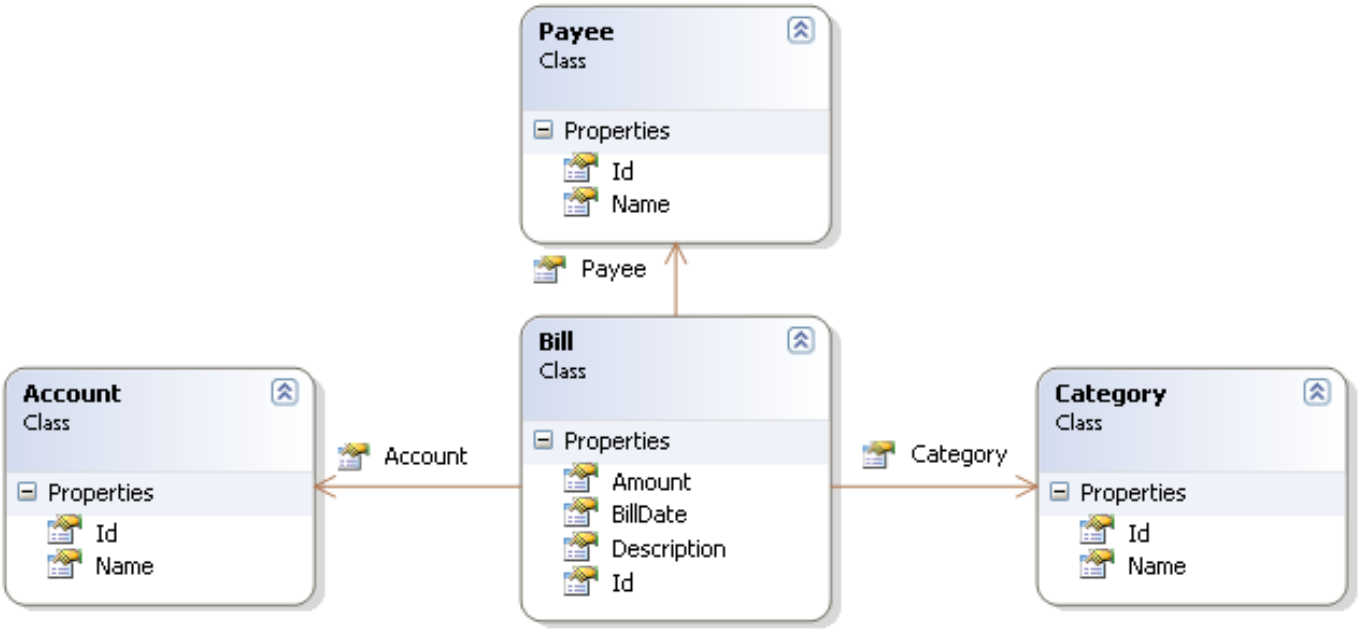
عنوان: **مقدار دهی کلیدهای خارجی در NHibernate و Entity framework**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۰۶ ۱۲:۵۴:۰۰
آدرس: www.dotnettips.info
گروه‌ها: NHibernate

ORM های NHibernate و Entity framework روش‌های متفاوتی را برای به روز رسانی کلید خارجی با حداقل رفت و برگشت به دیتابیس ارائه می‌دهند که در ادامه معرفی خواهند شد.

صورت مساله:
فرض کنید می‌خواهیم برنامه‌ای را بنویسیم که ریز پرداخت‌های روزانه‌ی ما را ثبت کند. برای اینکار حداقل به یک جدول گروه‌های اقلام خریداری شده، یک جدول حساب‌های تامین کننده‌ی مخارج، یک جدول فروشنده‌ها و نهایتا یک جدول صورتحساب‌های پرداختی بر اساس جداول ذکر شده نیاز خواهد بود.

الف) بررسی مدل برنامه

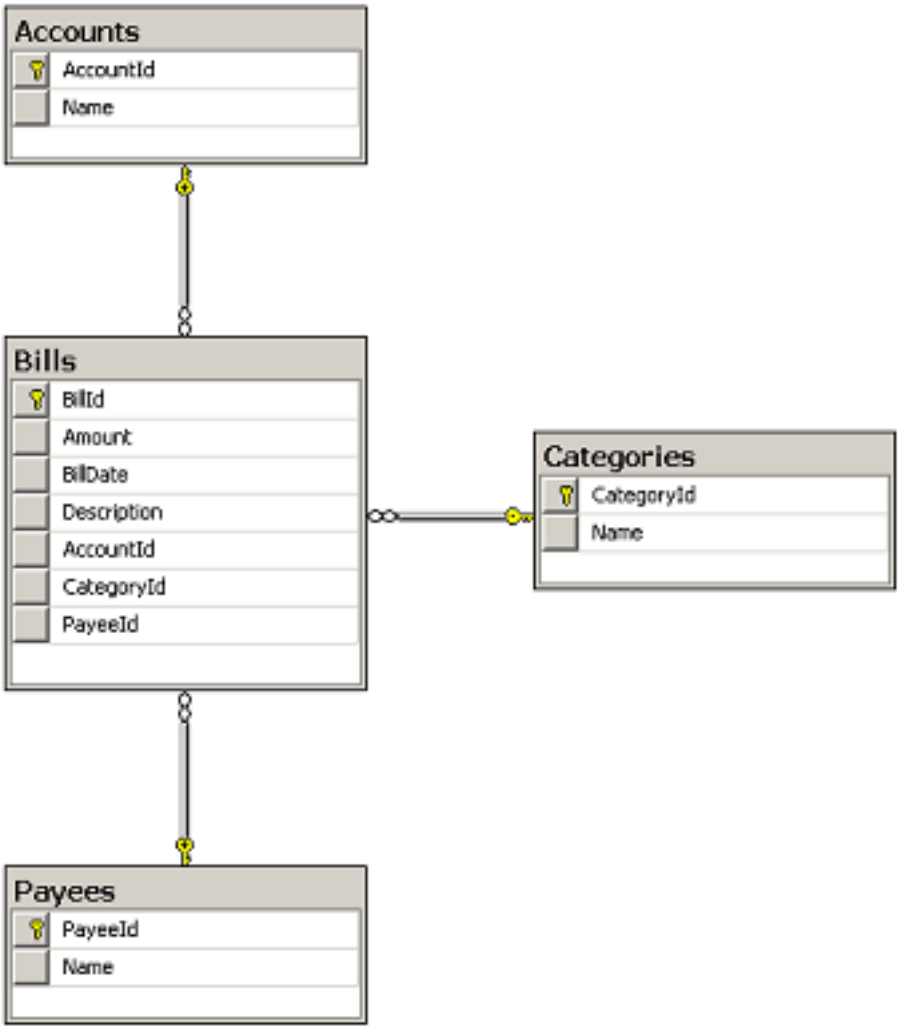


در اینجا جهت تعریف ویژگی‌ها یا Attributes تعریف شده در این کلاس‌ها از NHibernate validator استفاده شده (+). مزیت اینکار هم علاوه بر اعتبارسنجی سمت کلاینت (پیش از تبادل اطلاعات با بانک اطلاعاتی)، تولید جداولی با همین مشخصات است. برای مثال NHibernate Fluent بر اساس ویژگی Length تعریف شده با طول حداکثر 120 ، یک فیلد nvarchar با همین طول را ایجاد می‌کند.

```
public class Account
{
	public virtual int Id { get; set; }

	[NotNullNotEmpty]
	[Length(Min = 3, Max = 120, Message = "طول نام باید بین 3 و 120 کاراکتر باشد")]
	public virtual string Name { get; set; }
}
```

```
public class Category
{
	public virtual int Id { get; set; }

	[NotNullNotEmpty]
	[Length(Min = 3, Max = 130, Message = "طول نام باید بین 3 و 130 کاراکتر باشد")]
	public virtual string Name { get; set; }
}

public class Payee
{
	public virtual int Id { get; set; }

	[NotNullNotEmpty]
	[Length(Min = 3, Max = 120, Message = "طول نام باید بین 3 و 120 کاراکتر باشد")]
	public virtual string Name { get; set; }
}

public class Bill
{
	public virtual int Id { get; set; }

	[NotNull]
	public virtual Account Account { get; set; }

	[NotNull]
	public virtual Category Category { get; set; }

	[NotNull]
	public virtual Payee Payee { get; set; }

	[NotNull]
	public virtual decimal Amount { set; get; }

	[NotNull]
	public virtual DateTime BillDate { set; get; }

	[NotNullNotEmpty]
	[Length(Min = 1, Max = 500, Message = "طول توضیحات باید بین 1 و 500 کاراکتر باشد")]
	public virtual string Description { get; set; }
}
```

ب) ساختار جداول متناظر (تولید شده به صورت خودکار توسط Fluent NHibernate در اینجا)



در مورد نحوه‌ی استفاده از ویژگی AutoMapping و همچنین تولید خودکار ساختار بانک اطلاعاتی از روی جداول در NHibernate قبلا توضیح داده شده است. البته بدیهی است که ترکیب مقاله‌ی Validation و آشنایی با AutoMapping در اینجا جهت اعمال ویژگی‌ها باید بکار گرفته شود که در همان مقاله‌ی Validation مفصل توضیح داده شده است.

نکته‌ی مهم database schema تولیدی، کلیدهای خارجی (foreign key) تعریف شده بر روی جدول Bills است (همان AccountId، CategoryId و PayeeId تعریف شده) که به primary key جداول متناظر اشاره می‌کند.

```
create table Accounts (
       AccountId INT IDENTITY NOT NULL,
      Name NVARCHAR(120) not null,
      primary key (AccountId)
   )

   create table Bills (
       BillId INT IDENTITY NOT NULL,
      Amount DECIMAL(19,5) not null,
      BillDate DATETIME not null,
      Description NVARCHAR(500) not null,
      AccountId INT not null,
      CategoryId INT not null,
      PayeeId INT not null,
      primary key (BillId)
   )

   create table Categories (
       CategoryId INT IDENTITY NOT NULL,
      Name NVARCHAR(130) not null,
      primary key (CategoryId)
   )
```

```
create table Payees (
    PayeeId INT IDENTITY NOT NULL,
   Name NVARCHAR(120) not null,
   primary key (PayeeId)
)

alter table Bills
    add constraint fk_Account_Bill
    foreign key (AccountId)
    references Accounts

alter table Bills
    add constraint fk_Category_Bill
    foreign key (CategoryId)
    references Categories

alter table Bills
    add constraint fk_Payee_Bill
    foreign key (PayeeId)
    references Payees
```

ج) صفحه‌ی ثبت صورتحساب‌ها

صفحات ثبت گروه‌های اقلام، حساب‌ها و فروشنده‌ها، نکته‌ی خاصی ندارند. چون این جداول وابستگی خاصی به جایی نداشته و به سادگی اطلاعات آن‌ها را می‌توان ثبت یا به روز کرد.
صفحه‌ی مشکل در این مثال، همان صفحه‌ی ثبت صورتحساب‌ها است که از سه کلید خارجی به سه جدول دیگر تشکیل شده است.
عموما برای طراحی این نوع صفحات، کلیدهای خارجی را با drop down list نمایش می‌دهند و اگر در جهت سهولت کار کاربر قدم برداشته شود، باید از یک Auto complete drop down list استفاده کرد تا کاربر برنامه جهت یافتن آیتم‌های از پیش تعریف شده کمتر سختی بکشد.

اگر از Silverlight یا WPF استفاده شود، امکان بایند یک لیست کامل از اشیاء با تمام خواص مرتبط به آن‌ها وجود دارد (هر رکورد نمایش داده شده در دراپ داون لیست، دقیقا معادل است با یک شیء متناظر با کلاس‌های تعریف شده است). اگر از ASP.NET استفاده شود (یعنی یک محیط بدون حالت که پس از نمایش یک صفحه دیگر خبری از لیست اشیاء بایند شده وجود نخواهد داشت و همگی توسط وب سرور جهت صرفه جویی در منابع تخریب شده‌اند)، بهتر است datatextfield را با فیلد نام و datavaluefield را با فیلد Id مقدار دهی کرد تا کاربر نهایی، نام را جهت ثبت اطلاعات مشاهده کند و برنامه از Id موجود در لیست جهت ثبت کلیدهای خارجی استفاده نماید.
و نکته‌ی اصلی هم همینجا است که چگونه؟! چون ما زمانیکه با یک ORM سر و کار داریم، برای ثبت یک رکورد در جدول Bills باید یک وهله از کلاس Bill را ایجاد کرده و خواص آن‌را مقدار دهی کنیم. اگر به تعریف کلاس Bill مراجعه کنید، سه خاصیت آن از نوع سه کلاس مجزا تعریف شده است. به به عبارتی با داشتن فقط یک id از رکوردهای این کلاس‌ها باید بتوان سه وهله‌ی متناظر آن‌ها را از بانک اطلاعاتی خواند و سپس به این خواص انتساب داد:

```
var newBill = new Bill
                  {
                      Account = accountRepository.GetByKey(1),
                      Amount = 1,
                      BillDate = DateTime.Now,
                      Category = categoryRepository.GetByKey(1),
                      Description = "testestest...",
                      Payee = payeeRepository.GetByKey(1)
                  };
```

یعنی برای ثبت یک رکورد در جدول Bills فوق، چهار بار رفت و برگشت به دیتابیس خواهیم داشت:
- یکبار برای دریافت رکورد متناظر با گروه‌ها بر اساس کلید اصلی آن (که از دراپ داون لیست مربوطه دریافت می‌شود)
- یکبار برای دریافت رکورد متناظر با فروشنده‌ها بر اساس کلید اصلی آن (که از دراپ داون لیست مربوطه دریافت می‌شود)
- یکبار برای دریافت رکورد متناظر با حساب‌ها بر اساس کلید اصلی آن (که از دراپ داون لیست مربوطه دریافت می‌شود)
- یکبار هم ثبت نهایی اطلاعات در بانک اطلاعاتی

متد GetByKey فوق همان متد session.Get استاندارد NHibernate است (چون به primary key ها از طریق drop down list دسترسی داریم، به سادگی می‌توان بر اساس متد Get استاندارد ذکر شده عمل کرد).

SQL نهایی تولیدی هم به صورت واضحی این مشکل را نمایش می‌دهد (4 بار رفت و برگشت؛ سه بار select یکبار هم insert نهایی):

```
SELECT account0_.AccountId as AccountId0_0_, account0_.Name as Name0_0_
FROM Accounts account0_ WHERE account0_.AccountId=@p0;@p0 = 1 [Type: Int32 (0)]

SELECT category0_.CategoryId as CategoryId2_0_, category0_.Name as Name2_0_
FROM Categories category0_ WHERE category0_.CategoryId=@p0;@p0 = 1 [Type: Int32 (0)]

SELECT payee0_.PayeeId as PayeeId3_0_, payee0_.Name as Name3_0_
FROM Payees payee0_ WHERE payee0_.PayeeId=@p0;@p0 = 1 [Type: Int32 (0)]

INSERT INTO Bills (Amount, BillDate, Description, AccountId, CategoryId, PayeeId)
VALUES (@p0, @p1, @p2, @p3, @p4, @p5);
select SCOPE_IDENTITY();
@p0 = 1 [Type: Decimal (0)],
@p1 = 2010/12/27 11:48:33 ق.ظ [Type: DateTime (0)],
@p2 = 'testestest...' [Type: String (500)],
@p3 = 1 [Type: Int32 (0)],
@p4 = 1 [Type: Int32 (0)],
@p5 = 1 [Type: Int32 (0)]
```

کسانی که قبلا با رویه‌های ذخیره شده کار کرده باشند (stored procedures) احتمالا الان خواهند گفت؛ ما که گفتیم این روش کند است! سربار زیادی دارد! فقط کافی است یک SP بنویسید و کل عملیات را با یک رفت و برگشت انجام دهید.
اما در ORMs نیز برای انجام این مورد در طی یک حرکت یک ضرب راه حل‌هایی وجود دارد که در ادامه بحث خواهد شد:

د) پیاده سازی با NHibernate

برای حل این مشکل در NHibernate با داشتن primary key (برای مثال از طریق datavaluefield ذکر شده)، بجای session.Get از session. Load استفاده کنید.

session.Get یعنی همین الان برو به بانک اطلاعاتی مراجعه کن و رکورد متناظر با کلید اصلی ذکر شده را بازگشت بده و یک شیء از آن را ایجاد کن (حالت‌های دیگر دسترسی به اطلاعات مانند استفاده از LINQ یا Criteria API یا هر روش مشابه دیگری نیز در اینجا به همین معنا خواهد بود).

session.Load یعنی فعلا دست نگه دار! مگر در جدول نهایی نگاشت شده، اصلا چیزی به نام شیء مثلا گروه وجود دارد؟ مگر این مورد واقعا یک فیلد عددی در جدول Bills بیشتر نیست؟ ما هم که الان این عدد را داریم (به کمک عناصر دراپ داون لیست)، پس لطفا در پشت صحنه یک پروکسی برای ایجاد شیء مورد نظر ایجاد کن (uninitialized proxy to the entity) و سپس عملیات مرتبط را در حین تشکیل SQL نهایی بر اساس این عدد موجود انجام بده. یعنی نیازی به رفت و برگشت به بانک اطلاعاتی نیست. در این حالت اگر SQL نهایی را بررسی کنیم فقط یک سطر زیر خواهد بود (سه select ذکر شده حذف خواهند شد):

```
INSERT INTO Bills (Amount, BillDate, Description, AccountId, CategoryId, PayeeId)
VALUES (@p0, @p1, @p2, @p3, @p4, @p5);
select SCOPE_IDENTITY();
@p0 = 1 [Type: Decimal (0)],
@p1 = 2010/12/27 11:58:22 ظ.ق [Type: DateTime (0)],
@p2 = 'testestest...' [Type: String (500)],
@p3 = 1 [Type: Int32 (0)],
@p4 = 1 [Type: Int32 (0)],
@p5 = 1 [Type: Int32 (0)]
```

ه) پیاده سازی با Entity framework

Entity framework زمانیکه بانک اطلاعاتی فوق را (به روش database first) به کلاس‌های متناظر تبدیل/نگاشت می‌کند، حاصل نهایی مثلا در مورد کلاس Bill به صورت خلاصه به شکل زیر خواهد بود:

```
public partial class Bill : EntityObject
{
	public global::System.Int32 BillId {set;get;}
	public global::System.Decimal Amount {set;get;}
	public global::System.DateTime BillDate {set;get;}
	public global::System.String Description {set;get;}
	public global::System.Int32 AccountId {set;get;}
	public global::System.Int32 CategoryId {set;get;}
	public global::System.Int32 PayeeId {set;get;}
	public Account Account {set;get;}
	public Category Category {set;get;}
}
```

به عبارتی فیلدهای کلیدهای خارجی، در تعریف نهایی این کلاس هم مشاهده می‌شوند. در اینجا فقط کافی است سه کلید خارجی، از نوع int مقدار دهی شوند (و نیازی به مقدار دهی سه شیء متناظر نیست). در این حالت نیز برای ثبت اطلاعات، فقط یکبار رفت و برگشت به بانک اطلاعاتی خواهیم داشت.

نظرات خوانندگان

نویسنده: Afshar Mohebbi
تاریخ: ۱۳۸۹/۱۰/۰۶ ۲۳:۲۰:۲۴

سلام،

من هم این روزها خیلی درگیر این مسئله با NH هستم. تا حالا دو تا راه پیدا کردم. یکی استفاده از HQL برای update کردن و دیگری استفاده از خاصیت Future برای کاهش رفت و آمدها به دیتابیس. البته تا حالا از هیچ کدام اونها به طور عملی استفاده نکنم. ولی با این راه حلی که شما گفتید تعداد راه حل‌ها سه تا می‌شود.

فقط مسئله کوچکی که می‌ماند این است که من عمدتاً از Castle ActiveRecord و Linq-to-NH استفاده می‌کنم و نمی‌دانم با نبود Load (به جای Get) در این حالت چه کار کنم.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۰/۰۷ ۰۰:۳۴:۴۰

سلام، به نظر مطابق مستندات آن (+) اگر SessionScope تعریف شود و داخل آن کار کنید، متد Find شبیه به همان Load ذکر شده در مطلب فوق عمل می‌کند. تست کنید ببینید در این حالت تعداد کوئری‌ها چه فرقی می‌کند.

نویسنده: iMAN
تاریخ: ۱۳۸۹/۱۰/۰۸ ۱۱:۴۸:۳۹

برای Castle ActiveRecord استفاده از SessionScope همونطور که اشاره کردین موضوع را حل می کند.