عنوان: **آشنایی با NHibernate - قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۱۸ ۱۳:۱۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

آزمون واحد کلاس نگاشت تهیه شده

در مورد آشنایی با آزمون‌های واحد لطفا به برچسب مربوطه در سمت راست سایت مراجعه بفرمائید. همچنین در مورد اینکه چرا به این نوع API کلمه Fluent اطلاق می‌شود، می‌توان به تعریف آن جهت مطالعه بیشتر مراجعه نمود.

در این قسمت قصد داریم برای بررسی وضعیت کلاس نگاشت تهیه شده یک آزمون واحد تهیه کنیم. برای این منظور ارجاعی را به اسمبلی nunit.framework.dll به پروژه UnitTests که در ابتدای کار به solution جاری در VS.Net افزوده بودیم، اضافه نمائید (همچنین ارجاع‌هایی به اسمبلی‌های پروژه NHSample1 ، FluentNHibernate ، System.Data.SQLite ، NHibernate.ByteCode.Castle و Nhibernate نیز نیاز هستند). تمام اسمبلی‌های این فریم ورک‌ها از پروژه FluentNHibernate قابل استخراج هستند.

سپس سه کلاس زیر را به پروژه آزمون واحد اضافه خواهیم کرد.
کلاس TestModel : (جهت مشخص سازی محل دریافت اطلاعات نگاشت)

```
using FluentNHibernate;
using NHSample1.Domain;

namespace UnitTests
{
   public class TestModel : PersistenceModel
   {
       public TestModel()
       {
          AddMappingsFromAssembly(typeof(CustomerMapping).Assembly);
       }
   }
}
```

کلاس FixtureBase : (جهت ایجاد سشن NHibernate در ابتدای آزمون واحد و سپس پاکسازی اشیاء در پایان کار)

```
using NUnit.Framework;
using NHibernate;
using FluentNHibernate;
using FluentNHibernate.Cfg;
using FluentNHibernate.Cfg.Db;

namespace UnitTests
{
   public class FixtureBase
   {
       protected SessionSource SessionSource { get; set; }
       protected ISession Session { get; private set; }

       [SetUp]
       public void SetupContext()
       {
           var cfg = Fluently.Configure().Database(SQLiteConfiguration.Standard.InMemory);

           SessionSource = new SessionSource(
                               cfg.BuildConfiguration().Properties,
                               new TestModel());

           Session = SessionSource.CreateSession();
           SessionSource.BuildSchema(Session);
       }
```

```
        [TearDown]
        public void TearDownContext()
        {
            Session.Close();
            Session.Dispose();
        }
    }
}
```

و کلاس CustomerMapping_Fixture.cs : (جهت بررسی صحت نگاشت تهیه شده با کمک دو کلاس قبل)

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

namespace UnitTests
{
   [TestFixture]
   public class CustomerMapping_Fixture : FixtureBase
   {
       [Test]
       public void can_correctly_map_customer()
       {
           new PersistenceSpecification<Customer>(Session)
                   .CheckProperty(c => c.Id, 1001)
                   .CheckProperty(c => c.FirstName, "Vahid")
                   .CheckProperty(c => c.LastName, "Nasiri")
                   .CheckProperty(c => c.AddressLine1, "Addr1")
                   .CheckProperty(c => c.AddressLine2, "Addr2")
                   .CheckProperty(c => c.PostalCode, "1234")
                   .CheckProperty(c => c.City, "Tehran")
                   .CheckProperty(c => c.CountryCode, "IR")
                   .VerifyTheMappings();
       }
   }
}
```

توضیحات:

اکنون به عنوان یک برنامه نویس متعهد نیاز است تا کار صورت گرفته در قسمت قبل را آزمایش کنیم.

کار بررسی صحت نگاشت تعریف شده در قسمت قبل توسط کلاس استاندارد PersistenceSpecification فریم ورک FluentNHibernate انجام خواهد شد (در کلاس CustomerMapping_Fixture). این کلاس برای انجام عملیات آزمون واحد نیاز به کلاس پایه دیگری به نام FixtureBase دارد که در آن کار ایجاد سشن NHibernate (در قسمت استاندارد SetUp آزمون واحد) و سپس آزاد سازی آن را در هنگام خاتمه کار ، انجام می‌دهد (در قسمت TearDown آزمون واحد).

این ویژگی‌ها که در مباحث آزمون واحد نیز به آن‌ها اشاره شده است، سبب اجرای متدهایی پیش از اجرا و بررسی هر آزمون واحد و سپس آزاد سازی خودکار منابع خواهند شد.

برای ایجاد یک سشن NHibernate نیاز است تا نوع دیتابیس و همچنین رشته اتصالی به آن (کانکشن استرینگ) مشخص شوند. فریم ورک Fluent NHibernate با ایجاد کلاس‌های کمکی برای این امر، به شدت سبب ساده سازی انجام آن شده است. در این مثال، نوع دیتابیس به SQLite و در حالت دیتابیس در حافظه (in memory)، تنظیم شده است (برای انجام امور آزمون واحد با سرعت بالا).

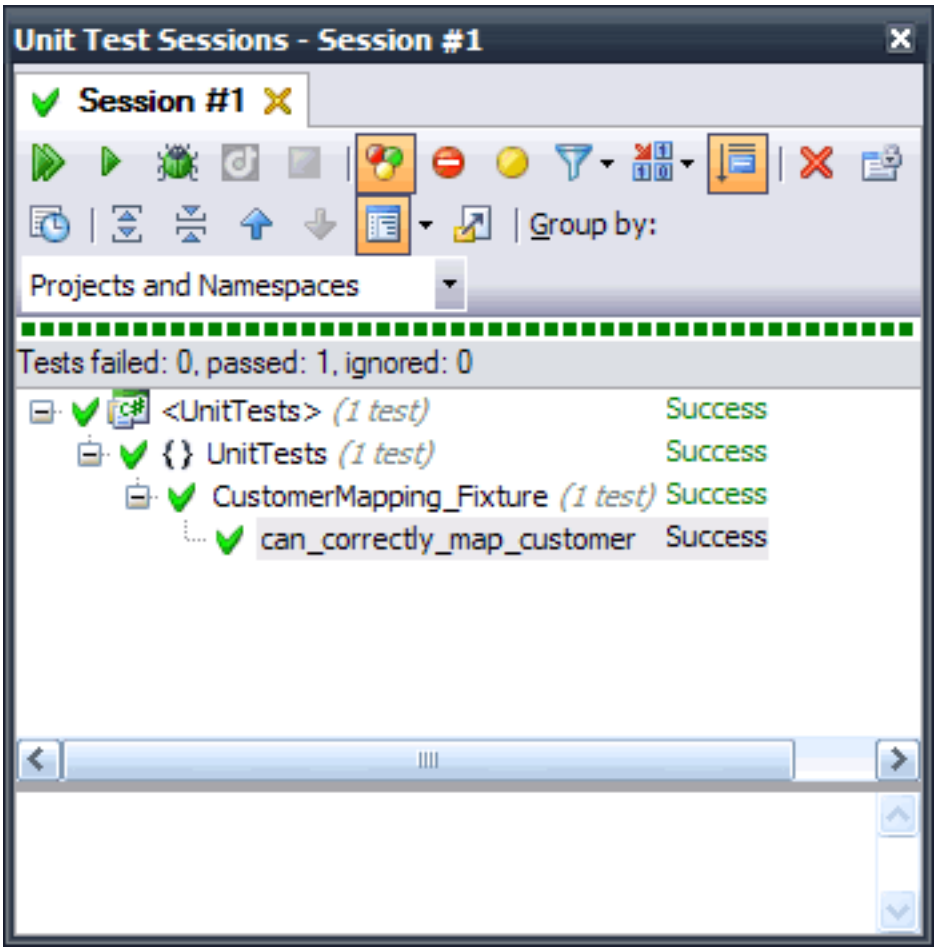
جهت اجرای هر دستوری در NHibernate نیاز به یک سشن می‌باشد. برای تعریف شیء سشن، نه تنها نیاز به مشخص سازی نوع و حالت دیتابیس مورد استفاده داریم، بلکه نیاز است تا وهله‌ای از کلاس استاندارد PersistanceModel را نیز جهت مشخص سازی کلاس نگاشت مورد استفاده مشخص نمائیم. برای این منظور کلاس TestModel فوق تعریف شده است تا این نگاشت را از اسمبلی مربوطه بخواند و مورد استفاده قرار دهد (بر پایی اولیه این مراحل شاید در ابتدای امر کمی زمانبر باشد اما در نهایت یک پروسه استاندارد است). توسط این کلاس به سیستم اعلام خواهیم کرد که اطلاعات نگاشت را باید از کدام کلاس دریافت کند.

تا اینجای کار شیء SessionSource را با معرفی نوع دیتابیس و همچنین محل دریافت اطلاعات نگاشت اشیاء معرفی کردیم. در دو سطر بعدی متد SetupContext کلاس FixtureBase ، ابتدا یک سشن را از این منبع سشن تهیه می‌کنیم. شیء منبع سشن در این فریم ورک در حقیقت یک factory object است (الگوهای طراحی برنامه نویسی شیءگرا) که امکان دسترسی به انواع و اقسام

دیتابیس‌ها را فراهم می‌سازد. برای مثال اگر روزی نیاز بود از دیتابیس اس کیوال سرور استفاده شود، می‌توان از کلاس MsSqlConfiguration بجای SQLiteConfiguration استفاده کرد و همینطور الی آخر.
در ادامه توسط شیء SessionSource کار ساخت database schema را نیز به صورت پویا انجام خواهیم داد. بله، همانطور که متوجه شده‌اید، کار ساخت database schema نیز به صورت پویا توسط فریم ورک NHibernate با توجه به اطلاعات کلاس‌های نگاشت، صورت خواهد گرفت.

این مراحل، نحوه ایجاد و بر پایی یک آزمایشگاه آزمون واحد فریم ورک Fluent NHibernate را مشخص ساخته و در پروژه‌های شما می‌توانند به کرات مورد استفاده قرار گیرند.

در ادامه اگر آزمون واحد را اجرا نمائیم (متد can_correctly_map_customer در کلاس CustomerMapping_Fixture)، نتیجه باید شبیه به شکل زیر باشد:

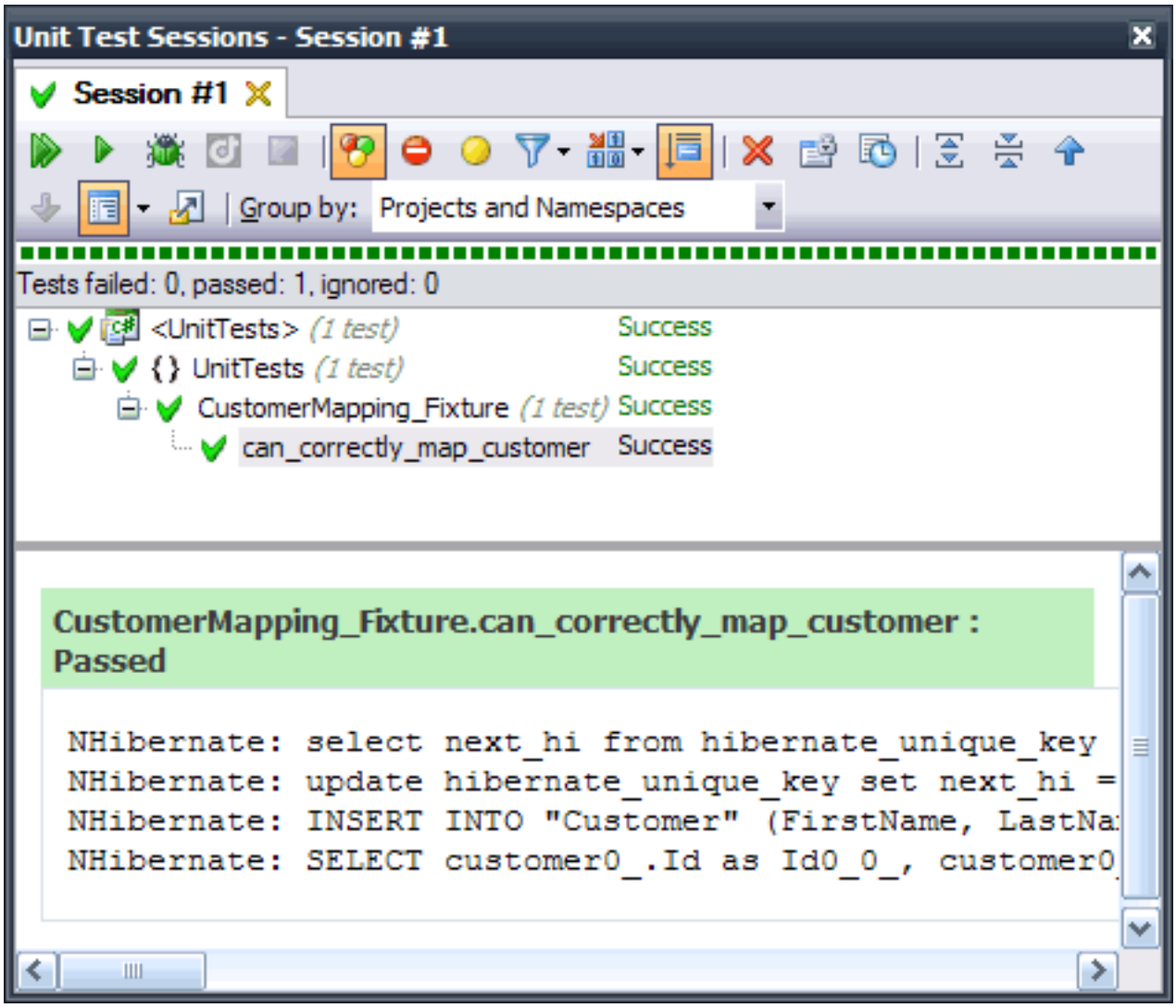


توسط متد CheckProperty کلاس PersistenceSpecification ، امکان بررسی نگاشت تهیه شده میسر است. اولین پارامتر آن، یک lambda expression خاصیت مورد نظر جهت بررسی است و دومین آرگومان آن، مقداری است که در حین آزمون به خاصیت تعریف شده انتساب داده می‌شود.

نکته:
شاید سؤال بپرسید که در تابع can_correctly_map_customer عملا چه اتفاقاتی رخ داده است؟ برای بررسی آن در متد SetupContext کلاس FixtureBase ، اولین سطر آن‌را به صورت زیر تغییر دهید تا عبارات SQL نهایی تولید شده را نیز بتوانیم در

حین عملیات تست مشاهده نمائیم:

```
var cfg = Fluently.Configure().Database(SQLiteConfiguration.Standard.ShowSql().InMemory);
```



مطابق متد تست فوق، عبارات تولید شده به شرح زیر هستند:

```
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 "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 = 1001
NHibernate: SELECT customer0_.Id as Id0_0_, customer0_.FirstName as FirstName0_0_, customer0_.LastName
as LastName0_0_, customer0_.AddressLine1 as AddressL4_0_0_, customer0_.AddressLine2 as AddressL5_0_0_,
customer0_.PostalCode as PostalCode0_0_, customer0_.City as City0_0_, customer0_.CountryCode as
CountryC8_0_0_ FROM "Customer" customer0_ WHERE customer0_.Id=@p0;@p0 = 1001
```

نکته جالب این عبارات، استفاده از کوئری‌های پارامتری است به صورت پیش فرض که در نهایت سبب بالا رفتن امنیت بیشتر برنامه (یکی از راه‌های جلوگیری از تزریق اس کیوال در ADO.Net که در نهایت توسط تمامی این فریم ورک‌ها در پشت صحنه مورد

استفاده قرار خواهند گرفت) و همچنین سبب بکار افتادن سیستم‌های کش دیتابیس‌های پیشرفته مانند اس کیوال سرور می‌شوند (execution plan کوئری‌های پارامتری در اس کیوال سرور جهت بالا رفتن کارآیی سیستم کش می‌شوند و اهمیتی هم ندارد که حتما رویه ذخیره شده باشند یا خیر).

ادامه دارد ...

نظرات خوانندگان

نویسنده: نیما
تاریخ: ۱۳۸۸/۰۷/۱۸ ۱۹:۳۶:۰۱

سلام استاد نصیری
با تشکر از بزرگواری جنابعالی در شریک کردن ما در تجربیات و دانسته هاتون در مورد قسمت دوم سوال داشتم: اینکه این سه تا کلاس رو تو داکیومنتهای NHibernate گفته شده که برای آزمایش واحد باید ایجاد کرد؟ امکانش هست مرحله به مرحله به زبان ساده تر بفرمایید وقتی آزمایش واحد شروع میشه چه اتفاقی میفته در اینجا؟
ممنون از شما استاد گرامی

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۱۸ ۱۹:۵۳:۱۶

سلام
- ضمن تشکر از لطف شما، بنده استاد نیستم. یک سری مطلب رو از این طرف اون طرف پیدا می‌کنم و با هم تقسیم می‌کنیم. فقط همین و لطفا این لفظ رو دیگر بکار نبرید.
- خیر. می‌شد برای آزمایش یک برنامه کنسول هم نوشت. اما دیگر مرسوم نیست. بجای استفاده از یک برنامه کنسول، آزمایش واحد بنویسید. هم روشی است استاندارد، هم به عنوان مستندات نحوه استفاده از متدهای پروژه می‌تونه مورد استفاده قرار بگیره، هم سبب میشه کد بهتری بنویسید چون مجبور خواهید شد در هم تنیدگی کدهای خودتون رو برای متد تست نوشتن کمتر کنید و هم .... در مقالات مربوطه (تگ unit test سمت راست صفحه) مابقی مزایا، نحوه تولید استفاده و غیره را لطفا مطالعه کنید.

نویسنده: Ahmadreza
تاریخ: ۱۳۸۸/۰۸/۲۱ ۲۱:۰۱:۵۸

سلام
آقای نصیری عزیز

من هم مثل شما از Resharper به عنوان اجرا کننده های تست های NUnit استفاده می کنم ولی من در تست های Pass شده مثل شما Log مربوط به NHibernate رو نمی بینم فقط تست های fail رو میبینم. مخواستم ببینم تنظیم خواصی داره؟ اگه لطفا کنید راهنمایی کنید ممنون میشم.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۱ ۲۱:۱۲:۰۷

سلام
نکته نمایش خروجی SQL فوق، افزودن متد ShowSql است (نمونه آن در کدهای مقاله هست).