Hibernate Second Level Cache with Spring an Ehcache

# Maven Configuration(pom.xml)

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>hibernate-second-level-cache</groupId>

<artifactId>hibernate-second-level-cache</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-hibernate</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

<!-- For Second Level Cache, Ehcache Configuration -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-ehcache</artifactId>

<version>4.3.4.Final</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.10</version>

</dependency>

</dependencies>

</project>

# Spring-Hibernate Configuration (app-context.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />

<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />

<property name=*"user"* value=*"deba"* />

<property name=*"password"* value=*"deba"* />

<!-- these are C3P0 properties -->

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"initialPoolSize"* value=*"5"* />

<property name=*"minPoolSize"* value=*"5"* />

<property name=*"maxPoolSize"* value=*"20"* />

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>

<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>

</bean>

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"* />

<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true

</value> </property> -->

<property name=*"hibernateProperties"*>

<value>

**hibernate.format\_sql=true**

**hibernate.show\_sql=true**

**hibernate.cache.use\_second\_level\_cache=true**

**hibernate.cache.use\_query\_cache=true**

**hibernate.generate\_statistics=true**

**org.hibernate.cache.ehcache.configurationResourceName=/ehcache.xml**

**hibernate.cache.region.factory\_class=org.hibernate.cache.ehcache.EhCacheRegionFactory**

</value>

<!-- <value>

hibernate.id.new\_generator\_mappings=true

hibernate.show\_sql=true

hibernate.show\_sql=true

hibernate.cache.use\_second\_level\_cache=true

hibernate.cache.use\_query\_cache=true

hibernate.generate\_statistics=true

hibernate.cache.provider\_configuration\_file\_resource\_path=classpath:ehcache.xml

net.sf.ehcache.configurationResourceName=/ehcache.xml

net.sf.ehcache.configurationResourceName=/ehcache.xml

hibernate.cache.provider\_configuration\_file\_resource\_path=classpath:ehcache.xml

hibernate.cache.ehcache.configurationResourceName=/ehcache.xml

org.hibernate.cache.ehcache.configurationResourceName=/ehcache.xml

hibernate.cache.region.factory\_class=net.sf.ehcache.hibernate.EhCacheRegionFactory

org.hibernate.cache.ehcache.configurationResourceName=/ehcache.xml

hibernate.cache.region.factory\_class=org.hibernate.cache.ehcache.EhCacheRegionFactory

hibernate.cache.region.factory\_class=org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory

hibernate.generate\_statistics=true

</value> -->

</property>

<property name=*"annotatedClasses"*>

<list>

<value>com.ddlab.rnd.orm.Employee</value>

</list>

</property>

</bean>

</beans>

# Ehcache Configuration(ehcache.xml)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<ehcache>

<diskStore path=*"java.io.tmpdir"* />

<cache name=*"org.hibernate.cache.internal.StandardQueryCache"*

maxElementsInMemory=*"20"* eternal=*"false"* timeToLiveSeconds=*"120"*

overflowToDisk=*"true"* />

<cache name=*"org.hibernate.cache.spi.UpdateTimestampsCache"*

maxElementsInMemory=*"5000"* eternal=*"true"* overflowToDisk=*"true"* />

<defaultCache maxElementsInMemory=*"100"* eternal=*"false"*

timeToIdleSeconds=*"10000"* timeToLiveSeconds=*"60000"* overflowToDisk=*"false"* />

<cache name=*"myemployee"* maxElementsInMemory=*"500"* eternal=*"true"* statistics=*"true"*

timeToIdleSeconds=*"0"* timeToLiveSeconds=*"0"* overflowToDisk=*"false"* />

</ehcache>

# Java Source Code

## Employee.java

**package** com.ddlab.rnd.orm;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** org.hibernate.annotations.Cache;

**import** org.hibernate.annotations.CacheConcurrencyStrategy;

@Entity

@Table(name = "selfjoinemp")

//@Cache(usage=CacheConcurrencyStrategy.READ\_WRITE)

@Cache(usage=CacheConcurrencyStrategy.***READ\_WRITE***, include = "all")

//@Cache(usage = CacheConcurrencyStrategy.READ\_WRITE,region="myemployee")

**public** **class** Employee {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name = "empId")

**private** **long** empId;

@Column(name = "firstname")

**private** String firstName;

@Column(name = "city")

**private** String city;

**public** Employee() {

}

**public** Employee(String firstName, String city) {

**this**.firstName = firstName;

**this**.city = city;

}

**public** **long** getEmpId() {

**return** empId;

}

**public** **void** setEmpId(**long** empId) {

**this**.empId = empId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

}

## App.java

**package** com.ddlab.rnd.orm;

**import** java.util.List;

**import** net.sf.ehcache.CacheManager;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Query;

**import** org.hibernate.ScrollableResults;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.stat.SecondLevelCacheStatistics;

**import** org.hibernate.stat.Statistics;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App

{

**public** **static** **void** bulkInsert(Session session) {

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

**for** ( **int** i=0; i<3000; i++ ) {

Employee employee = **new** Employee("Name-"+i, "City-"+i);

session.save(employee);

//Otherwise it will throw OutOfMemory Error

**if**( i % 1000 == 0 ) {

//flush a batch of inserts and release memory:

session.flush();

session.clear();

}

}

transaction.commit();

} **catch** (HibernateException e) {

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** bulkUpdate(Session session) {

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

ScrollableResults empResults = session.createQuery("FROM Employee").scroll();

**int** count = 0;

**while** ( empResults.next() ) {

Employee employee = (Employee) empResults.get(0);

employee.setCity("My City-"+count);

count++;

session.update(employee);

//Otherwise it will throw OutOfMemory Error

**if**( count % 1000 == 0 ) {

//flush a batch of inserts and release memory:

session.flush();

session.clear();

}

}

transaction.commit();

} **catch** (HibernateException e) {

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** showQueryResult(Session session) {

Query query = session.createQuery("from Employee e where e.empId > :empId").setCacheable(**true**);

query.setCacheRegion("myemployee");

query.setInteger("empId", 2000);

List<Employee> empList = (List<Employee>) query.list();

// query.uniqueResult();

System.***out***.println("Employee List Size :::"+empList.size());

**for**(Employee emp : empList) {

System.***out***.println(emp.getEmpId()+"---"+emp.getFirstName()+"---"+emp.getCity());

}

}

**public** **static** **void** printStatistics(SessionFactory sessionFactory) {

Statistics stat = sessionFactory.getStatistics();

// sessionFactory.getStatistics().logSummary();

String regions[] = stat.getSecondLevelCacheRegionNames();

System.***out***.println("regions----->"+regions.toString());

**for**(String regionName:regions) {

SecondLevelCacheStatistics stat2 = stat.getSecondLevelCacheStatistics(regionName);

System.***out***.println("2nd Level Cache(" +regionName+") Put Count: "+stat2.getPutCount());

System.***out***.println("2nd Level Cache(" +regionName+") HIt Count: "+stat2.getHitCount());

System.***out***.println("2nd Level Cache(" +regionName+") Miss Count: "+stat2.getMissCount());

System.***out***.println("2nd Level Cache stat2.getElementCountInMemory() : "+stat2.getElementCountInMemory());

System.***out***.println("2nd Level Cache getElementCountOnDisk() : "+stat2.getElementCountOnDisk());

}

}

**public** **static** **void** main( String[] args )

{

ApplicationContext context = **new** ClassPathXmlApplicationContext(

"app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context

.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

// bulkInsert(session);

// bulkUpdate(session);

*showQueryResult*(session);

System.***out***.println("=========================");

*showQueryResult*(session);

List<CacheManager> tempManagers = CacheManager.***ALL\_CACHE\_MANAGERS***;

System.***out***.println("# of CMs : " + tempManagers.size());

**for** (CacheManager tempCM : tempManagers) {

System.***out***.println("Got: " + tempCM.getName());

String[] cacheNames = tempCM.getCacheNames();

**for** (**int** i = 0; i < cacheNames.length; i++) {

String cacheName = cacheNames[i];

System.***out***.println(cacheName+" - "+ tempCM.getEhcache(cacheName).getStatistics().toString());

}

}

System.***out***.println("=========================");

*printStatistics*(sessionFactory);

session.flush();

session.clear();

session.close();

}

}