Advance Hibernate Usages

Version 1.0
Date 08/11/10

Revision history

Doc No:	Version no.	Change reference no.	Author	Published date	Sections changed	Description of changes
	1.0		Dikshit	08/11/10		Base Document

CONTENTS

Generic Class For Hibernate Operations

The class given below , “HibernateUtil.java” can be used for all basic CRUD operations

Listing 1 : HibernateUtil.java

```
import java.util.List;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

/**
 *
 * Generic Hibernate Helper Utility
 */
public class HibernateUtil {
    private static final SessionFactory sessionFactory;


   static {
      try {
         // Create the SessionFactory from standard (hibernate.cfg.xml)config file.

         sessionFactory = new Configuration().configure().buildSessionFactory();
      } catch (Throwable ex) {
         // Log the exception.
         System.err.println("Initial SessionFactory creation failed." + ex);
         throw new ExceptionInInitializerError(ex);
      }
   }

   public static SessionFactory getSessionFactory() {
      return sessionFactory;
   }

   public static List<Object> select(String queryString) {
      List<Object> result = null;
      Transaction tx =null;
      try {
         Session session = getSessionFactory().getCurrentSession();
          tx = session.beginTransaction();
         Query query = session.createQuery(queryString);
         result = query.list();
```

```
            tx.commit();
        } catch (HibernateException e) {
            tx.rollback();
        }
        return result;
    }
    public static Object uniqueSelect(String queryString) {
        Object result = null;
         Transaction tx=null;
        try {
            Session session = getSessionFactory().getCurrentSession();
             tx = session.beginTransaction();
            Query query = session.createQuery(queryString);
            result = query.uniqueResult();
            tx.commit();
        } catch (HibernateException e) {
            tx.rollback();
        }
        return result;
    }

    public void update(Object obj){
        Transaction tx =null;
        try {
            Session session = getSessionFactory().getCurrentSession();
             tx = session.beginTransaction();
            session.update(obj);

            tx.commit();

        } catch (HibernateException e) {

            tx.rollback();
        }

    }
    public void delete(Object obj){
        Transaction tx =null;
        try {
            Session session = getSessionFactory().getCurrentSession();
             tx = session.beginTransaction();
            session.delete(obj);
            tx.commit();
        } catch (HibernateException e) {
            tx.rollback();
        }
```

```
    }
    public void save(Object obj){
        Transaction tx =null;
        try {
            Session session = getSessionFactory().getCurrentSession();
             tx = session.beginTransaction();
            session.save(obj);
            tx.commit();
        } catch (HibernateException e) {
            tx.rollback();
        }
    }
}
```

Generic Hibernate helper class for Using with IceFaces Project

The class, DaoUtil.java given below can be used for all generic JSF usages like getting a list of SelectItems objects etc.

It depends on the “HibernateUtil” class given above.

Listing 2 : DaoUtil.java

```
import com.sun.webui.jsf.model.Option;
import java.util.Collection;
import java.util.List;
import javax.faces.model.SelectItem;
import org.hibernate.Session;
import org.hibernate.StatelessSession;
import org.hibernate.Transaction;

/**
 *
 * Hibernate helper class
 * to be used with Icefaces
 */
public class DaoUtil {

    private static HibernateUtil hutil;


    static {
```

```
    hutil = new HibernateUtil();
}

public static Option[] getOptionsForDD(String query) {
    Option[] DDOptions = null;
    List<Object> result = HibernateUtil.select(query);
    if (!result.isEmpty()) {
        int size = result.size();
        DDOptions = new Option[size];

        for (int i = 0; i < size; i++) {
            Object[] row = (Object[]) result.get(i);
            Option opt = new Option(row[0], row[1].toString());
            DDOptions[i] = opt;
        }
    }
    return DDOptions;
}

public static SelectItem[] getSelectItemsForDD(String query) {
    SelectItem[] DDitems = null;
    List<Object> result = HibernateUtil.select(query);
    if (!result.isEmpty()) {
        int size = result.size();
        DDitems = new SelectItem[size];

        for (int i = 0; i < size; i++) {
            Object[] row = (Object[]) result.get(i);
            SelectItem st = new SelectItem(row[0], row[1].toString());
            DDitems[i] = st;
        }
    }
    return DDitems;
}

public static SelectItem[] getMSelectItemsForDD(String query) {
    SelectItem[] DDitems = null;
    List<Object> result = HibernateUtil.select(query);
    if (!result.isEmpty()) {
        result.add(0, new Object[]{0, "Select"});
        int size = result.size();
        DDitems = new SelectItem[size];

        for (int i = 0; i < size; i++) {
            Object[] row = (Object[]) result.get(i);
            SelectItem st = new SelectItem(row[0], row[1].toString());
```

```
            DDitems[i] = st;
        }
    }
    return DDitems;
}

public static List<Object> doQuery(String query) {
    List<Object> result = HibernateUtil.select(query);
    return result;

}

public static boolean isCodeExists(String query) {
    List<Object> result = HibernateUtil.select(query);
    if (!result.isEmpty()) {
        return true;
    } else {
        return false;
    }

}

public static <T> T load(Class<T> type, int id) {
    Object obj = null;
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    Transaction tx = session.beginTransaction();
    obj = session.load(type, id);
    tx.commit();
    return type.cast(obj);
}

public static <T> T fetchLoad(Class<T> type, String query) {
    Object obj = HibernateUtil.uniqueSelect(query);
    return type.cast(obj);
}

public static void save(Object obj) {
    hutil.save(obj);
}

public static void update(Object obj) {
    hutil.update(obj);
}

public static void delete(Object obj) {
    hutil.delete(obj);
```

```
    }


    public static void batchInsert(Object[] list) {
        long startTime = System.currentTimeMillis();
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        Transaction tx = session.beginTransaction();
        int size = list.length;
        for (int i = 0; i < size; i++) {
            Object obj = list[i];

            session.save(obj);

            if (i % 30 == 0) { //20, same as the JDBC batch size
                //flush a batch of inserts and release memory:
                session.flush();
                session.clear();
            }

        }
        tx.commit();

    }

    public static void refreshState() {
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();
        Transaction tx = session.beginTransaction();

        session.clear();
        tx.commit();
    }
}
```