عنوان: **آشنایی با NHibernate - قسمت پنجم**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۲۱ ۱۰:۴۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

استفاده از LINQ جهت انجام کوئری‌ها توسط NHibernate

نگارش نهایی 1.0 کتابخانه‌ی LINQ to NHibernate اخیرا (حدود سه ماه قبل) منتشر شده است. در این قسمت قصد داریم با کمک این کتابخانه، اعمال متداول انجام کوئری‌ها را بر روی دیتابیس قسمت قبل انجام دهیم.
توسط این نگارش ارائه شده، کلیه اعمال قابل انجام با criteria API این فریم ورک را می‌توان از طریق LINQ نیز انجام داد (NHibernate برای کار با داده‌ها و جستجوهای پیشرفته بر روی آن‌ها، HQL : Hibernate Query Language و Criteria API را سال‌ها قبل توسعه داده است).

جهت دریافت پروایدر LINQ مخصوص NHibernate به آدرس زیر مراجعه نمائید:

http://sourceforge.net/projects/nhibernate/files

پس از دریافت آن، به همان برنامه کنسول قسمت قبل، دو ارجاع را باید افزود:
الف) ارجاعی به اسمبلی NHibernate.Linq.dll
ب) ارجاعی به اسمبلی استاندارد System.Data.Services.dll دات نت فریم ورک سه و نیم

در ابتدای متد Main برنامه قصد داریم تعدادی مشتری را به دیتابیس اضافه نمائیم. به همین منظور متد AddNewCustomers را به کلاس CDbOperations برنامه کنسول قسمت قبل اضافه نمائید. این متد لیستی از مشتری‌ها را دریافت کرده و آن‌ها را در طی یک تراکنش به دیتابیس اضافه می‌کند:

```
public void AddNewCustomers(params Customer[] customers)
        {
            using (ISession session = _factory.OpenSession())
            {
                using (ITransaction transaction = session.BeginTransaction())
                {
                    foreach (var data in customers)
                        session.Save(data);

                    session.Flush();

                    transaction.Commit();
                }
            }
        }
```

در اینجا استفاده از واژه کلیدی params سبب می‌شود که بجای تعریف الزامی یک آرایه از نوع مشتری‌ها، بتوانیم تعداد دلخواهی پارامتر از نوع مشتری را به این متد ارسال کنیم.

پس از افزودن این ارجاعات، کلاس جدیدی را به نام CLinqTest به برنامه کنسول اضافه نمائید. ساختار کلی این کلاس که قصد استفاده از پروایدر LINQ مخصوص NHibernate را دارد باید به شکل زیر باشد (به کلاس پایه NHibernateContext دقت نمائید):

```
using System.Collections.Generic;
using System.Linq;
using NHibernate;
using NHibernate.Linq;
using NHSample1.Domain;

namespace ConsoleTestApplication
```

```
{
   class CLinqTest : NHibernateContext
   {  }
}
```

اکنون پس از مشخص شدن context یا زمینه، نحوه ایجاد یک کوئری ساده LINQ to NHibernate به صورت زیر می‌تواند باشد:

```
using System.Collections.Generic;
using System.Linq;
using NHibernate;
using NHibernate.Linq;
using NHSample1.Domain;

namespace ConsoleTestApplication
{
   class CLinqTest : NHibernateContext
   {
       ISessionFactory _factory;

       public CLinqTest(ISessionFactory factory)
       {
           _factory = factory;
       }

       public List<Customer> GetAllCustomers()
       {
           using (ISession session = _factory.OpenSession())
           {
               var query = from x in session.Linq<Customer>() select x;
               return query.ToList();
           }
       }
   }
}
```

ابتدا علاوه بر سایر فضاهای نام مورد نیاز، فضای نام NHibernate.Linq به پروژه افزوده می‌شود. سپس از extension متدی به نام Linq بر روی اشیاء ISession از نوع یکی از موجودیت‌های تعریف شده در برنامه در قسمت‌های قبل، می‌توان جهت تهیه کوئری‌های Linq مورد نظر بهره برد.
در این کوئری، لیست تمامی مشتری‌ها بازگشت داده می‌شود.

سپس جهت استفاده و بررسی آن در متد Main برنامه خواهیم داشت:

```
static void Main(string[] args)
       {
           using (ISessionFactory session = Config.CreateSessionFactory(
                 MsSqlConfiguration
                    .MsSql2008
                    .ConnectionString("Data Source=(local);Initial Catalog=HelloNHibernate;Integrated
Security = true")
                    .ShowSql()
               ))
           {

               var customer1 = new Customer()
               {
                   FirstName = "Vahid",
                   LastName = "Nasiri",
                   AddressLine1 = "Addr1",
                   AddressLine2 = "Addr2",
                   PostalCode = "1234",
                   City = "Tehran",
                   CountryCode = "IR"
               };

               var customer2 = new Customer()
               {
                   FirstName = "Ali",
                   LastName = "Hasani",
                   AddressLine1 = "Addr..1",
                   AddressLine2 = "Addr..2",
                   PostalCode = "4321",
                   City = "Shiraz",
```

```
                CountryCode = "IR"
            };

            var customer3 = new Customer()
            {
                FirstName = "Mohsen",
                LastName = "Shams",
                AddressLine1 = "Addr...1",
                AddressLine2 = "Addr...2",
                PostalCode = "5678",
                City = "Ahwaz",
                CountryCode = "IR"
            };

            CDbOperations db = new CDbOperations(session);
            db.AddNewCustomers(customer1, customer2, customer3);

            CLinqTest lt = new CLinqTest(session);
            foreach (Customer customer in lt.GetAllCustomers())
            {
                Console.WriteLine("Customer: LastName = {0}", customer.LastName);
            }
        }

        Console.WriteLine("Press a key...");
        Console.ReadKey();

    }
```

در این متد ابتدا تعدادی رکورد تعریف و سپس به دیتابیس اضافه شدند. در ادامه لیست تمامی آن‌ها از دیتابیس دریافت و نمایش داده می‌شود.

مهمترین مزیت استفاده از LINQ در این نوع کوئری‌ها نسبت به روش‌های دیگر، استفاده از کدهای strongly typed دات نتی تحت نظر کامپایلر است، نسبت به رشته‌های معمولی SQL که کامپایلر کنترلی را بر روی آن‌ها نمی‌تواند داشته باشد (برای مثال اگر نوع یک ستون تغییر کند یا نام آن، در حالت استفاده از LINQ بلافاصله یک خطا را از کامپایلر جهت تصحیح مشکلات دریافت خواهیم کرد که این مورد در زمان استفاده از یک رشته معمولی صادق نیست). همچنین مزیت فراهم بودن Intellisense را حین نوشتن کوئری‌هایی از این دست نیز نمی‌توان ندید گرفت.

مثالی دیگر:
لیست تمام مشتری‌های شیرازی را نمایش دهید:
ابتدا متد GetCustomersByCity را به کلاس CLinqTest فوق اضافه می‌کنیم:

```
public List<Customer> GetCustomersByCity(string city)
    {
        using (ISession session = _factory.OpenSession())
        {
            var query = from x in session.Linq<Customer>()
                        where x.City == city
                        select x;
            return query.ToList();
        }
    }
```

سپس برای استفاده از آن، چند سطر ساده زیر به ادامه متد Main اضافه می‌شوند:

```
foreach (Customer customer in lt.GetCustomersByCity("Shiraz"))
            {
                Console.WriteLine("Customer: LastName = {0}", customer.LastName);
            }
```

یکی دیگر از مزایای استفاده از LINQ to NHibernate ، امکان بکارگیری LINQ بر روی تمامی دیتابیس‌های پشتیبانی شده توسط NHibernate است؛ برای مثال مای اس کیوال، اوراکل و ....
لیست کامل دیتابیس‌های پشتیبانی شده توسط NHibernate را در این آدرس می‌توانید مشاهده نمائید. (البته به نظر لیست آن،

آنچنان هم به روز نیست؛ چون در نگارش آخر NHibernate ، پشتیبانی از اس کیوال سرور 2008 هم اضافه شده است)

نکته:
در کوئری‌های مثال‌های فوق همواره باید session.Linq<T> را ذکر کرد. اگر علاقمند بودید شبیه به روشی که در LINQ to SQL موجود است مثلا db.TableName بجای session.Linq<T> در کوئری‌ها ذکر گردد، می‌توان اصلاحاتی را به صورت زیر اعمال کرد:
یک کلاس جدید را به نام SampleContext به برنامه کنسول جاری با محتویات زیر اضافه نمائید:

```
using System.Linq;
using NHibernate;
using NHibernate.Linq;
using NHSample1.Domain;

namespace ConsoleTestApplication
{
   class SampleContext : NHibernateContext
   {
       public SampleContext(ISession session)
           : base(session)
       { }

       public IOrderedQueryable<Customer> Customers
       {
           get { return Session.Linq<Customer>(); }
       }

       public IOrderedQueryable<Employee> Employees
       {
           get { return Session.Linq<Employee>(); }
       }

       public IOrderedQueryable<Order> Orders
       {
           get { return Session.Linq<Order>(); }
       }

       public IOrderedQueryable<OrderItem> OrderItems
       {
           get { return Session.Linq<OrderItem>(); }
       }

       public IOrderedQueryable<Product> Products
       {
           get { return Session.Linq<Product>(); }
       }
   }
}
```

در این کلاس به ازای تمام موجودیت‌های تعریف شده در پوشه domain برنامه اصلی خود (همان NHSample1 قسمت‌های اول و دوم)، یک متد از نوع IOrderedQueryable را باید تشکیل دهیم که پیاده سازی آن‌را ملاحظه می‌نمائید.
سپس بازنویسی متد GetCustomersByCity بر اساس SampleContext فوق به صورت زیر خواهد بود که به کوئری‌های LINQ to SQL بسیار شبیه است:

```
using System.Collections.Generic;
using System.Linq;
using NHibernate;
using NHSample1.Domain;

namespace ConsoleTestApplication
{
   class CSampleContextTest
   {
       ISessionFactory _factory;

       public CSampleContextTest(ISessionFactory factory)
       {
           _factory = factory;
       }

       public List<Customer> GetCustomersByCity(string city)
       {
           using (ISession session = _factory.OpenSession())
```

```
            {
                using (SampleContext db = new SampleContext(session))
                {
                    var query = from x in db.Customers
                                where x.City == city
                                select x;
                    return query.ToList();
                }
            }
        }
    }
}
```

دریافت سورس برنامه تا این قسمت

و در تکمیل این بحث، می‌توان به لیستی از 101 مثال LINQ ارائه شده در MSDN اشاره کرد که یکی از بهترین و سریع ترین مراجع یادگیری مبحث LINQ است.

ادامه دارد ...

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۱۲/۱۳ ۱۴:۳۴:۲۴

در مورد LINQ to NHibernate در نگارش‌های اخیر آن کمی تغییر وجود داشته که نیاز است مطلب زیر را مطالعه بفرمائید:
http://blogs.imeta.co.uk/sstrong/archive/2009/12/16/824.aspx

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۰۴ ۲۳:۱۵:۴۸

NH 3.0 پشتیبانی یکپارچه‌ای را از LINQ ارائه می‌دهد و دیگر نیازی نیست مانند نگارش 2 آن پروایدر مخصوصی را جداگانه دریافت کرد. آن پروایدر قدیمی هم به نظر کنار گذاشته شده و از یک کتابخانه‌ی پخته‌تر به نام Remotion Linq Library استفاده گردیده است (+) .
در این نگارش برای دسترسی به IQueryable interface می‌توان از متد session.Query استفاده کرد (بجای session.Linq نگارش قبلی).

نویسنده: fateme
تاریخ: ۱۳۸۹/۱۱/۱۲ ۱۱:۵۹:۲۸

سلام جناب آقای نصیری
ممنون از آموزشهای بسیار عالیتون
لینک دریافت پروایدر LINQ که گذاشتید موقع دانلود error میده می تونید لینک دیگه ای معرفی کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۱۲ ۱۲:۲۸:۲۶

در کامنت‌های فوق توضیح دادم. این توضیحات مربوط به نگارش 2 بود. الان نگارش 3 را که دریافت کنید LINQ با آن یکپارچه است و نیازی به دریافت پروایدر کمکی نیست و پروایدر قدیمی هم منسوخ شده و دیگر ادامه نخواهد یافت.

نویسنده: fateme
تاریخ: ۱۳۸۹/۱۱/۱۲ ۱۴:۳۴:۲۱

جناب آقای نصیری ممنون از راهنماییتون
یه سوال دیگه ای که دارم اینکه برنامه من کلاس پایه NHibernateContext نمیشناسه ممنون میشم اگه مجددا راهنماییم کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۱۲ ۱۵:۰۰:۳۲

به NHibernateContext در NH 3.0 نیازی نیست. آن‌را از مثال‌ها حذف کنید. فقط بجای Linq شما Query خواهید داشت (یک تغییر نام مختصر به همراه ساده سازی نحوه استفاده).