عنوان: **فعال سازی سطح دوم کش در Fluent NHibernate**
نویسنده: وحید نصیری
تاریخ: ۰۹:۱۸:۰۰ ۱۳۹۰/۰۲/۱۱
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

سطح اول کش در NHibernate در یک تراکنش معنا پیدا می‌کند (+)؛ اما نتایج حاصل از اعمال سطح دوم (+) آن، در اختیار تمام تراکنش‌های جاری برنامه خواهند بود. در ادامه قصد داریم نحوه فعال سازی سطح دوم کش NHibernate را توسط Fluent NHibernate بررسی کنیم.

الف) دریافت کش پروایدر
برای این منظور به صفحه اصلی آن در سایت سورس فورج مراجعه نمائید(+). اگر به علت تحریم‌ها امکان دریافت فایل‌های مرتبط را نداشتید از این برنامه استفاده کنید(+). پس از دریافت، می‌خواهیم نحوه فعال سازی
NHibernate.Caches.SysCache.dll را بررسی کنیم (این اسمبلی، در برنامه‌های وب و دسکتاپ بدون مشکل کار می‌کند).

ب) اعمال به قسمت تعاریف اولیه
پس از دریافت اسمبلی NHibernate.Caches.SysCache.dll و افزودن ارجاعی به آن، اکنون نوبت به معرفی آن به تنظیمات Fluent NHibernate می‌باشد. این‌کار هم بسیار ساده است:

```
...
     .ConnectionString(x => x.FromConnectionStringWithKey(...))
     .Cache(x => x.UseQueryCache()
                          .UseMinimalPuts()
                          .ProviderClass<NHibernate.Caches.SysCache.SysCacheProvider>())
...
```

ج) تعریف نوع کش در هنگام ایجاد نگاشت‌ها
اگر از ClassMapها برای تعریف نگاشت‌ها استفاده می‌کنید، در انتهای تعاریف یک سطر Cache.ReadWrite را اضافه کنید.
اگر از AutoMapping استفاده می‌کنید، نیاز است تا با استفاده از IAutoMappingOverride (+) سطر یاد شده اضافه گردد؛ برای مثال:

```
using FluentNHibernate.Automapping.Alterations;

namespace NH3Test.MappingDefinitions.Domain
{
   public class AccountOverrides : IAutoMappingOverride<Account>
   {
       public void Override(FluentNHibernate.Automapping.AutoMapping<Account> mapping)
       {
           mapping.Cache.ReadWrite();
       }
   }
}
```

تعریف یک سطر فوق هم مهم است؛ زیرا در غیراینصورت فقط primary key حاصل از بار اول فراخوانی کوئری‌های مرتبط کش می‌شوند؛ نه نتیجه عملیات. هرچند این مورد هم یک قدم مثبت به شمار می‌رود از این لحاظ که برای مثال تهیه نتایج کوئری بر روی فیلدی که ایندکس بر روی آن تعریف نشده است همیشه از حالت تهیه کوئری بر روی فیلد دارای ایندکس کندتر است. اما هدف ما در اینجا این است که پس از بار اول فراخوانی کوئری، بارهای دوم و بعدی دیگر کوئری خاصی را به بانک اطلاعاتی ارسال نکرده و نتایج از کش خوانده شوند (جهت استفاده عموم کاربران در کلیه تراکنش‌های جاری برنامه).

د) اعمال متد Cacheable به کوئری‌ها
سه مرحله قبل نحوه برپایی مقدماتی سطح دوم کش را بیان می‌کنند و تنها یکبار نیاز است انجام شوند. در ادامه هر جایی که نیاز

داشتیم نتایج کوئری مورد نظر کش شوند (و باید دقت داشت که این کش شدن سطح دوم به معنی در دسترس بودن نتایج آن جهت تمام کاربران برنامه در تمام تراکنش‌های جاری برنامه هستند؛ برای مثال نتایج آمار سایت که دسترسی عمومی دارد) تنها کافی است متد Cacheable را به کوئری مورد نظر اضافه کرد؛ برای مثال:

```
var data = session.QueryOver<Account>()
                .Where(s => s.Name == "name")
                .Cacheable()
                .List();
```

ه) چگونه صحت اعمال سطح دوم کش را بررسی کنیم؟

برای بررسی این امر باید به خروجی SQL نهایی مراجعه کرد (+). سه تراکنش مجزا را تعریف کنید. در تراکنش اول یک insert ساده، در تراکنش دوم کوئری از اطلاعات قبل (به همراه اعمال متد Cacheable) و در تراکنش سوم مجددا همان کوئری تراکنش دوم را (به همراه اعمال متد Cacheable) تکرار کنید. حاصل کار تنها باید دو عبارت SQL باشند. یک مورد جهت insert و یک مورد هم select . در تراکنش سوم، از نتایج کش شده تراکنش دوم استفاده خواهد شد؛ به همین جهت دیگری کوئری سومی به بانک اطلاعاتی ارسال نخواهد شد.

اگر اعمال مورد (ج) فوق را فراموش کنید، سه کوئری را مشاهده خواهید کرد، اما کوئری سوم با کوئری دوم اندکی متفاوت خواهد بود و بهینه‌تر؛ چون به صورت هوشمند بر اساس جستجوی بر روی primary key تغییر کرده است (صرفنظر از اینکه قسمت where کوئری شما چیست).

نظرات خوانندگان

نویسنده: Ahmadxml
تاریخ: ۱۳۹۰/۰۲/۱۱ ۱۷:۳۸:۳۰

سلام آقای نصیری

```
var q1 = session.QueryOver().Where(x => x.Id > 3).Cacheable().List();
var q2 = session.QueryOver().Where(x => x.Id == 4).List();
```

چرا با اینکه نتیجه کوئری دوم در کوئری اول وجود داره اما باز به دیتابیس مراجعه می کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۱۱ ۲۰:۲۸:۵۱

سلام،

کلا سطح دوم کش در NH بر اساس 4 مکانیزم در طول یک سشن فکتوری عمل می‌کند:
- کش مربوط به موجودیت‌ها (entities cache) که بر اساس متد session.Get یا Load فعال می‌شود
و همچنین Collections cache (متدهای List و Enumerable)
- کش مربوط به کوئری‌ها (queries cache) با اعمال متد Cacheable به کوئری مورد نظر.
- timestamp cache که به معنای آخرین زمان نوشتن در یک جدول می‌باشد (و این timestamp فقط و فقط بر اساس وجود تراکنش‌ها عمل می‌کند). به این ترتیب در زمان insert/update/delete به صورت خودکار کش موجود منقضی اعلام می‌شود تا اطلاعات قدیمی به کاربر تحویل داده نشود و کش سطح دوم جهت کوئری‌های بعدی بازسازی خواهد شد.

و در مثال شما:
- در کوئری دوم هم باید متد Cacheable ذکر شود اگر نشود به صورت متداول با آن برخورد خواهد شد.
- زمانیکه از متد Cacheable استفاده می‌شود، حالت queries cache فعال می‌شود. چون در مثال شما دو کوئری مختلف داریم، پس به کش مربوط به کوئری اول مراجعه نخواهد شد. این کوئری کش، با تغییر مقادیر پارامترهای یک کوئری هم مجددا به روز می‌شود. (این حالت برای کوئری‌هایی که با پارامترهای یکسان به طور متناوب فراخوانی می‌شوند، بسیار مناسب است)
- سطح دوم کش فقط پس از commit یک تراکنش معنا پیدا می‌کند. بنابراین اگر جهت آزمایش داخل یک تراکنش، پشت سر هم کوئری‌ها را نوشته‌اید ... در این لحظه از سطح دوم کش بی‌بهره خواهید بود (فقط سطح اول کش فعال است) و کوئری‌های پس از پایان تراکنش جاری، از نتیجه کش آن می‌توانند استفاده کنند.
- در مورد کش مربوط به موجودیت‌ها و تفاوت آن با کش کوئری‌ها در بالا صحبت شد (شما در یک جا کش کوئری را فعال کرده‌اید در جای دیگر کش entities را طلب می‌کنید که نمی‌شود).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۱۳ ۱۶:۲۸:۴۳

آقای پایروند این مجموعه رو تبدیل به فایل پی دی اف کردند برای کسانیکه می‌خواهند ساده‌تر آن‌را مطالعه یا حتی پرینت بگیرند
https://rapidshare.com/files/460383624/NHibernate_VN_.pdf