

---

# Hibernate Projection Steps Documentation

---

## Steps of Hibernate Projection

---

Author : Kiran

---

So far in criteria, we are able to load complete object right....! let us see [how to load the partial objects while working with criteria](#). The projections concept is introduced in hibernate 3.0 and mainly we can do the following 2 operations using the projection

- We can load partial object from the database
- We can find the Result of Aggregate functions

*Projection* is an [Interface](#) given in "[org.hibernate.criterion](#)" package, *Projections* is a class given in same package, actually Projection is an interface, and *Projections* is a class and is a factory for producing projection objects.

In Projections class, we have all [static methods](#) and each method of this class returns Projection interface object.

If we want to add a Projection object to [Criteria](#) then we need to call a method **setProjection()**

**Remember**, while adding projection object to criteria, it is possible to add one object at a time. It means if we add 2nd projection object then this 2nd one will [overrides](#) the first one (first one wont be work), so at a time we can only one projection object to criteria object.

## Projection Example

1. Create simple java project (Projection Example)
2. Add all mysql and all hibernate jar (configure build path)
3. Create hibernate.cfg.xml file ( Using Jboss Tool )
4. Create pojo class (using Hibernate Code Generation )
5. Create Projection\_Ex\_Test.java class
6. Create HibernateUtil.java class



HibernateUtil.java

```
package com.jbk.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import com.jbk.model.Employee;

public class HibernateUtil {
    private static SessionFactory sf=null;
    public static SessionFactory getSessionFactory()
    {
        Configuration configuration=new Configuration();
        configuration.configure().addAnnotatedClass(Employee.class);
        sf=configuration.buildSessionFactory();
        return sf;
    }
}
```

### 1) Example 1

If we want to read a partial entity (selected columns) from the database,hibernate has provided us with a Projection interface.

```
package com.jbk.projection;
```

```
import java.util.Iterator;
import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projection;
import org.hibernate.criterion.ProjectionList;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;

public class ProjectionEx {

    public static void main(String[] args) {

        Session session =
HibernateUtil.getSessionFactory().openSession();
        Criteria criteria = session.createCriteria(Employee.class);

        // Get Single Entity Data using property("propertyName")
        System.out.println("***** Single Coloumn *****");
        Projection projection = Projections.property("empName");
        criteria.setProjection(projection);
        List<String> list = criteria.list();
        for (String string : list) {
            System.out.println(string);
        }
    }
}
```

## 2) Example 2

If we want to read more than 1 column (employeeName, salary) with projections, we should have to use ProjectinList

```
package com.jbk.projection;

import java.util.Iterator;
```

```
import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projection;
import org.hibernate.criterion.ProjectionList;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;

public class ProjectionEx {

    public static void main(String[] args) {

        Session session =
            HibernateUtil.getSessionFactory().openSession();
        Criteria criteria = session.createCriteria(Employee.class);
        System.out.println("*** Multiple Coloumn ***");
        Criteria criteria2 = session.createCriteria(Employee.class);
        Projection projection1 = Projections.property("empName ");
        Projection projection2 = Projections.property("salary");

        ProjectionList projectionList = Projections.projectionList();
        projectionList.add(projection1);
        projectionList.add(projection2);

        criteria2.setProjection(projectionList);
        List<Object> empObj = criteria2.list();

        Iterator itr2 = empObj.iterator();
        while (itr2.hasNext()) {
            Object[] obj=(Object[]) itr2.next();
            System.out.print(obj[0]+"\\t");
            System.out.print(obj[1]+"\\n");
        }
    }
}
```

### Example 3

If we want to count the total number of rows ( records ) inside table.

```
package com.jbk.projection;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;

public class Test {

    public static void main(String[] args) {
        Session session =
            HibernateUtil.getSessionFactory().openSession();
        Criteria criteria = session.createCriteria(Employee.class);

        criteria.setProjection(Projections.rowCount());
        List rows=criteria.list();
        System.out.println(rows);
    }
}
```

### Example 4

If you want to count the distinct( On Specific column ) number of rows ( records ) inside table.

```
package com.jbk.projection;

import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;
```

```
public class Test {  
    public static void main(String[] args) {  
        Session session =  
        HibernateUtil.getSessionFactory().openSession();  
        Criteria criteria = session.createCriteria(Employee.class);  
        criteria.setProjection(Projections.distinct(Projections.countDistinct("jobName")));  
        List rows=criteria.list();  
        System.out.println(rows);  
    }  
}
```

### 3) Example 5

If you want to calculate the average, The avg() function aggregates the average value of the given column.

```
package com.jbk.projection;  
  
import java.util.List;  
import org.hibernate.Criteria;  
import org.hibernate.Session;  
import org.hibernate.criterion.Projections;  
import com.jbk.model.Employee;  
import com.jbk.util.HibernateUtil;  
  
public class Test {  
    public static void main(String[] args) {  
        Session session =  
        HibernateUtil.getSessionFactory().openSession();  
        Criteria criteria = session.createCriteria(Employee.class);  
        criteria.setProjection(Projections.avg("salary"));  
        List avg=criteria.list();  
        System.out.println(avg);  
    }  
}
```

#### 4) Example 6

If you want to find out the min and max number (ex.salary),The min() and max() function aggregates the min and max value of the given column.

```
package com.jbk.projection;

import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;

public class Test {

    public static void main(String[] args) {
        Session session =
            HibernateUtil.getSessionFactory().openSession();
        Criteria criteria = session.createCriteria(Employee.class);

        //criteria.setProjection(Projections.min("salary"));
        criteria.setProjection(Projections.max("salary"));
        List max=criteria.list();
        System.out.println(max);
    }
}
```

#### 5) Example 7

If you want to calculate sum of particular coloumn (ex.salary),The sum() function aggregates the sum of the given column.

```
package com.jbk.projection;

import java.util.List;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.criterion.Projections;
import com.jbk.model.Employee;
import com.jbk.util.HibernateUtil;
```



```
public class Test {  
  
    public static void main(String[] args) {  
        Session session =  
            HibernateUtil.getSessionFactory().openSession();  
        Criteria criteria = session.createCriteria(Employee.class);  
  
        criteria.setProjection(Projections.sum("salary"));  
        List max=criteria.list();  
        System.out.println(max);  
    }  
}
```

