## Hibernate Caching

Caching is a mechanism for storing the loaded objects into a cache memory.

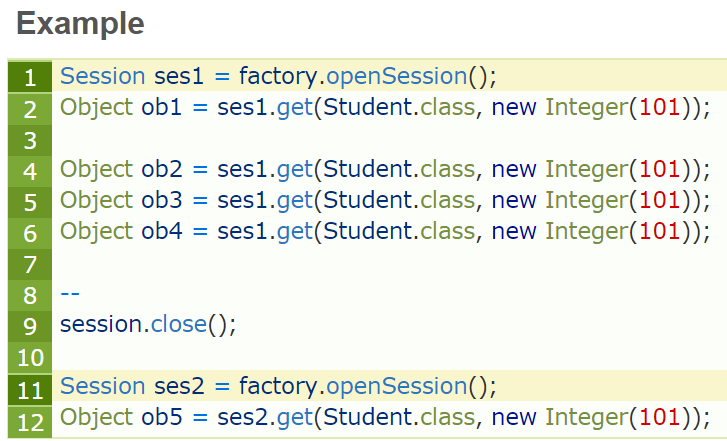
The advantage of cache mechanism is, ***whenever again we want to load the same object from the database then instead of hitting the database once again, it loads from the local cache memory only, so that the no. of round trips between an application and a database server got decreased.***  It means caching mechanism increases the performance of the application.

In hibernate we have two levels of caching

1. **First Level Cache [ or ] Session Cache**
2. **Second Level Cache [ or ] Session Factory Cache [ or  ] JVM Level Cache**

First Level Cache

* By default, for each hibernate application, the first level cache is automatically been enabled
* As a programmer, we no need to have any settings to enable the first level cache and also we cannot disable this first level cache
* the first level cache is associated with the session object and scope of the cache is limited to one session only
* When we load an object for the first time from the database then the object will be loaded from the database and the loaded object will be stored in the cache memory maintained by that session object
* If we load the same object once again, with in the same session, then the object will be loaded from the local cache memory not from the database
* If we load the same object by opening other session then again the object will loads from the database and the loaded object will be stored in the cache memory maintained by this new session



**Explanation:**

* In line number1, i have opened one session with object is ses1
* In line number2, loaded one object with id 101, now it will loads the object from the database only as it’s the first time, and keeps this object in the session cache
* **See at line number 4,5,6 i tried to load the same object 3 times, but here the object will be loaded from the stored cache only not from the database, as we are in the same session.**
* In line number 9, we close the first session, so the cache memory related this session also will be destroyed
* See line number 11, again i created one new session and loaded the same object with id 101 in line number 12, but this time hibernate will loads the object from the database

**Number of sessions = that many Number of cache memories**

The loaded objects will be stored in cache memory maintained by a session object and if we want to remove the objects that are stored in the cache memory, then we need to call either evict() or clear()methods.

Actually **evict()** is used to **remove a particular** object from the cache memory and **clear()** is to **remove all** objects in the cache memory.

# Second Level Caching

* Second level cache was introduced in hibernate 3.0
* Whenever we are loading any object from the database,  then hibernate verify whether that object is available in the local cache memory of that particular session [**means first level cache** ], if not available then hibernate verify whether the object is available in global cache or factory cache [ **second level cache** ], if not available then hibernate will hit the database and loads the object from there, and then first stores in the local cache of the session [ first level ] then in the global cache [ second level cache ]
* When another session need to load the same object from the database,  then hibernate copies that object from global cache [ second level cache ] into the local cache of this new session

**Second level cache in the hibernate is of  from  vendors…**

* Easy Hibernate [EHCache] Cache from hibernate framework
* Open Symphony [OS] cache from Open Symphony
* TreeCache from JBoss

How to enable second level cache in hibernate

To enable second level cache in the hibernate, then the following **3** changes are required

1. Add provider class in hibernate configuration

**<property name="hibernate.cache.provider\_class">**

**org.hibernate.cache.EhCacheProvider**

**</property>**

**2.** Configure cache element for a class in hibernate mapping file

**<cache usage="read-only" />**

**3.** Create xml file called ehcache.xml

**ehcache.xml**

**<ehcache>**

**<defaultCache maxElementsInMemory="100"**

**eternal="false"**

**timeToIdleSeconds="120"**

**timeToLiveSeconds="200" />**

**<cache name="str.Product"**

**maxElementsInMemory="100"**

**eternal="false"**

**timeToIdleSeconds="5"**

**timeToLiveSeconds="200" />**

**</ehcache>**

* In ehcache.xml, if eternal=”true” then we should not write timeToIdealSeconds, timeToLiveSeconds,  hibernate will take care about those values
* So if you want to give values manually better eternal=”false” always,  so that we can assign values into timeToIdealSeconds, timeToLiveSeconds manually, and play.

**timeToIdleSeconds** It defines that **how many seconds object can be idle in the second level** cache.

**timeToLiveSeconds** It defines that **how many seconds object can be stored in the second level cache whether it is idle or not.**

* Actually <defaultCache … /> will reflects to all the pojo classes in our application,  and we can also assign the ehcache values to specified pojo class by <cache name=”– your pojo class name —” …….. />