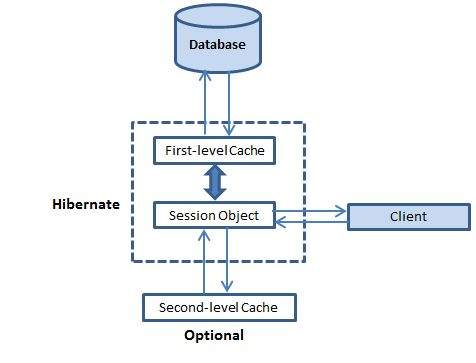
Caching is a mechanism to enhance the performance of a system. It is a buffer memory that lies between the application and the database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.



**Primary level cache** : only accessible within the session, if you query same data twice using the same session, it will not hit database twice as it uses existing data present in first level cache. Primary level cache enabled by default. It cannot be disabled.

**Secondary level cache** : can be accessible across all sessions, if you query same data twice using different session, it will not hit database twice as it uses existing data present in secondary level cache. Secondary level cache not enabled by default. It has to be enabled/configured.

**Implementing primary cache**

**1.First let mention required dependencies in pom.xml file**

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.0.1.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>4.2.0.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate.common</groupId>

<artifactId>hibernate-commons-annotations</artifactId>

<version>4.0.1.Final</version>

<classifier>tests</classifier>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.0-api</artifactId>

<version>1.0.1.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>4.1.8.Final</version>

</dependency>

<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>1.0.0.GA</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>org.jboss.logging</groupId>

<artifactId>jboss-logging</artifactId>

<version>3.1.0.CR2</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.10</version>

</dependency>

</dependencies>

**2.create hibernate.cfg.xml file inside source folder**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</property>

<property name="hibernate.connection.driver\_class">

com.mysql.jdbc.Driver

</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/cacheexample

</property>

<property name="hibernate.connection.username">

root

</property>

<property name="hibernate.connection.password">

root

</property>

<property name="show\_sql">true</property>

**<mapping class="Employee" />**

</session-factory>

</hibernate-configuration>

**3.Create Employee class(default package)**

import javax.persistence.\*;

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id @GeneratedValue

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "salary")

private int salary;

public Employee() {}

public int getId() {

return id;

}

public void setId( int id ) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName( String first\_name ) {

this.firstName = first\_name;

}

public String getLastName() {

return lastName;

}

public void setLastName( String last\_name ) {

this.lastName = last\_name;

}

public int getSalary() {

return salary;

}

public void setSalary( int salary ) {

this.salary = salary;

}

}

**4.Create HibernateConnector class which returns session factory. No need to point out where the cfg.xml is.**

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.service.ServiceRegistryBuilder;

public class HibernateConnector {

private static SessionFactory factory;

private static ServiceRegistry serviceRegistry;

public static SessionFactory getSessionFactory(){

try{

Configuration configuration = new Configuration();

configuration.configure();

serviceRegistry = new ServiceRegistryBuilder().applySettings(

configuration.getProperties()).buildServiceRegistry();

factory = configuration.buildSessionFactory(serviceRegistry);

return factory;

}catch (Throwable ex) {

System.err.println("Failed to create sessionFactory object." + ex);

throw new ExceptionInInitializerError(ex);

}

}

}

**5.Create table**

CREATE TABLE `EMPLOYEE` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(20) DEFAULT NULL,

`last\_name` varchar(20) DEFAULT NULL,

`salary` int(11) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=latin1;

**6.Create EmployeeDAO Class**

import java.util.Iterator;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class EmployeeDAO {

SessionFactory factory = HibernateConnector.getSessionFactory();

/\* Method to CREATE an employee in the database \*/

public Integer addEmployee(String fname, String lname, int salary){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try{

tx = session.beginTransaction();

Employee employee = new Employee();

employee.setFirstName(fname);

employee.setLastName(lname);

employee.setSalary(salary);

employeeID = (Integer) session.save(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

return employeeID;

}

/\* Method to READ all the employees \*/

public void listEmployees( ){

Session session = factory.openSession();

session.beginTransaction();

List employees = session.createQuery("FROM Employee").list();

employees = session.createQuery("FROM Employee").list();

for (Iterator iterator =

employees.iterator(); iterator.hasNext();){

Employee employee = (Employee) iterator.next();

System.out.print("First Name: " + employee.getFirstName());

System.out.print(" Last Name: " + employee.getLastName());

System.out.println(" Salary: " + employee.getSalary());

}

session.getTransaction().commit();

session.close();

}

/\* Method to UPDATE salary for an employee \*/

public void updateEmployee(Integer EmployeeID, int salary ){

Session session = factory.openSession();

Transaction tx = null;

try{

tx = session.beginTransaction();

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

employee.setSalary( salary );

session.update(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

}

/\* Method to DELETE an employee from the records \*/

public void deleteEmployee(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

try{

tx = session.beginTransaction();

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

session.delete(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

}

public void getEmployee(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

tx = session.beginTransaction();

**//querying twice to database but session hits database only once, second time it uses primary cache**

**Employee employee =**

**(Employee)session.get(Employee.class, EmployeeID);**

**Employee employee2 =**

**(Employee)session.get(Employee.class, EmployeeID);**

System.out.println(employee.getFirstName());

tx.commit();

session.close();

}

}

**7.Create Manage Employee class with main method**

public class ManageEmployee {

public static void main(String[] args) {

EmployeeDAO employeeDao = new EmployeeDAO();

employeeDao.getEmployee(4);

/\* Add few employee records in database \*/

/\*Integer empID1 = employeeDao.addEmployee("Zara", "Ali", 1000);

Integer empID2 = employeeDao.addEmployee("Daisy", "Das", 5000);

Integer empID3 = employeeDao.addEmployee("John", "Paul", 10000);\*/

/\* List down all the employees \*/

//employeeDao.listEmployees();

//employeeDao.getEmployeeSecondLevelCache(4);

/\* Update employee's records \*/

//employeeDao.updateEmployee(empID1, 5000);

/\* Delete an employee from the database \*/

//employeeDao.deleteEmployee(empID2);

}

}

o/p : for employee id 4

Hibernate: select employee0\_.id as id0\_0\_, employee0\_.first\_name as first2\_0\_0\_, employee0\_.last\_name as last3\_0\_0\_, employee0\_.salary as salary0\_0\_ from EMPLOYEE employee0\_ where employee0\_.id=?

arun

**Secondary Level cache**

1.We will use the same database table

2.use the same hibernate connector class

3.add secondary cache properties into hibernate.cfg.xml

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</property>

<property name="hibernate.connection.driver\_class">

com.mysql.jdbc.Driver

</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/cacheexample</property>

<property name="hibernate.connection.username">

root

</property>

<property name="hibernate.connection.password">

root

</property>

**<property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.EhCacheRegionFactory </property>**

**<property name="hibernate.cache.use\_second\_level\_cache">true</property>**

<property name="show\_sql">true</property>

**<mapping class="Employee" />**

</session-factory>

</hibernate-configuration>

**4.add secondary cache dependency in pom file, then do clean/maven update**

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-ehcache -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-ehcache</artifactId>

<version>4.0.1.Final</version>

</dependency>

**5.add cacheable annotation in Employee class**

import javax.persistence.\*;

import org.hibernate.annotations.CacheConcurrencyStrategy;

@Entity

**@Cacheable**

**@org.hibernate.annotations.Cache(usage = CacheConcurrencyStrategy.READ\_ONLY)**

@Table(name = "EMPLOYEE")

public class Employee {

@Id @GeneratedValue

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "salary")

private int salary;

public Employee() {}

public int getId() {

return id;

}

public void setId( int id ) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName( String first\_name ) {

this.firstName = first\_name;

}

public String getLastName() {

return lastName;

}

public void setLastName( String last\_name ) {

this.lastName = last\_name;

}

public int getSalary() {

return salary;

}

public void setSalary( int salary ) {

this.salary = salary;

}

}

**6.Create EmployeeDAO class** , in one of method we will use two different sessions to access same query/data. Session will hit database only once, second time it uses secondary level cache.

import java.util.Iterator;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class EmployeeDAO {

SessionFactory factory = HibernateConnector.getSessionFactory();

/\* Method to CREATE an employee in the database \*/

public Integer addEmployee(String fname, String lname, int salary){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try{

tx = session.beginTransaction();

Employee employee = new Employee();

employee.setFirstName(fname);

employee.setLastName(lname);

employee.setSalary(salary);

employeeID = (Integer) session.save(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

return employeeID;

}

/\* Method to READ all the employees \*/

public void listEmployees( ){

Session session = factory.openSession();

session.beginTransaction();

List employees = session.createQuery("FROM Employee").list();

employees = session.createQuery("FROM Employee").list();

for (Iterator iterator =

employees.iterator(); iterator.hasNext();){

Employee employee = (Employee) iterator.next();

System.out.print("First Name: " + employee.getFirstName());

System.out.print(" Last Name: " + employee.getLastName());

System.out.println(" Salary: " + employee.getSalary());

}

session.getTransaction().commit();

session.close();

}

/\* Method to UPDATE salary for an employee \*/

public void updateEmployee(Integer EmployeeID, int salary ){

Session session = factory.openSession();

Transaction tx = null;

try{

tx = session.beginTransaction();

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

employee.setSalary( salary );

session.update(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

}

/\* Method to DELETE an employee from the records \*/

public void deleteEmployee(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

try{

tx = session.beginTransaction();

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

session.delete(employee);

tx.commit();

}catch (HibernateException e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}finally {

session.close();

}

}

public void getEmployee(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

tx = session.beginTransaction();

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

Employee employee2 =

(Employee)session.get(Employee.class, EmployeeID);

System.out.println(employee.getFirstName());

tx.commit();

session.close();

}

public void getEmployeeSecondLevelCache(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

tx = session.beginTransaction();

//querying first time

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

System.out.println(employee.getFirstName());

tx.commit();

session.close();

//querying second time with different session

Session session2 = factory.openSession();

Transaction tx2 = session2.beginTransaction();

Employee employee2 =

(Employee)session2.get(Employee.class, EmployeeID);

System.out.println(employee2.getFirstName());

tx2.commit();

session2.close();

}

}

7.Create ManageEmployee class with main method

public class ManageEmployee {

public static void main(String[] args) {

EmployeeDAO employeeDao = new EmployeeDAO();

employeeDao.getEmployeeSecondLevelCache(4);

/\* Add few employee records in database \*/

/\*Integer empID1 = employeeDao.addEmployee("Zara", "Ali", 1000);

Integer empID2 = employeeDao.addEmployee("Daisy", "Das", 5000);

Integer empID3 = employeeDao.addEmployee("John", "Paul", 10000);\*/

/\* List down all the employees \*/

//employeeDao.listEmployees();

//employeeDao.getEmployee(4);

/\* Update employee's records \*/

//employeeDao.updateEmployee(empID1, 5000);

/\* Delete an employee from the database \*/

//employeeDao.deleteEmployee(empID2);

}

}

Output

Hibernate: select employee0\_.id as id0\_0\_, employee0\_.first\_name as first2\_0\_0\_, employee0\_.last\_name as last3\_0\_0\_, employee0\_.salary as salary0\_0\_ from EMPLOYEE employee0\_ where employee0\_.id=?//hits db only once

arun

arun

**In first level cache use below code in EmployeeDAO to see why second cache level is not activated by default**

public void getEmployee(Integer EmployeeID){

Session session = factory.openSession();

Transaction tx = null;

tx = session.beginTransaction();

//querying twice to database but session hits database only once, second time it uses primary cache

Employee employee =

(Employee)session.get(Employee.class, EmployeeID);

Employee employee2 =

(Employee)session.get(Employee.class, EmployeeID);

System.out.println(employee.getFirstName());

tx.commit();

session.close();

**//second time**

**/\* Session session2 = factory.openSession();**

**Transaction tx2 = session2.beginTransaction();**

**Employee emp3 = (Employee) session2.get(Employee.class, EmployeeID);**

**System.out.println(emp3.getFirstName());**

**tx2.commit();**

**session2.close();\*/**

}