عنوان: **سطح دوم کش در Nhibernate 4**
نویسنده: احسان زینلی
تاریخ: ۱۳۹۳/۰۷/۰۸ ۹:۴۵
آدرس: www.dotnettips.info
گروه‌ها: NHibernate, Caching, NHibernate 4

Second Level Cache In NHibernate 4

همان طور که می‌دانیم کش در NHibernate در دو سطح قابل انجام می‌باشد:
- کش سطح اول که همان اطلاعات سشن، در تراکنش جاری هست و با اتمام تراکنش، محتویات آن خالی می‌گردد. این سطح همیشه فعال می‌باشد و در این بخش قصد پرداختن به آن را نداریم.
- کش سطح دوم که بین همه‌ی تراکنش‌ها مشترک و پایدار می‌باشد. این مورد به طور پیش فرض فعال نمی‌باشد و می‌بایستی از طریق کانفیگ برنامه فعال گردد.

جهت پیاده سازی باید قسمت‌های ذیل را در کانفیگ مربوط به NHibernate اضافه نمود:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<configSections>
        <section name="hibernate-configuration" type="NHibernate.Cfg.ConfigurationSectionHandler,
NHibernate"/>
         <section name="syscache" type="NHibernate.Caches.SysCache.SysCacheSectionHandler,
NHibernate.Caches.SysCache" requirePermission="false"/>
</configSections>

<hibernate-configuration  xmlns="urn:nhibernate-configuration-2.2" >
<session-factory>
       <property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>
     <property name="dialect">NHibernate.Dialect.MsSql2005Dialect</property>
     <property name="connection.driver_class">NHibernate.Driver.SqlClientDriver</property>
     <property name="connection.connection_string_name">LocalSqlServer</property>
     <property name="show_sql">false</property>
     <property name="hbm2ddl.keywords">none</property>
     <property name="cache.use_second_level_cache">true</property>
     <property name="cache.use_query_cache" >true</property>
     <property name="cache.provider_class">NHibernate.Caches.SysCache.SysCacheProvider,
NHibernate.Caches.SysCache</property>
</session-factory>
</hibernate-configuration>

<syscache>
      <cache region="LongExpire" expiration="3600" priority="5"/>
      <cache region="ShortExpire" expiration="600" priority="3"/>
</syscache>
</configuration>
```

پیاده سازی Caching در NHibernate در سه مرحله قابل اعمال می‌باشد :
- کش در سطح Load موجودیت‌های مستقل
- کش در سطح Load موجودیت‌های وابسته Set , List , Bag , ...
- کش در سطح Query ها

Providerهای مختلفی برای اعمال و پیاده سازی آن وجود دارند که معروف‌ترین آن‌ها SysCache بوده و ما هم از همان استفاده می‌نماییم.
- مدت زمان پیش فرض کش سطح دوم، ۵ دقیقه می‌باشد و در صورت نیاز به تغییر آن، باید تگ مربوط به SysCache را تنظیم نمود. محدودیتی در تعریف تعداد متفاوتی از زمان‌های خالی شدن کش وجود ندارد و مدت زمان آن بر حسب ثانیه مشخص می‌گردد. نحوه‌ی تخصیص زمان انقضای کش به هر مورد بدین شکل صورت می‌گیرد که region مربوطه در آن معرفی می‌گردد.

جهت اعمال کش در سطح Load موجودیت‌های مستقل، علاوه بر کانفیگ اصلی، می‌بایستی کدهای زیر را به Mapping موجودیت اضافه نمود مانند :

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="Core.Domain"
namespace="Core.Domain.Model">
  <class name="Organization" table="Core_Enterprise_Organization">
    <cache usage="nonstrict-read-write" region="ShortExpire"/>
    <id name="Id" >
      <generator/>
    </id>
    <version name="Version"/>
    <property name="Title" not-null="true" unique="true"/>
    <property name="Code" not-null="true" unique="true"/>
  </class>
</hibernate-mapping>
```

این مورد برای موجودیت‌های وابسته هم نیز صادق است؛ به شکل کد زیر:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="Core.Domain"
namespace="Core.Domain.Model">
  <class name="Party" table="Core_Enterprise_Party">
    <id name="Id" >
      <generator />
    </id>
    <version name="Version"/>
    <property name="Username" unique="true"/>
    <property name="DisplayName" not-null="true"/>
    <bag name="PartyGroups" inverse="true" table="Core_Enterprise_PartyGroup" cascade="all-delete-
orphan">
      <cache usage="nonstrict-read-write" region="ShortExpire"/>
      <key column="Party_id_fk"/>
      <one-to-many/>
    </bag>
  </class>
</hibernate-mapping>
```

ویژگی usage نیز با مقادیر زیر قابل تنظیم است:

- read-only : این مورد جهت موجودیت‌هایی مناسب است که امکان بروزرسانی آن‌ها توسط کاربر وجود ندارد. این مورد بهترین کارآیی را دارد.
- read-write : این مورد جهت موجودیت‌هایی بکار می‌رود که امکان بروزرسانی آن‌ها توسط کاربر وجود دارد. این مورد کارآیی پایین‌تری دارد.
- nonstrict-read-write : این مورد جهت موجودیت‌هایی مناسب می‌باشد که امکان بروزرسانی آن‌ها توسط کاربر وجود دارد؛ اما امکان همزمان بروز کردن آن‌ها توسط چندین کاربر وجود نداشته باشد. این مورد در قیاس، کارآیی بهتر و بهینه‌تری نسبت به مورد قبل دارد.

جهت اعمال کش در کوئری‌ها نیز باید مراحل خاص خودش را انجام داد. به عنوان مثال برای یک کوئری Linq به شکل زیر خواهیم داشت:

```
public IList<Organization> Search(QueryOrganizationDto dto)
{
     var q = SessionInstance.Query<Organization>();

     if (!String.IsNullOrEmpty(dto.Title))
          q = q.Where(x => x.Title.Contains(dto.Title));

     if (!String.IsNullOrEmpty(dto.Code))
          q = q.Where(x => x.Code.Contains(dto.Code));

     q = q.OrderBy(x => x.Title);

      q = q.CacheRegion("ShortExpire").Cacheable();
     return q.ToList();
}
```

در واقع کد اضافه شده به کوئری بالا، قابل کش بودن کوئری را مشخص می‌نماید و مدت زمان کش شدن آن نیز از طریق کانفیگ مربوطه مشخص می‌گردد. این نکته را هم درنظر داشته باشید که کش در سطح کوئری برای کوئری‌هایی که دقیقا مثل هم هستند اعمال می‌گردد و با افزوده یا کاسته شدن یک شرط جدید به کوئری، مجددا کوئری سمت پایگاه داده ارسال می‌گردد.

در انتها لینک‌های زیر هم جهت مطالعه بیشتر پیشنهاد می‌گردند:

http://www.nhforge.org/doc/nh/en/index.html#performance-cache-readonly

http://nhforge.org/blogs/nhibernate/archive/2009/02/09/quickly-setting-up-and-using-nhibernate-s-second-level-cache.aspx

http://www.klopfenstein.net/lorenz.aspx/using-syscache-as-secondary-cache-in-nhibernate

http://stackoverflow.com/questions/1837651/hibernate-cache-strategy