**Hibernate Merging and Refreshing Entities**

In previous tutorial, we learned about [**saving entities using save() or saveOrUpdate() methods in hibernate**](https://howtodoinjava.com/hibernate/save-and-saveorupdate-for-saving-hibernate-entities/). There we learned that in most of the cases, you will not even need those methods as hibernate manages the updates on [**persistent entities**](https://howtodoinjava.com/hibernate/hibernate-entity-persistence-lifecycle-states/) automatically. You only need to care about transient objects in most of cases. In this tutorial, I am discussing few thoughts around refresh() and merge() method present in hibernate session class.

**Refreshing Hibernate Entities Using refresh() Method**

Sometimes we face situation where we application database is modified with some external application/agent and thus corresponding hibernate entity in your application actually becomes out of sync with it’s database representation i.e. having old data. In this case, you can use **session.refresh()method to re-populate the entity with latest data available in database**.

You can use one of the refresh() methods on the Session interface to refresh an instance of a persistent object, as follows:

public void refresh(Object object) throws HibernateException

public void refresh(Object object, LockMode lockMode) throws HibernateException

These methods will reload the properties of the object from the database, overwriting them. In real life applications, you do not have to use the refresh() method very often apart from above stated scenario.

Let’s look at an example of refresh() method.

public class RefreshEntityExample

{

public static void main(String[] args)

{

Session sessionOne = HibernateUtil.getSessionFactory().openSession();

sessionOne.beginTransaction();

//Create new Employee object

EmployeeEntity emp = new EmployeeEntity();

emp.setEmployeeId(1);

emp.setFirstName("Lokesh");

emp.setLastName("Gupta");

//Save employee

sessionOne.save(emp);

sessionOne.getTransaction().commit();

sessionOne.close();

//Verify employee's firstname

System.out.println(verifyEmployeeFirstName(1, "Lokesh"));

Session sessionTwo = HibernateUtil.getSessionFactory().openSession();

sessionTwo.beginTransaction();

//This

emp.setFirstName("Vikas");

sessionTwo.refresh(emp);

sessionTwo.getTransaction().commit();

sessionTwo.close();

System.out.println(emp.getFirstName().equals("Lokesh"));

HibernateUtil.shutdown();

}

private static boolean verifyEmployeeFirstName(Integer employeeId, String firstName){

Session session = HibernateUtil.getSessionFactory().openSession();

EmployeeEntity employee = (EmployeeEntity) session.load(EmployeeEntity.class, employeeId);

//Verify first name

boolean result = firstName.equals(employee.getFirstName());

session.close();

//Return verification result

return result;

}

}

Output:

true

true

Look above the highlighted lines.

* Line 15 save the employee with first name “Lokesh”
* Line 26 set the first name “Vikas”. As entity is detached, DB will not be updated.
* Line 27 refresh the entity with database using refresh() method.
* Line 32 verify that firstname set in entity has been updated with “Lokesh” as it is what database have this moment.

This was all about refresh method. Let’s look an another similar method merge().

**Merging Hibernate Entities Using merge() Method**

Method merge() does exactly opposite to what refresh() does i.e. It updates the database with values from a detached entity. Refresh method was updating the entity with latest database information. So basically, both are exactly opposite.

Merging is performed when you desire to have a **detached entity changed to persistent state** again, with the detached entity’s changes migrated to (or overriding) the database. The method signatures for the merge operations are:

Object merge(Object object)

Object merge(String entityName, Object object)

Hibernate official documentation give a very good explanation of merge() method:

Copy the state of the given object onto the persistent object with the same identifier. If there is no persistent instance currently associated with the session, it will be loaded. Return the persistent instance. If the given instance is unsaved, save a copy of and return it as a newly persistent instance. The given instance does not become associated with the session. This operation cascades to associated instances if the association is mapped with cascade=”merge”.

So if I take below code for example then below listed points should be clear to you.

EmployeeEntity mergedEmpEntity = session.merge(empEntity);

* ’empEntity’ is detached entity when it is passed to merge() method.
* merge() method will search for an already loaded EmployeeEntity instance with identifier information taken from empEntity. If such persistent entity is found then it will be used for updates. Other wise a new EmployeeEntity is loaded into session using same identifier information as present in ’empEntity’.
* Data is copied from ’empEntity’ to new found/loaded entity.
* Because new/found entity is persistent, all data copied to it from ’empEntity’ is automatically saved into database.
* Reference of that new entity is returned from merge() method and is assigned to ‘mergedEmpEntity’ variable.
* ’empEntity’ is still detached entity.

public class MergeEntityExample

{

public static void main(String[] args)

{

Session sessionOne = HibernateUtil.getSessionFactory().openSession();

sessionOne.beginTransaction();

//Create new Employee object

EmployeeEntity emp = new EmployeeEntity();

emp.setEmployeeId(1);

emp.setFirstName("Lokesh");

emp.setLastName("Gupta");

//Save employee

sessionOne.save(emp);

sessionOne.getTransaction().commit();

sessionOne.close();

//Verify employee's firstname

System.out.println(verifyEmployeeFirstName(1, "Lokesh"));

Session sessionTwo = HibernateUtil.getSessionFactory().openSession();

sessionTwo.beginTransaction();

//Set new first name

emp.setFirstName("Vikas");

//Merge the emp object using merge() method

EmployeeEntity mergedPersistentEmpEntity = (EmployeeEntity) sessionTwo.merge(emp);

sessionTwo.getTransaction().commit();

sessionTwo.close();

//Verify employee's firstname again in database

System.out.println(verifyEmployeeFirstName(1, "Vikas"));

HibernateUtil.shutdown();

}

private static boolean verifyEmployeeFirstName(Integer employeeId, String firstName){

Session session = HibernateUtil.getSessionFactory().openSession();

EmployeeEntity employee = (EmployeeEntity) session.load(EmployeeEntity.class, employeeId);

//Verify first name

boolean result = firstName.equals(employee.getFirstName());

session.close();

//Return verification result

return result;

}

}

Output:

true

true

In above example, ‘mergedPersistentEmpEntity’ is new entity which is persistent. So if you want to any more change, then make in in ‘mergedPersistentEmpEntity’ instance.

That’s all for this tutorial covering **merge() and refresh() methods in hibernate**. Remember that question can be asked as **difference between merge() and refresh()**, or**difference between merge()and load()** etc.