***Hibernate and Combination of Annotations:***

**Q) *Is it possible to execute hibernate applications without providing mapping file?***

**Yes it is possible we are achieving this by using Annotations.**

**Note: Hibernate 3.x version made XML mapping file as an optional, as an alternative to xml mapping file Hibernate 3.x version has provided a set of annotations.**

**The main intension of introducing Annotations in 3.x version describes Meta data by eliminating xml files. That is Annotations are used to reduce the xml dependency in Enterprise Applications as much as possible.**

**Note: In hibernate applications it is possible to use Annotations as an alternative to mapping xml files only, it is not possible to use annotations as an alternative for XML Configuration file. Because Hibernate software was designed in such a way that to pickup all the configuration details from xml Configuration file only.**

**Jar files;**

**ejb3-persistance.jar;**

**Hibernate-annotations.jar;**

**hibernate-commons-annotations.jar;**

**The fallowing annotations from javax.persistance pacakage.**

**@Entity**

**@Table**

**@Id**

**@Column**

1. **@Entity: It is a Marker Annotation, Class level annotation, it can be used to represent or describe the present java class is representing an Entity available in database.**
2. **@Table: It is a Multi valued annotation, Class level annotation. It can be used to provide the mapping between present pojo class and the respective Database Table.**

**Syntax: @Table(name:” “)**

**Where “name” attribute will take the respective Database table name.**

**Eg:**

**import javax.persistance.\*;**

**@Entity**

**@Table(name:”emp”)**

**public class Employee**

**{**

**//POJO class variables and fallowed by setters and getters**

**}**

1. **@Id: It is a Multi valued annotation, a variable level and method level annotation, it can be used to provide the mapping between Id property from POJO class and primary key column on the respective DataBase Table.**

**Eg:**

**public class Employee**

**{ @Id**

**private int eno;**

**}**

***or***

**public class Employee**

**{ private int eno;**

**@Id**

**public int getEno()**

**{**

**}**

**}**

1. **@Column: It is a Multi valued annotation, a variable level and method level annotation, it can be used to provide the mapping between normal properties from POJO class and the normal columns in the respective DataBase Table.**

**Syntax: @Column(name:” “)**

**Eg:**

**public class Employee**

**{**

**@Column(name:”e\_name”)**

**private String ename;**

**}**

***or***

**public class Employee**

**{**

**private int eno;**

**@Column(name:”e\_name”)**

**public int getEno()**

**{**

**}**

**}**

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="product")

**public** **class** Product {

**private** String pid;

**public** **void** setPid(String pid) {

**this**.pid = pid;

}

@Id

@Column(name="pid")

**public** String getPid() {

**return** pid;

}

@Column(name="pname")

**public** String getPname() {

**return** pname;

}

**public** **void** setPname(String pname) {

**this**.pname = pname;

}

@Column(name="pcost")

**public** **int** getPcost() {

**return** pcost;

}

**public** **void** setPcost(**int** pcost) {

**this**.pcost = pcost;

}

**private** String pname;

**private** **int** pcost;

}

***@@@@IMPORTANT to NOTE***

***Main step:***

***Configure Annotated POJO class in Hibernate Configuration file, for this we have to use the fallowing tag in hibernate configuration file.***

***Syntax: <mapping class=”Employee“/>***

***<!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">***

***sun.jdbc.odbc.JdbcOdbcDriver***

***</property>***

***<property name="connection.url">***

***jdbc:odbc:challa***

***</property>***

***<property name="connection.user">***

***scott***

***</property>***

***<property name="connection.password">***

***tiger***

***</property>***

***<property name="dialect">org.hibernate.dialect.OracleDialect</property>***

***<property name="show\_sql">true</property>***

***<mapping class="Product"/>***

***</session-factory>***

***</hibernate-configuration>***

***@@@@IMPORTANT to NOTE***

***Main step:Create Annotation Configuration Object instead of Configuration Object in order to process POJO Class Annotations.***

***Configuration cfg=new AnnotationConfiguration();***

***cfg.configure();***

*import org.hibernate.\*;*

*import org.hibernate.cfg.\*;*

*import java.util.\*;*

*public class ClientApp*

*{*

*public static void main(String[] args)throws Exception*

*{*

*Configuration cfg=new AnnotationConfiguration();*

*cfg.configure();*

*SessionFactory sf=cfg.buildSessionFactory();*

*Session s=sf.openSession();*

*Product p=new Product();*

*p.setPid("xyz111");*

*p.setPname("ddd");*

*p.setPcost(800);*

*Transaction tx=s.beginTransaction();*

*s.save(p);*

*tx.commit();*

*System.out.println("Product Insertion Success");*

*s.close();*

*}*

*}*