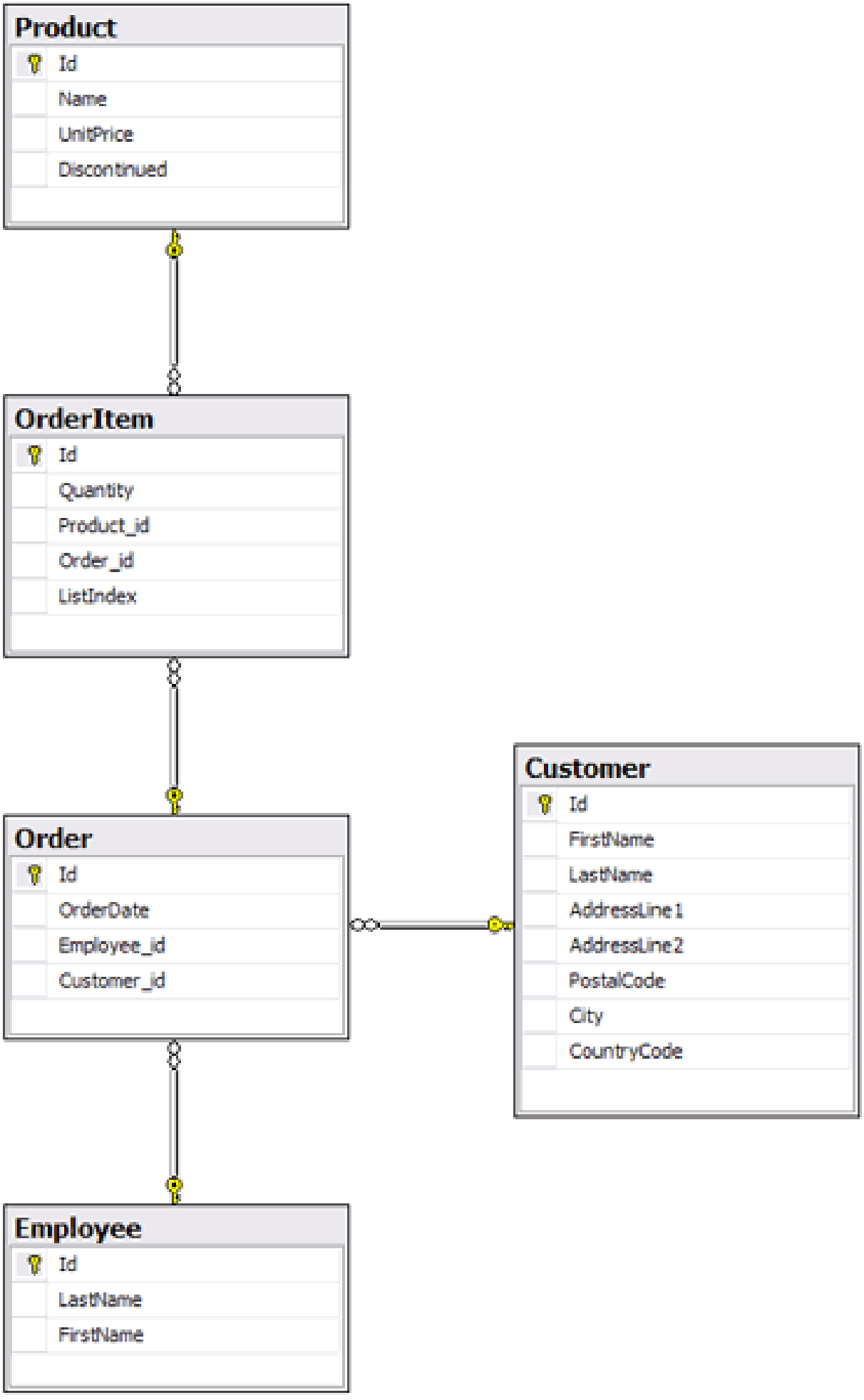
الف) جداول مطابق نام کلاس‌های ما تولید شده‌اند.

ب) نام فیلدها دقیقا مطابق نام خواص کلاس‌های ما تشکیل شده‌اند.

ج) Id ها به صورت primary key تعریف شده‌اند (از آنجائیکه ما در هنگام تعریف نگاشت‌ها، آن‌ها را از نوع identity مشخص کرده بودیم).

د) رشته‌ها به نوع nvarchar با اندازه 50 نگاشت شده‌اند.

ه) کلیدهای خارجی بر اساس نام جدول با پسوند _id تشکیل شده‌اند.


Product
Id
Name
UnitPrice
Discontinued
OrderItem
Id
Quantity
Product_id
Order_id
ListIndex
Order
Id
OrderDate
Employee_id
Customer_id
Customer
Id
FirstName
LastName
AddressLine1
AddressLine2
PostalCode
City
CountryCode
Employee
Id
LastName
FirstName

ادامه دارد ...

نظرات خوانندگان

نویسنده: DotNetCoders
تاریخ: ۱۳۸۸/۰۷/۱۹ ۱۲:۳۵:۳۱

عالیه جناب نصیری !

یه سوال بی ربط : توی جدول OrderItem فیلد ListIndex رو برای چی گذاشتید ؟

خسته نباشید...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۱۹ ۱۲:۴۷:۱۸

سلام،
در یک order_id مشخص، میشه اسمش رو گذاشت شماره ردیف سند.

نویسنده: Ali
تاریخ: ۱۳۸۹/۰۹/۱۷ ۲۱:۳۳:۵۰

با سلام
من در NHibernate 2 از این کلاس استفاده می کردم:(نگاشت، مربوط به جدول tblAhkam است)

```
public class Ahkam
}
{ ;public virtual int Id { get; set
{ ;public virtual int HDate { get; set

public virtual string SepratedDate
}
get
}
return Functions.SepratePersianDate(HDate);//Convert 890221 to 89/02/21
{
{
{
```

و در لایه BLL تبدیل به Dataset می کردم و استفاده می شد و مشکلی هم وجود نداشت. اما وقتی با NHibernate 3 برنامه رو اجرا کردم به مشکل برخورد و فهمیدم چون فیلد SepratedDate در جدول بانک وجود ندارد باعث خطا شده است. راه حلی وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۷ ۲۲:۴۰:۳۹

سؤال شما مرتبط است به موضوع "nhibernate derived properties" (+) و برای بررسی مشکل شما نیاز به این موارد است:

- چگونه نگاشت‌ها را تعریف کرده‌اید. (نیاز به سورس است)
- دقیقا چه خطایی می‌گیرید. متن آن خیلی مهم است.

لطفا از امکانات انجمن‌ها برای ادامه‌ی بحث استفاده کنید.

+ اگر از fluent NHibernate استفاده می‌کنید، نگارش سازگار با NHibernate 3 آن هنوز ارائه نشده (به زودی) : (+)

نویسنده: fateme
تاریخ: ۱۳۸۹/۱۰/۲۰ ۱۲:۱۲:۴۱

سلام آقای نصیری
خیلی ممنون از آموزش بسیار عالی تون
من با اینکه در قسمت unit tests کلاس CustomEqualityComparer ساختم ولی و این کلاسو در orderItemMapping_Fixture هم آوردم ولی باز error not-null property references a null or transient valueNHSample1.Domain.OrderItem.Product رو میده وقتی unit testرو اجرا می کنم
لطف می کنید راهنماییم کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۲۰ ۱۲:۲۶:۴۵

این خطا زمانی حاصل میشه که شیءایی که خودش یک یا چند خواصش شیء دیگر هستند (ارجاعات به جداول دیگر)، به درستی مقدار دهی نشده و حداقل یکی از این موارد نال است.

نویسنده: fateme
تاریخ: ۱۳۸۹/۱۰/۲۰ ۱۵:۲۴:۵۷

ببخشید من تونستم مشکلو حل کنم
دقیقا کدی که شما واسه آموزش گذاشتیدو وارد کردم
ولی وارد کلاس CustomEqualityComparer نمی شه

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۲۰ ۱۸:۱۰:۱۶

شاید جایی رو از قلم انداخته‌اید (solution کامل شما باید برای دیباگ موجود باشد).
سورس کامل قابل دریافت این موارد در پایان قسمت چهارم ارائه شده. به آن مراجعه کنید.