Practical 8 – Creating a JPA Application using ORM associations and a Hibernate application.

a) Develop a JPA Application to demonstrate use of ORM associations.

Step 1-

Create a table book(bookid int primary key auto_increment, bookname varchar(50), author varchar(100), price int);

Step 2 –

Create a new project.

Step 3 –

Create persistent unit.

Select a new data source. Create JNDI.

Step 4 –

Create class entity for database.

Select the table book.

Step 5 – Create JSP

Books.jsp

```
<%@page import="java.util.Iterator"%>
<%@page import="javax.persistence.Persistence"%>
<%@page import="tyit.Book"%>
<%@page import="java.util.List"%>
<%@page import=" javax.persistence.EntityTransaction"%>
<%@page import=" javax.persistence.EntityManager"%>
<%@page import=" javax.persistence.EntityManagerFactory"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
<head>
<title>JSP Page</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
Book Details <hr><br><br>
<form>
Book Name    <input type="text" name="bn"><br>
Author Name <input type="text" name="au"><br>
Price            <input type="text" name="pr"><br>
<br><input type="submit" name=”ab” value="ADD BOOK">
</form>
<hr>
<%!
        private EntityManagerFactory emf;
        private EntityManager em;
        private EntityTransaction tx;
        private List<Book> allbooks;
%>
```

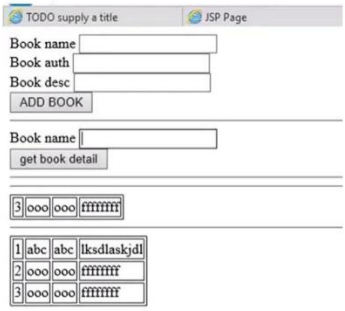
```
<%
   if(request.getParameter("ab")!=null)
   {
        emf = Persistence.createEntityManagerFactory("BookPU");
        em=emf.createEntityManager();
        tx = em.getTransaction();
      Book bk = new Book();
      bk.setBookname(request.getParameter("bn"));
      bk.setAuthor(request.getParameter("au"));
      bk.setPrice(Integer.parseInt(request.getParameter("pr")));
      tx.begin();
      em.persist(bk);
      tx.commit();
      em.close();
        }
        emf = Persistence.createEntityManagerFactory("BookPU");
        em = emf.createEntityManager();
        allbooks = em.createQuery("SELECT b FROM Book b").getResultList();
        Iterator it = allbooks.iterator();
        out.print("<table border>");
        while (it.hasNext())
      {
        Book bk=(Book)it.next();

        out.print("<tr><td>"+bk.getBookid()+"</td><td>"+bk.getBookname()+"</td>
        <td>"+bk.getAuthor()+"</td><td>"+(bk.getPrice()));
      }
      out.print("</table>");
      em.close();
   %>
</body>
</html>
```

glassfish-resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application
Server 3.1 Resource Definitions//EN" "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <jdbc-connection-pool allow-non-component-callers="false" associate-with-
thread="false" connection-creation-retry-attempts="0" connection-creation-retry-interval-in-
seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-seconds="0"
connection-validation-method="auto-commit" datasource-
classname="com.mysql.cj.jdbc.MysqlDataSource" fail-all-connections="false" idle-timeout-
in-seconds="300" is-connection-validation-required="false" is-isolation-level-
guaranteed="true" lazy-connection-association="false" lazy-connection-enlistment="false"
match-connections="false" max-connection-usage-count="0" max-pool-size="32" max-wait-
time-in-millis="60000" name="mysql_logindb_rootPool" non-transactional-
connections="false" pool-resize-quantity="2" res-type="javax.sql.DataSource" statement-
```

```
timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-seconds="0"
wrap-jdbc-objects="false">
        <property name="serverName" value="localhost"/>
        <property name="portNumber" value="3306"/>
        <property name="databaseName" value="logindb"/>
        <property name="User" value="root"/>
        <property name="Password" value="tiger"/>
        <property name="URL" value="jdbc:mysql://localhost:3306/logindb"/>
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
    </jdbc-connection-pool>
    </resources>
```

Output:



Note: Instead of Book desc put price

b) **Develop a Hibernate application to store Feedback of Website Visitor in MySQL Database.**

Step 1: MySql Command:-
Select Services -> right click on database -> connect -> password -> ok ->again right click on database -> create database -> db -> ok.
Expand db -> Select and right click table -> click on Execute command ->
Create table guestbook (no int primary key auto_increment, name varchar(20), msg varchar(100), dt varchar(40));
Step 2: Create a Hibernate Project :-
File -> New Project -> Java Web -> Web application - > Next -> give the project name -> browse the location as required -> select the checkbox – "dedicated folder for storing libraries" -> Next
Select glassfish server -> next
Select frame work - hibernate -> select the respective database connection -> finish.
Step 3: Adding Reverse Engineering File :-
Right click on Project -> new -> other -> select Hibernate -> Hibernate Reverse Engineering wizard file type -> next -> file name (hibernate.reveng) , folder -> click on browse and select src->java -> next -> select guestbook table name from the available tables option -> click add (select the checkbox – include related files) -> finish.
Step 4: Adding Hibernate mapping files and POJOs from Database file type:-
Right click on Project -> new -> other -> select Hibernate -> Hibernate mapping files and POJOs from Database file type) -> next -> keep the default configuration file name file name (hibernate.cfg) and Hibernate Reverse Engineering File (hibernate.reveng) -> type the package name (hibernate) -> finish.
Step 5: Creating JSP File :-
Right click on project -> new -> JSP -> filename -> guestbookview -> select radiobutton -> JSP file (Standard syntax) -> Finish.
File name - Guestbook.java

```
package hibernate;
public class Guestbook implements java.io.Serializable {
private Integer no;
private String name;
private String msg;
private String dt;
public Guestbook() {
}
public Guestbook(String name, String msg, String dt) {
this.name = name;
this.msg = msg;
this.dt = dt;
}
public Integer getNo() {
return this.no;
}
public void setNo(Integer no) {
this.no = no;
}
public String getName() {
return this.name;
}
public void setName(String name) {
this.name = name;
}
public String getMsg() {
return this.msg;
}
public void setMsg(String msg) {
```

```
this.msg = msg;
}
public String getDt() {
return this.dt;
}
public void setDt(String dt) {
this.dt = dt;
}
}
```

File name - hibernate.cfg.xml

```
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/db</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">tiger</property>
<mapping resource="hibernate/Guestbook.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

File name – hibernate.reveng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
  <schema-selection match-catalog="logindb"/>
  <table-filter match-name="guestbook"/>
</hibernate-reverse-engineering>
```

File name - Guestbook.hbm.xml

```
<hibernate-mapping>
<class name="hibernate.Guestbook" table="guestbook" catalog="db">
<id name="no" type="java.lang.Integer">
<column name="no" />
<generator class="identity" />
</id>
<property name="name" type="string">
<column name="name" length="20" />
</property>
<property name="msg" type="string">
<column name="msg" length="100" />
</property>
<property name="dt" type="string">
<column name="dt" length="40" />
</property>
</class>
</hibernate-mapping>
```

File name - index.jsp

```
<html>
<head>
<title>Guest Book</title>
</head>
<body>
Guest Book <hr><br><br>
<form action="guestbookview.jsp" >
Name <input type="text" name="name" maxlength="20"><br>
```

```
Message <textarea rows="5" cols="40" maxlength="100" name="msg"></textarea>
<br><input type="submit" value="submit">
</form>
</body>
</html>
```

File name - guestbookview.jsp

```
<%@page import="org.hibernate.SessionFactory"%>
<%@page import="org.hibernate.Session"%>
<%@page import="org.hibernate.cfg.Configuration"%>
<%@page import="org.hibernate.Transaction"%>
<%@page import="java.util.List"%>
<%@page import="java.util.Iterator"%>
<%@page import="hibernate.Guestbook"%>
<%!
SessionFactory sf;
org.hibernate.Session ss;
List<hibernate.Guestbook> gbook;
%>
<%
sf = new Configuration().configure().buildSessionFactory();
ss= sf.openSession();
Transaction tx=null;
Guestbook gb=new Guestbook();
try
{
tx=ss.beginTransaction();
String name=request.getParameter("name");
String msg=request.getParameter("msg");
String dt=new java.util.Date().toString();
gb.setName(name);
gb.setMsg(msg);
gb.setDt(dt);
ss.save(gb);
tx.commit();
}
catch(Exception e){ out.println("Error"+e.getMessage()); }
try
{ ss.beginTransaction();
gbook=ss.createQuery("from Guestbook").list();
}
catch(Exception e){ }
%>
<html>
<head>
<title>Guest View</title>
</head>
<body>
Guest View
Click here to go <a href="index.jsp"> BACK </a>
<br><br>
<% Iterator it=gbook.iterator();
while(it.hasNext())
{
Guestbook eachrecord=(Guestbook)it.next();
out.print(eachrecord.getDt()+" ");
```

```
out.print(eachrecord.getName()+"<br>");
out.print(eachrecord.getMsg()+"<br><hr>");
}
%>
</body>
</html>
```

Output :