**[clear(), evict() and close() methods in Hibernate](http://www.connect2java.com/tutorials/hibernate/clear-evict-and-close-methods-in-hibernate/" \o "Permalink to clear(), evict() and close() methods in Hibernate)**

**clear():**Completely clear the session and is used to dissociate/disconnect all the objects from the session.

public void clear()

**Example on clear()**

**Database table before execution**

SQL> SELECT \* FROM EMPLOYEE;

     EMPNO EMPNAME              EMPDEPT

---------- -------------------- --------------------

         2 Sree                 R&D

         1 Kalyan               Developement

ExampleOnClear.java

Java

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import com.connect2java.HibernateSample1.Employee;

public class ExampleOnClear {

    public static void main(String[] args) {

        Session session = getSession();

        session.getTransaction().begin();

        try{

            Employee emp1 = (Employee)session.get(Employee.class, 1);

            Employee emp2 = (Employee)session.get(Employee.class, 2);

            //emp1 and emp2 are in persistent state.

            emp1.setEmpName("NEW SREE");

            emp2.setEmpName("NEW KALYAN");

            session.clear();

            //emp1 and emp2 are in detached state.

            session.getTransaction().commit();

            session.close();

        }catch(HibernateException e){

            e.printStackTrace();

        }

    }

public static Session getSession() {

        Session session=null;

        try{

            Configuration configuration=  new Configuration().

                    configure("hibernate.cfg.xml");

            StandardServiceRegistryBuilder serviceRegistryBuilder =

                    new StandardServiceRegistryBuilder();

            serviceRegistryBuilder.applySettings(configuration.getProperties());

            ServiceRegistry serviceRegistry = serviceRegistryBuilder.build();

            SessionFactory sessionFactory = configuration.buildSessionFactory

                    (serviceRegistry);

            session = sessionFactory.openSession();

            return session;

        }catch(Exception e){

            e.printStackTrace();

        }

        return session;

     }

}

**Database table after execution**

SQL> SELECT \* FROM EMPLOYEE;

     EMPNO EMPNAME              EMPDEPT

---------- -------------------- --------------------

         2 Sree                 R&D

         1 Kalyan               Developement

* After calling session.clear(), emp1 and emp2 objects are disconnected from the session object. i.e emp1 and emp2 objects are moved from persistent state to detached state.
* The table is not updated even though transaction is committed becuase emp1 and emp2 are in non-transaction state.

**evict():** Removes the object from the session. This method is used to dissociate/disconnect the specified object from the session

public void evict(Object object) throws HibernateException

**Example on evict()**

**Database table before execution** SQL> SELECT \* FROM EMPLOYEE;

     EMPNO EMPNAME              EMPDEPT

---------- -------------------- --------------------

         2 Sree                 R&D

         1 Kalyan               Developement

ExampleOnEvict.java

Java

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import com.connect2java.HibernateSample1.Employee;

public class ExampleOnEvict {

    public static void main(String[] args) {

        Session session = getSession();

        session.getTransaction().begin();

        try{

            Employee emp1 = (Employee)session.get(Employee.class, 1);

            Employee emp2 = (Employee)session.get(Employee.class, 2);

            //emp1 and emp2 are in persistent state.

            emp1.setEmpName("NEW KALYAN");

            emp2.setEmpName("NEW SREE");

            session.evict(emp1);

            //emp1 is in detached state and emp2 is in persistent state.

            session.getTransaction().commit();

            session.close();

        }catch(HibernateException e){

            e.printStackTrace();

        }

    }

public static Session getSession() {

Session session=null;

        try{

            Configuration configuration=  new Configuration().

                    configure("hibernate.cfg.xml");

            StandardServiceRegistryBuilder serviceRegistryBuilder =

                    new StandardServiceRegistryBuilder();

        serviceRegistryBuilder.applySettings(configuration.getProperties());

            ServiceRegistry serviceRegistry = serviceRegistryBuilder.build();

            SessionFactory sessionFactory = configuration.buildSessionFactory

                    (serviceRegistry);

            session = sessionFactory.openSession();

            return session;

        }catch(Exception e){

        e.printStackTrace();

        }

        return session;

     }

}

**Database table after execution**

SQL> SELECT \* FROM EMPLOYEE;

     EMPNO EMPNAME              EMPDEPT

---------- -------------------- --------------------

         2 NEW SREE              R&D

         1 Kalyan                Developement

**Output in Eclipse**

Hibernate: select employee0\_.EMPNO as EMPNO1\_0\_0\_, employee0\_.EMPNAME as EMPNAME2\_0\_0\_, employee0\_.EMPDEPT as EMPDEPT3\_0\_0\_ from EMPLOYEE employee0\_ where employee0\_.EMPNO=?

Hibernate: select employee0\_.EMPNO as EMPNO1\_0\_0\_, employee0\_.EMPNAME as EMPNAME2\_0\_0\_, employee0\_.EMPDEPT as EMPDEPT3\_0\_0\_ from EMPLOYEE employee0\_ where employee0\_.EMPNO=?

Hibernate: update EMPLOYEE set EMPNAME=?, EMPDEPT=? where EMPNO=?

emp1 is not updated becuase we have called session.evict(emp1) and emp2 is updated.

**close():**

public Connection close() throws HibernateException

close the session by calling session.close() after transaction is completed.All the associated objects will be dissociated after calling session.close().It is not  strictly necessary to close the session but you must at least using disconnct it using session.disconnect()

Related Posts :

* [Hibernate: One To Many Example using Annotation](http://www.connect2java.com/tutorials/hibernate/hibernate-one-to-many-example-using-annotation/)
* [Hibernate: One To Many Example using XML mapping](http://www.connect2java.com/tutorials/hibernate/hibernate-one-to-many-example-using-xml-mapping/)
* [load() and get() methods of Hibernate Session](http://www.connect2java.com/tutorials/hibernate/load-and-get-methods-of-hibernate-session/)
* [saveOrUpdate() method in Hibernate Session](http://www.connect2java.com/tutorials/hibernate/saveorupdate-in-hibernate-session/)
* [What is dialect class in Hibernate?](http://www.connect2java.com/tutorials/hibernate/what-is-dialect-class-in-hibernate/)
* [contains() and isConnected methods in Hibernate](http://www.connect2java.com/tutorials/hibernate/contains-and-isconnected-methods-in-hibernate/)
* [flush() method in Hibernate](http://www.connect2java.com/tutorials/hibernate/flush-method-in-hibernate/)
* [refresh() method in Hibernate](http://www.connect2java.com/tutorials/hibernate/refresh-method-in-hibernate/)
* [merge() method in Hibernate](http://www.connect2java.com/tutorials/hibernate/merge-method-in-hibernate/)