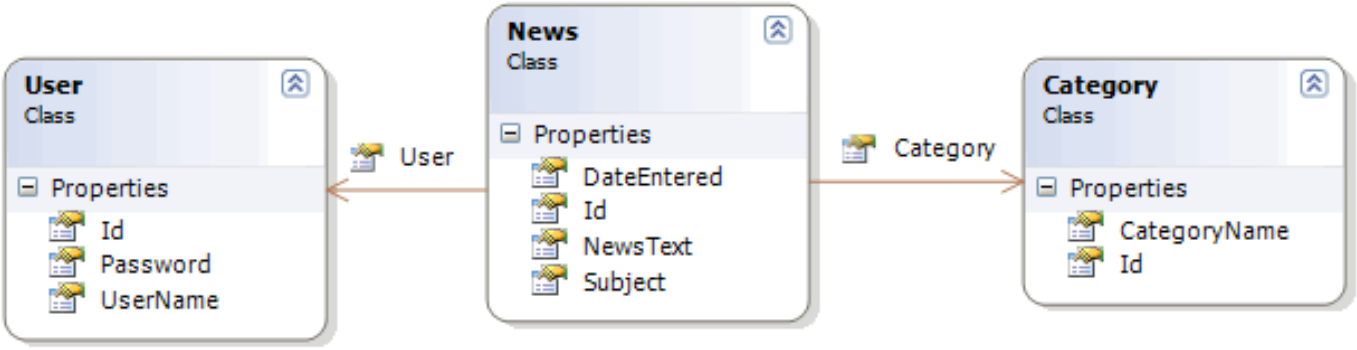
عنوان: **آشنایی با NHibernate - قسمت ششم**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۲۲ ۱۱:۰۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

آشنایی با Automapping در فریم ورک Fluent NHibernate

اگر قسمت‌های قبل را دنبال کرده باشید، احتمالا به پروسه طولانی ساخت نگاشت‌ها توجه کرده‌اید. با کمک فریم ورک Fluent NHibernate می‌توان پروسه نگاشت domain model خود را به data model متناظر آن به صورت خودکار نیز انجام داد و قسمت عمده‌ای از کار به این صورت حذف خواهد شد. (این مورد یکی از تفاوت‌های مهم NHibernate با نمونه‌های مشابهی است که مایکروسافت تا تاریخ نگارش این مقاله ارائه داده است. برای مثال در نگارش‌های فعلی LINQ to SQL یا Entity framework ، اول دیتابیس مطرح است و بعد ساخت کد از روی آن، در حالیکه در اینجا ابتدا کد و طراحی سیستم مطرح است و بعد نگاشت آن به سیستم داده‌ای و دیتابیس)

امروز قصد داریم یک سیستم ساده ثبت خبر را از صفر با NHibernate پیاده سازی کنیم و همچنین مروری داشته باشیم بر قسمت‌های قبلی.



مطابق کلاس دیاگرام فوق، این سیستم از سه کلاس خبر، کاربر ثبت کننده‌ی خبر و گروه خبری مربوطه تشکیل شده است.

ابتدا یک پروژه کنسول جدید را به نام NHSample2 آغاز کنید. سپس ارجاعاتی را به اسمبلی‌های زیر به آن اضافه نمائید:

FluentNHibernate.dll

NHibernate.dll

NHibernate.ByteCode.Castle.dll

NHibernate.Linq.dll

و ارجاعی به اسمبلی استاندارد System.Data.Services.dll دات نت فریم ورک سه و نیم

سپس پوشه‌ای را به نام Domain به این پروژه اضافه نمائید (کلیک راست روی نام پروژه در VS.Net و سپس مراجعه به منوی Add->New folder). در این پوشه تعاریف موجودیت‌های برنامه را قرار خواهیم داد. سه کلاس جدید User ، Category و News را در این پوشه ایجاد نمائید. محتویات این سه کلاس به شرح زیر هستند:

```
namespace NHSample2.Domain
{
  public class User
  {
      public virtual int Id { get; set; }
```

```
      public virtual string UserName { get; set; }
      public virtual string Password { get; set; }
  }
}
```

```
namespace NHSample2.Domain
{
  public class Category
  {
      public virtual int Id { get; set; }
      public virtual string CategoryName { get; set; }
  }
}
```

```
using System;

namespace NHSample2.Domain
{
  public class News
  {
      public virtual Guid Id { get; set; }
      public virtual string Subject { get; set; }
      public virtual string NewsText { get; set; }
      public virtual DateTime DateEntered { get; set; }
      public virtual Category Category { get; set; }
      public virtual User User { get; set; }
  }
}
```

همانطور که در قسمت‌های قبل نیز ذکر شد، تمام خواص پابلیک کلاس‌های Domain ما به صورت virtual تعریف شده‌اند تا lazy loading را در NHibernate فعال سازیم. در حالت lazy loading ، اطلاعات تنها زمانیکه به آن‌ها نیاز باشد بارگذاری خواهند شد. این مورد در حالتیکه نیاز به نمایش اطلاعات تنها یک شیء وجود داشته باشد بسیار مطلوب می‌باشد، یا هنگام ثبت و به روز رسانی اطلاعات نیز یکی از بهترین روش‌ها است. اما زمانیکه با لیستی از اطلاعات سروکار داشته باشیم باعث کاهش افت کارآیی خواهد شد زیرا برای مثال نمایش آن‌ها سبب خواهد شد که 100 ها کوئری دیگر جهت دریافت اطلاعات هر رکورد در حال نمایش اجرا شود (مفهوم دسترسی به اطلاعات تنها در صورت نیاز به آن‌ها). Lazy loading و eager loading (همانند مثال‌های قبلی) هر دو در NHibernate به سادگی قابل تنظیم هستند (برای مثال LINQ to SQL به صورت پیش فرض همواره lazy load است و تا این تاریخ راه استانداردی برای امکان تغییر و تنظیم این مورد پیش بینی نشده است).

اکنون کلاس جدید Config را به برنامه اضافه نمائید:

```
using FluentNHibernate.Automapping;
using FluentNHibernate.Cfg;
using FluentNHibernate.Cfg.Db;
using NHibernate;
using NHibernate.Cfg;
using NHibernate.Tool.hbm2ddl;

namespace NHSample2
{
  class Config
  {
      public static Configuration GenerateMapping(IPersistenceConfigurer dbType)
      {
          var cfg = dbType.ConfigureProperties(new Configuration());

          new AutoPersistenceModel()
              .Where(x => x.Namespace.EndsWith("Domain"))
              .AddEntityAssembly(typeof(NHSample2.Domain.News).Assembly).Configure(cfg);

          return cfg;
      }

      public static void GenerateDbScript(Configuration config, string filePath)
      {
          bool script = true;//فقط اسکریپت دیتابیس تولید گردد
```

```
            bool export = false;//نیازی نیست بر روی دیتابیس هم اجرا شود
            new SchemaExport(config).SetOutputFile(filePath).Create(script, export);
        }

        public static void BuildDbSchema(Configuration config)
        {
            bool script = false;//آیا خروجی در کنسول هم نمایش داده شود
            bool export = true;//آیا بر روی دیتابیس هم اجرا شود
            bool drop = false;//آیا اطلاعات موجود دراپ شوند
            new SchemaExport(config).Execute(script, export, drop);
        }

        public static void CreateSQL2008DbPlusScript(string connectionString, string filePath)
        {
            Configuration cfg =
                GenerateMapping(
                MsSqlConfiguration
                .MsSql2008
                .ConnectionString(connectionString)
                .ShowSql()
                );
            GenerateDbScript(cfg, filePath);
            BuildDbSchema(cfg);
        }

        public static ISessionFactory CreateSessionFactory(IPersistenceConfigurer dbType)
        {
            return
               Fluently.Configure().Database(dbType)
                .Mappings(m => m.AutoMappings
                            .Add(
                                new AutoPersistenceModel()
                                    .Where(x => x.Namespace.EndsWith("Domain"))
                                    .AddEntityAssembly(typeof(NHSample2.Domain.News).Assembly))
                                    )
                 .BuildSessionFactory();
        }
    }
}
```

در متد GenerateMapping از قابلیت Automapping موجود در فریم ورک Fluent Nhibernate استفاده شده است (بدون نوشتن حتی یک سطر جهت تعریف این نگاشت‌ها). این متد نوع دیتابیس مورد نظر را جهت ساخت تنظیمات خود دریافت می‌کند. سپس با کمک کلاس AutoPersistenceModel این فریم ورک، به صورت خودکار از اسمبلی برنامه نگاشت‌های لازم را به کلاس‌های موجود در پوشه Domain ما اضافه می‌کند (مرسوم است که این پوشه در یک پروژه Class library مجزا تعریف شود که در این برنامه جهت سهولت کار در خود برنامه قرار گرفته است). قسمت Where ذکر شده به این جهت معرفی گردیده است تا Fluent Nhibernate برای تمامی کلاس‌های موجود در اسمبلی جاری، سعی در تعریف نگاشت‌های لازم نکند. این نگاشت‌ها تنها به کلاس‌های موجود در پوشه دومین ما محدود شده‌اند.

سه متد بعدی آن، جهت ایجاد اسکریپت دیتابیس از روی این نگاشت‌های تعریف شده و سپس اجرای این اسکریپت بر روی دیتابیس جاری معرفی شده، تهیه شده‌اند. برای مثال CreateSQL2008DbPlusScript یک مثال ساده از استفاده دو متد قبلی جهت ایجاد اسکریپت و دیتابیس متناظر اس کیوال سرور 2008 بر اساس نگاشت‌های برنامه است.

با متد CreateSessionFactory در قسمت‌های قبل آشنا شده‌اید. تنها تفاوت آن در این قسمت، استفاده از کلاس AutoPersistenceModel جهت تولید خودکار نگاشت‌ها است.

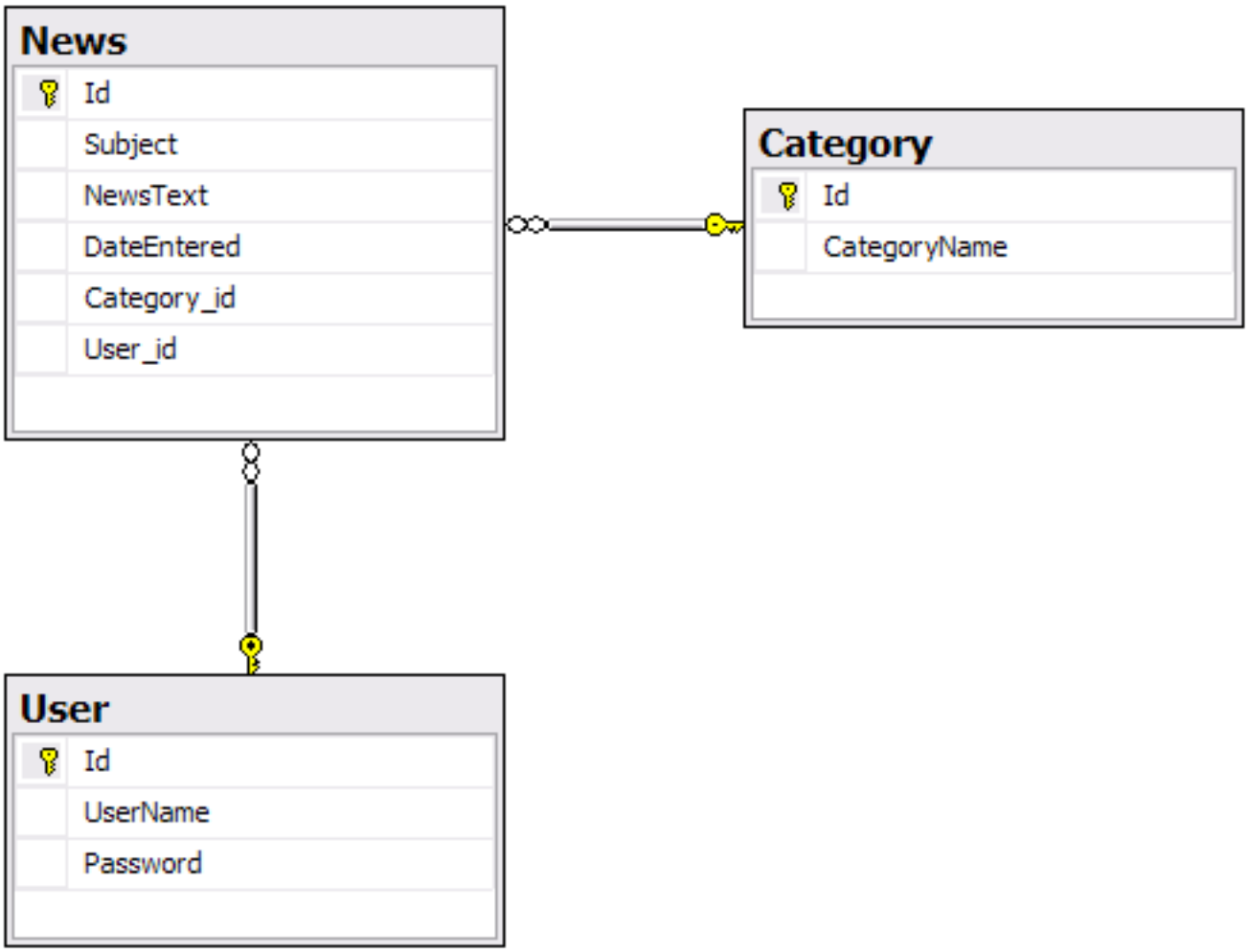
در ادامه دیتابیس متناظر با موجودیت‌های برنامه را ایجاد خواهیم کرد:

```
using System;

namespace NHSample2
{
  class Program
  {
      static void Main(string[] args)
      {
         Config.CreateSQL2008DbPlusScript(
              "Data Source=(local);Initial Catalog=HelloNHibernate;Integrated Security = true",
              "db.sql");
```

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

پس از اجرای برنامه، ابتدا فایل اسکریپت دیتابیس به نام db.sql در پوشه اجرایی برنامه تشکیل خواهد شد و سپس این اسکریپت به صورت خودکار بر روی دیتابیس معرفی شده اجرا می‌گردد. دیتابیس دیاگرام حاصل را در شکل زیر می‌توانید ملاحظه نمائید:



همچنین اسکریپت تولید شده آن، صرفنظر از عبارات drop اولیه، به صورت زیر است:

```
create table [Category] (
      Id INT IDENTITY NOT NULL,
     CategoryName NVARCHAR(255) null,
     primary key (Id)
  )

  create table [User] (
      Id INT IDENTITY NOT NULL,
```

```
    UserName NVARCHAR(255) null,
    Password NVARCHAR(255) null,
    primary key (Id)
  )

  create table [News] (
      Id UNIQUEIDENTIFIER not null,
    Subject NVARCHAR(255) null,
    NewsText NVARCHAR(255) null,
    DateEntered DATETIME null,
    Category_id INT null,
    User_id INT null,
    primary key (Id)
  )

  alter table [News]
      add constraint FKE660F9E1C9CF79
      foreign key (Category_id)
      references [Category]

  alter table [News]
      add constraint FKE660F95C1A3C92
      foreign key (User_id)

      references [User]
```

اکنون یک سری گروه خبری، کاربر و خبر را به دیتابیس خواهیم افزود:

```
using System;
using FluentNHibernate.Cfg.Db;
using NHibernate;
using NHSample2.Domain;

namespace NHSample2
{
  class Program
  {
      static void Main(string[] args)
      {
          using (ISessionFactory sessionFactory = Config.CreateSessionFactory(
                  MsSqlConfiguration
                      .MsSql2008
                      .ConnectionString("Data Source=(local);Initial Catalog=HelloNHibernate;Integrated
Security = true")
                      .ShowSql()
                  ))
          {
              using (ISession session = sessionFactory.OpenSession())
              {
                  using (ITransaction transaction = session.BeginTransaction())
                  {
                      //با توجه به کلیدهای خارجی تعریف شده ابتدا باید گروه‌ها را اضافه کرد
                      Category ca = new Category() {  CategoryName = "Sport" };
                      session.Save(ca);
                      Category ca2 = new Category() { CategoryName = "IT" };
                      session.Save(ca2);
                      Category ca3 = new Category() { CategoryName = "Business" };
                      session.Save(ca3);

                      //سپس یک کاربر را به دیتابیس اضافه می‌کنیم
                      User u = new User() {  Password = "123$5@1", UserName = "VahidNasiri" };
                      session.Save(u);

                      //اکنون می‌توان یک خبر جدید را ثبت کرد

                      News news = new News()
                      {
                          Category = ca,
                          User = u,
                          DateEntered = DateTime.Now,
                          Id = Guid.NewGuid(),
                          NewsText = "متن خبر جدید",
                          Subject = "عنوانی دلخواه"
                      };
                      session.Save(news);
```

```
                    transaction.Commit(); //پایان تراکنش
                }
            }
        }

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

جهت بررسی انجام عملیات ثبت هم می‌توان به دیتابیس مراجعه کرد، برای مثال:

```
SELECT TOP 1000 [Id]
      ,[Subject]
      ,[NewsText]
      ,[DateEntered]
      ,[Category_id]
      ,[User_id]
  FROM [HelloNHibernate].[dbo].[News]
```

Results | Messages

	Id	Subject	NewsText	DateEntered	Category_id	User_id
1	0552730B-83A...	عنواني دلخواه	متن خبر جديد	2009-10-13 21:55:08.000	1	1

و یا می‌توان از LINQ استفاده کرد:
برای مثال کاربر VahidNasiri تعریف شده را یافته، اطلاعات آن‌را نمایش دهید؛ سپس نام او را به Vahid ویرایش کرده و دیتابیس را به روز کنید.

برای اینکه کوئری‌های LINQ ما شبیه به LINQ to SQL شوند، کلاس NewsContext را به صورت ذیل تشکیل می‌دهیم. این کلاس از کلاس پایه NHibernateContext مشتق شده و سپس به ازای تمام موجودیت‌های برنامه، یک متد از نوع IOrderedQueryable را تشکیل خواهیم داد.

```
using System.Linq;
using NHibernate;
using NHibernate.Linq;
using NHSample2.Domain;

namespace NHSample2
{
  class NewsContext : NHibernateContext
  {
      public NewsContext(ISession session)
          : base(session)
      { }

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

```
    public IOrderedQueryable<Category> Categories
    {
        get { return Session.Linq<Category>(); }
    }

    public IOrderedQueryable<User> Users
    {
        get { return Session.Linq<User>(); }
    }
  }
}
```

اکنون جهت یافتن کاربر و به روز رسانی اطلاعات او در دیتابیس خواهیم داشت:

```
using System;
using FluentNHibernate.Cfg.Db;
using NHibernate;
using System.Linq;
using NHSample2.Domain;

namespace NHSample2
{
  class Program
  {
      static void Main(string[] args)
      {
          using (ISessionFactory sessionFactory = Config.CreateSessionFactory(
                  MsSqlConfiguration
                      .MsSql2008
                      .ConnectionString("Data Source=(local);Initial Catalog=HelloNHibernate;Integrated
Security = true")
                      .ShowSql()
                  ))
          {
              using (ISession session = sessionFactory.OpenSession())
              {
                  using (ITransaction transaction = session.BeginTransaction())
                  {
                      using (NewsContext db = new NewsContext(session))
                      {
                          var query = from x in db.Users
                                      where x.UserName == "VahidNasiri"
                                      select x;

                          //اگر چیزی یافت شد
                          if (query.Any())
                          {
                              User vahid = query.First();
                              //نمایش اطلاعات کاربر
                              Console.WriteLine("Id: {0}, UserName: {0}", vahid.Id, vahid.UserName);
                              //به روز رسانی نام کاربر
                              vahid.UserName = "Vahid";
                              session.Update(vahid);

                              transaction.Commit(); //پایان تراکنش
                          }
                      }
                  }
              }
          }

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

مباحث تکمیلی AutoMapping

اگر به اسکریپت دیتابیس تولید شده دقت کرده باشید، عملیات AutoMapping یک سری پیش فرض‌هایی را اعمال کرده است. برای مثال فیلد Id را از نوع identity و به صورت کلید تعریف کرده، یا رشته‌ها را به صورت nvarchar با طول 255 ایجاد نموده است. امکان سفارشی سازی این موارد نیز وجود دارد.

مثال:

```
using FluentNHibernate.Conventions.Helpers;

      public static Configuration GenerateMapping(IPersistenceConfigurer dbType)
      {
          var cfg = dbType.ConfigureProperties(new Configuration());

          new AutoPersistenceModel()
          .Conventions.Add()
              .Where(x => x.Namespace.EndsWith("Domain"))
               .Conventions.Add(
                 PrimaryKey.Name.Is(x => "ID"),
                 DefaultLazy.Always(),
                 ForeignKey.EndsWith("ID"),
                 Table.Is(t => "tbl" + t.EntityType.Name)
                 )
              .AddEntityAssembly(typeof(NHSample2.Domain.News).Assembly)
              .Configure(cfg);

          return cfg;
      }
```

تابع GenerateMapping معرفی شده را اینجا با قسمت Conventions.Add تکمیل کرده‌ایم. به این صورت دقیقا مشخص شده است که فیلدهایی با نام ID باید primary key در نظر گرفته شوند، همواره lazy loading صورت گیرد و نام کلید خارجی به ID ختم شود. همچنین نام جداول با tbl شروع گردد.
روش دیگری نیز برای معرفی این قرار دادها و پیش فرض‌ها وجود دارد. فرض کنید می‌خواهیم طول رشته پیش فرض را از 255 به 500 تغییر دهیم. برای اینکار باید اینترفیس IPropertyConvention را پیاده سازی کرد:

```
using FluentNHibernate.Conventions;
using FluentNHibernate.Conventions.Instances;

namespace NHSample2.Conventions
{
  class MyStringLengthConvention : IPropertyConvention
  {
      public void Apply(IPropertyInstance instance)
      {
          instance.Length(500);
      }
  }
}
```

سپس نحوه‌ی معرفی آن به صورت زیر خواهد بود:

```
public static Configuration GenerateMapping(IPersistenceConfigurer dbType)
      {
          var cfg = dbType.ConfigureProperties(new Configuration());

          new AutoPersistenceModel()
          .Conventions.Add()
              .Where(x => x.Namespace.EndsWith("Domain"))
               .Conventions.Add<MyStringLengthConvention>()
              .AddEntityAssembly(typeof(NHSample2.Domain.News).Assembly)
              .Configure(cfg);

          return cfg;
      }
```

نکته:
اگر برای یافتن اطلاعات بیشتر در این مورد در وب جستجو کنید، اکثر مثال‌هایی را که مشاهده خواهید کرد بر اساس نگارش بتای fluent NHibernate هستند و هیچکدام با نگارش نهایی این فریم ورک کار نمی‌کنند. در نگارش رسمی نهایی ارائه شده، تغییرات

بسیاری صورت گرفته که آن‌ها را در این آدرس می‌توان مشاهده کرد.

دریافت سورس برنامه قسمت ششم

ادامه دارد ...

نظرات خوانندگان

نویسنده: Majid
تاریخ: ۱۳۸۸/۱۲/۰۹ ۱۲:۴۱:۵۴

با سلام، می‌خواستم بدانم چطور می‌شود توسط FNH تعریف کرد که در یک جدول بیشتر از یک PK داشته باشیم.
متشکرم.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۱۲/۰۹ ۱۴:۳۴:۲۱

سلام
نحوه پیاده سازی کامپوزیت کی در FHN:
http://devlicio.us/blogs/derik_whittaker/archive/2009/01/16/using-fluentnhibernate-to-map-composite-keys-for-a-table.aspx
و اینکه چرا کامپوزیت کی خوب نیست اساسا:
http://codebetter.com/blogs/jeremy.miller/archive/2007/02/01/Composite-keys-are-evil.aspx

نویسنده: Majid
تاریخ: ۱۳۸۸/۱۲/۰۹ ۱۷:۲۷:۰۲

سپاس بخاطر پاسخگویی سریع

نویسنده: Ahmad
تاریخ: ۱۳۸۹/۰۱/۰۵ ۱۶:۲۵:۰۳

اگر بخواهیم از Criteria استفاده کنیم(selectهای joinدار) روابط بین جداول را چگونه تشخیص می دهد؟(با استفاده از AutoMapping) اگر ممکن است مثالی ارائه بفرمائید.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۱/۰۵ ۱۷:۰۷:۱۰

تشخیص روابط بین جداول یعنی همان mapping خودکار، یعنی همان نحوه‌ی تعریف کلاس‌های شما و برقراری روابطی که در طی چند قسمت مثال زده شد. سیستم پیش فرض NHibernate بر اساس اول طراحی کلاس‌ها و بعد ایجاد ارتباط با دیتابیس است که اینجا به صورت خودکار صورت می‌گیرد.
برای مثال در قسمت هشتم یک سیستم many-to-many مثال زده شده است به همراه کوئری‌هایی از نوع Linq . اینجا فقط تعریف کلاس‌هایی که بیانگر روابط many-to-many باشند مهم است؛ نحوه‌ی نگاشت خودکار آن‌ها به دیتابیس کار Fluent NHibernate است. (از این نوع مثال‌ها در هر قسمت پیاده سازی شده)
جزئیات ریز نحوه‌ی نگاشت خودکار با مطالعه‌ی سورس کتابخانه NHibernate و مشتقات آن قابل درک است (برای علاقمندان).
ضمنا فرقی نمی‌کند از Linq قسمت پنجم استفاده کنید یا هر روش موجود دیگری برای کوئری گرفتن (زمانیکه Linq هست و نگارش‌های جدید آن برای NHibernate پیشرفت زیادی داشته، چرا روش‌های دیگر؟).

نویسنده: Ahmad
تاریخ: ۱۳۸۹/۰۱/۰۵ ۱۷:۵۳:۴۳

بسیار عالی
اما مثل اینکه Linq با NH مشکل داره:(در Join)
http://guildsocial.web703.discountasp.net/dasblogce/2009/07/29/LinqToNHibernateJoins.aspx

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۱/۰۵ ۱۹:۲۳:۵۹

یک سری وبلاگ در مورد NHibernate هست که مربوط به توسعه دهنده‌های اصلی آن است و مرجع محسوب می‌شوند. برای مثال:

http://ayende.com/Blog

در این پست زیر قید شده که هر آنچه که در criteria API پشتیبانی می‌شود در نگارش یک LINQ آن هم وجود دارد (بنابراین group joins or subqueries پشتیبانی نمی‌شود چون در criteria API وجود ندارد+ join هم به نظر هنوز تکمیل نشده).

http://ayende.com/Blog/archive/2009/07/26/nhibernate-linq-1.0-released.aspx

وبلاگ توسعه دهنده‌ی اصلی LINQ برای NHibernate

http://blogs.imeta.co.uk/sstrong/Default.aspx

join هم اکنون در نگارش بتای جدید آن کار می‌کند:

http://blogs.imeta.co.uk/sstrong/archive/2009/12/16/823.aspx

وبلاگ یکی از مدیر پروژه‌های NHibernate

http://fabiomaulo.blogspot.com

نویسنده: Ahmad
تاریخ: ۱۳۸۹/۰۱/۰۵ ۲۳:۱۰:۳۷

Thanks