عنوان: NHibernate و سطح اول cache آن
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۰۱ ۱۸:۲۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: NHibernate

این روزها هیچکدام از فناوری‌های دسترسی به داده بدون امکان یکپارچگی آن‌ها با سیستم‌ها و روش‌های متفاوت caching ، مطلوب شمرده نمی‌شوند. ایده اصلی caching هم به زبان ساده به این صورت است : فراهم آوردن روش‌هایی جهت میسر ساختن دسترسی سریعتر به داده‌هایی که به صورت متناوب در برنامه مورد استفاده قرار می‌گیرند، بجای مراجعه مستقیم به بانک اطلاعاتی و خواندن اطلاعات از دیسک سخت.
یکی از تفاوت‌های مهم NHibernate با اکثر ORM های موجود داشتن دو سطح متفاوت cache است : first level cache & second level cache .
برای نمونه Entity framework (در زمان نگارش این مطلب) تنها first level caching را پشتیبانی می‌کند و پروایدر توکار و یکپارچه‌ای را جهت second level caching ارائه نمی‌دهد.
در این قسمت قصد داریم First Level Cache را بررسی کنیم.

سطح اول caching در NHibernate چیست؟

سطح اول caching در تمام ORM هایی که آن‌را پشتیبانی می‌کنند مانند NHibernate ، در طول عمر یک تراکنش تعریف می‌گردد. در این حالت در طی یک تراکنش و طول عمر یک سشن، دریافت اطلاعات هر رکورد از بانک اطلاعاتی، تنها یکبار انجام خواهد شد؛ صرفنظر از اینکه کوئری دریافت اطلاعات آن چندبار فراخوانی می‌گردد. یکی از دلایل این روش هم آن است که هیچ دو شیء متفاوتی که هم اکنون در حافظه قرار دارند نباید بیانگر یک رکورد واحد از بانک اطلاعاتی باشند.
در NHibernate به صورت پیش فرض هر زمانیکه از شیء استاندارد session استفاده می‌کنید، سطح اول caching نیز فعال است. درست در زمانیکه سشن خاتمه می‌یابد، این سطح از caching نیز به صورت خودکار تخلیه خواهد گردید.
به first level caching اصطلاحا thought-out cache system یا Cache Through pattern و یا identity map هم گفته می‌شود.

مثال:

روش متداول و استاندارد کار با NHibernate عموما به صورت زیر است:

الف) دریافت شیء Session از Session Factory
ب) شروع یک تراکنش با فراخوانی متد BeginTransaction شیء Session
ج) برای مثال دریافت اطلاعات رکوردی با ID مساوی یک به کمک متد Get مرتبط با شیء Session : این اطلاعات مستقیما از بانک اطلاعاتی دریافت خواهد شد.
د) سپس مجددا سعی در دریافت رکوردی با ID مساوی یک. اینبار اطلاعات این شیء مستقیما از cache خوانده می‌شود و رفت و برگشتی به بانک اطلاعاتی نخواهیم داشت. به همین جهت به این روش identity map هم گفته می‌شود، زیرا NHibernate بر اساس ID منحصربفرد این اشیاء ، identity map خود را تشکیل می‌دهد.
ه) خاتمه‌ی سشن با فراخوانی متد Close آن
بلافاصله
الف) دریافت شیء Session از Session Factory
ب) شروع یک تراکنش با فراخوانی متد BeginTransaction شیء Session
ج) برای مثال دریافت اطلاعات رکوردی با ID مساوی یک به کمک متد Get مرتبط با شیء Session : این اطلاعات مستقیما از بانک اطلاعاتی دریافت خواهد شد (زیرا در یک سشن جدید قرار داریم و همچنین سشن قبلی بسته شده و کش آن تخلیه گشته است).
د) خاتمه‌ی سشن با فراخوانی متد Close آن

سؤال: آیا استفاده از یک سشن سراسری در برنامه صحیح است؟
پاسخ: خیر!
توضیحات: زمانیکه از یک سشن سراسری استفاده می‌کنید، کش NHibernate را در اختیار تمام کاربران همزمان سیستم قرار داده‌اید. در طی یک سشن، همانطور که عنوان شد، بر اساس IDهای اشیاء، یک identity map تشکیل می‌شود و در این حالت به ازای هر رکورد بانک اطلاعاتی فقط و فقط یک شیء در حافظه وجود خواهد داشت که این روش در محیط‌های چندکاربره مانند برنامه‌های وب به زودی تبدیل به نشت اطلاعات و یا تخریب اطلاعات می‌گردد. به همین جهت در این نوع برنامه‌ها روش session-per-request بهترین حالت کاری است.

سؤال: حین به روز رسانی اشیاء جدید، به خطا بر می‌خورم. مشکل در کجاست؟
فرض کنید شیء مفروض Customer را توسط متد session.Get از بانک اطلاعاتی دریافت و تعدادی از خواص آن‌را جهت ساخت شیء جدیدی از کلاس Customer استفاده کرده‌ایم. اکنون اگر بخواهیم این شیء جدید را در بانک اطلاعاتی ذخیره یا به روز رسانی کنیم، NHibernate این اجازه را نمی‌دهد! چرا؟
پاسخ:
خطای متداول این حالت عموما به صورت زیر است:

```
a different object with the same identifier value was already associated with the session
```

اگر شخصی با مکانیزم سطح اول caching در NHibernate آشنایی نداشته باشد، شاید ساعاتی را در انجمن‌های مرتبط، جهت یافتن روش حل خطای فوق سپری کند.
همانطور که عنوان شد، در طول یک سشن، نمی‌توان دو شیء با یک ID را به عنوان یک رکورد بانک اطلاعاتی مورد استفاده قرار داد. اولین فراخوانی Get ، سبب کش شدن آن شیء در identity map سطح اول caching می‌گردد.
راه حل:
الف) از چندین و چند شیء استفاده نکنید. هر رکورد باید تنها با یک وهله از شیءایی متناظر باشد.
ب) می‌توان پیش از update، کش سطح اول را به صورت دستی خالی کرد. برای این منظور از متد Clear شیء سشن استفاده کنید.
ج) بجای استفاده از متد saveOrUpdate شیء سشن، از متد Merge آن استفاده کنید. به این صورت شیء جدید ایجاد شده با شیء موجود در کش یکی خواهد شد.
د) می‌توان بجای تخلیه کل کش (حالت ب)، کش مرتبط با شیء Customer را به صورت دستی خالی کرد. برای این منظور از متد Evict شیء سشن استفاده نمائید.

و لازم به ذکر است که متد Flush سبب تخلیه کش نمی‌گردد. کار این متد اعمال کلیه تغییرات اعمالی موجود در کش به بانک اطلاعاتی است و بیشتر جهت هماهنگ سازی این دو مورد استفاده قرار می‌گیرد.

سؤال: آیا می‌توان سطح اول caching را غیرفعال کرد؟
پاسخ:بله.
توضیحات:
عموما کلیه ORMs جهت Batching یا Bulk data operations (برای مثال ثبت تعداد زیادی رکورد یا به روز رسانی تعداد بالایی از آن‌ها، یا نمایش فقط خواندنی تعداد زیادی رکورد و گزارشگیری از آن‌ها) کارآیی مطلوبی ندارند. نمونه‌ای از آن‌را در مبحث جاری ملاحظه کرده‌اید. هر شیءایی که به نحوی به سشن جاری وارد می‌شود تحت نظر قرار می‌گیرد و این مورد در تعداد بالای ثبت یا به روز رسانی رکوردها، یعنی کاهش سرعت و کارآیی، به علاوه مصرف بالای حافظه. به همین جهت باید به خاطر داشت که ORMs جهت سناریوهای OLTP مناسب هستند و کسانی که سرعت و کارآیی ORMs را با Batch processing اندازه گیری می‌کنند، کلا درکی از فلسفه‌ی وجودی ORMs و ساختار درونی آن‌ها ندارند!
خوشبختانه NHibernate با معرفی Stateless Sessions بر این مشکل فائق آمده است. در اینجا بجای ISession تنها کافی است از IStatelessSession استفاده گردد:

```
using (IStatelessSession statelessSession = sessionFactory.OpenStatelessSession())
using (ITransaction transaction = statelessSession.BeginTransaction())
```

```
{
 //now insert 1,000,000 records!
}
```

در این حالت سیستم دو مزیت عمده را تجربه خواهد کرد: سرعت بالای ثبت اطلاعات با تعداد زیاد رکورد و همچنین مصرف پایین حافظه از آنجائیکه یک IStatelessSession ارجاعی را به اشیایی که بارگذاری می‌کند، در خود نگهداری نخواهد کرد.
تنها باید به خاطر داشت که در این حالت lazy loading پشتیبانی نمی‌شود و همچنین رخدادهای درونی NHibernate نیز لغو خواهند شد.

نظرات خوانندگان

نویسنده: Afshar Mohebbi
تاریخ: ۱۳۸۹/۱۱/۱۷ ۰۹:۱۸:۴۳

سپاس