Once an object is saved in a database, we can modify that object any number of times right, If we want to know how many no of times that an object is modified then we need to apply this versioning concept.  
When ever we use versioning then hibernate inserts version number as **zero**, when ever object is saved for the first time in the database.  Later hibernate increments that version no by one automatically when ever a modification is done on that particular object.  
In order to use this versioning concept, we need the following two changes in our application

* Add one property of type int in our pojo class
* In hibernate mapping file, add an element called version soon after id element

Files required to execute this program..

* Product.java (My POJO class)
* Product.hbm.xml  (Xml mapping file )
* hibernate.cfg.xml  (Xml configuration file)
* ClientForSave\_1.java (java file to write our hibernate logic)
* ClientForUpdate\_2.java

**Product.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | package str;    public class Product{        private int productId;      private String proName;      private double price;            private int v;        public void setProductId(int productId)      {          this.productId = productId;      }      public int getProductId()      {          return productId;      }        public void setProName(String proName)      {          this.proName = proName;      }      public String getProName()      {          return proName;      }        public void setPrice(double price)      {          this.price = price;      }      public double getPrice()      {          return price;      }           public void setV(int v)      {          this.v = v;      }      public int getV()      {          return v;      }  } |

**Product.hbm.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0"?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>">    <hibernate-mapping>  <class name="str.Product" table="products">    <id name="productId" column="pid"  />  <version name="v" column="ver" />  <property name="proName" column="pname" length="10"/>  <property name="price"/>    </class>  </hibernate-mapping> |

**Note**:

* In this above mapping file, find the <version> element, there i have given column name as **version**, actually you can write any name its not predefined.

**hibernate.cfg.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <?xml version='1.0' encoding='UTF-8'?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "<http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd>">    <hibernate-configuration>  <session-factory>  <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver  </property>  <property name="connection.url">jdbc:oracle:thin:@www.java4s.com:1521:XE</property>  <property name="connection.username">system</property>  <property name="connection.password">admin</property>    <property name="dialect">org.hibernate.dialect.OracleDialect</property>  <property name="show\_sql">true</property>  <property name="hbm2ddl.auto">update</property>    <mapping resource="Product.hbm.xml"></mapping>  </session-factory>  </hibernate-configuration> |

**ClientForSave\_1.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package str;    import org.hibernate.\*;  import org.hibernate.cfg.\*;    public class ClientForSave\_1 {        public static void main(String[] args)      {            Configuration cfg = new Configuration();          cfg.configure("hibernate.cfg.xml");            SessionFactory factory = cfg.buildSessionFactory();          Session session = factory.openSession();          Product p=new Product();            p.setProductId(104);          p.setProName("AC");          p.setPrice(14000);            Transaction tx = session.beginTransaction();          session.save(p);          System.out.println("Object saved successfully.....!!");          tx.commit();          session.close();          factory.close();      }    } |

**Note**:

* Remember friends, first we must run the logic to **save** the object then hibernate will inset 0 (Zero) by default in the version column of the database, its very important point in the interview point of view also
* First save logic to let the hibernate to insert zero in the version column, then any number ofupdate logic’s (programs) we run, hibernate will increments +1 to the previous value
* But if we run the update logic for the first time, hibernate will not insert zero..! it will try toincrement the previous value which is NULL in the database so we will get the exception.

**ClientForUpdate\_2.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package str;    import org.hibernate.\*;  import org.hibernate.cfg.\*;    public class ClientForUpdate\_2 {        public static void main(String[] args)      {            Configuration cfg = new Configuration();          cfg.configure("hibernate.cfg.xml");            SessionFactory factory = cfg.buildSessionFactory();          Session session = factory.openSession();          Object o=session.load(Product.class,new Integer(104));          Product s=(Product)o;            Transaction tx = session.beginTransaction();             s.setPrice(4000); // implicitly update method will be call            tx.commit();            System.out.println("Object Updated successfully.....!!");          session.close();          factory.close();      }    } |

**Note**(**Important..!!!**)\_\_\_\_\_\_  
Guys your know some thing.., actually we can run any logic (Save or Update)  for the first time, but make sure the versioning column is a number (>=0), but save logic has ability to insert zero by default if there is no value, and update logic will directly tries to increments already existing value by 1, it wont insert any value by default if its null, hope you are clear about this point, and mates its the very important concept in the interview point of view, if you have any doubt just Ask a Question on this our team will respond ASAP