Procedure to create a jsp file(servlet) with mysql database connection using Apache tomcat 10.1.11 server

1 seelct new >new>dynamic webproject

2 put the cursor on the main folder right click >new>html file type the file name(html file name)

and click finish button

3 create the html file frontend

Ex:
```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action=”sample.jsp” method=”post” enctype=” ”>
 Enter the employee id:<input type=”text” name=”empid”/>
 Enter the employee name:<input type=”text” name=”ename”/>
 Enter the employee eaddress:<input type=”text” name=”eadress”/>
 <input type="submit" value="Submit">
</form>
```

```
</body>
</html>
```

4 then save it and down load the server, setup into your mechine

Download the following sever from apache website version of apache 10.1.11

- o 32-bit/64-bit Windows Service Installer (pgp, sha512)

5 add this server to the eclipse ide

Select help>newsoftware installation select the following option from text box

2023-06 - https://download.eclipse.org/releases/2023-06

Then select > Web, XML, Java EE and OSGi Enterprise Development

Click finish and setup the port number without clash

6 set the java build path

Put the cursor on webapp folder click the right button select the build path and configure the build path add the servlet-api-jar files from apache tomcat source folder

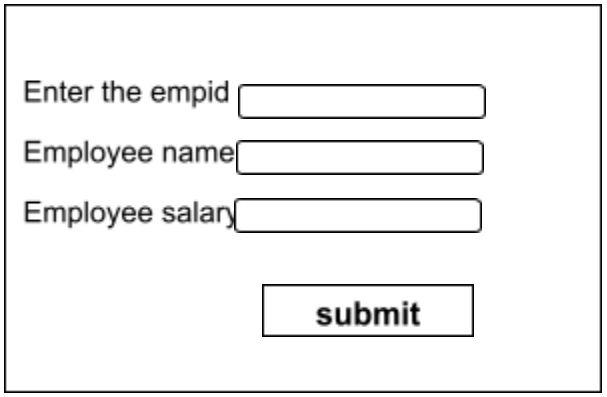
7 Write the jsp file

```
<% @page import="java.sql.*", language="java" contentType="text/html; charset=ISO-8859-1" %>
 <!DOCTYPE html>
      <html>
      <head>
      <meta charset="ISO-8859-1">
      <title>Insert title here</title>
      </head>
      <body>
      <%
```

```
  try{
  int sno=Integer.parseInt(request.getParameter("sno"));
  String sname=request.getParameter("sname");
  String address=request.getParameter("address");
  Class.forName("com.mysql.jdbc.Driver");Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/ramu","root","amar");
 Statement smt=con.createStatement();
 String s2="insert into kokila values(?,?,?)";
 PreparedStatement ps=con.prepareStatement(s2);
 ps.setInt(1,sno);
 ps.setString(2,sname);
 ps.setString(3,address);
 ps.execute();
 String s1="select * from kokila";
 ResultSet rs=smt.executeQuery(s1);
 out.println("INSERTION");
 while(rs.next()){
  System.out.println(rs.getInt("sno")+" "+rs.getString("sname")+"
"+rs.getString("address"));
 }
}catch(Exception e){
  System.out.println(e);
 }
%>
</body>
</html>
```

8 save the file and run in server it reads the data from fronted and insert into database

Ex 2 Ddevelop the java application with frontend **jsp file and**
To **store the data into backend database using MySQL workbench**



Hibernate

Hibernate is an open source object relational mapping (ORM) tool that provides a framework to map object-oriented domain models to relational databases for web applications. Hibernate was started in 2001 by Gavin King with colleagues from Cirrus Technologies as an alternative to using EJB2-style entity beans. The original goal was to offer better persistence capabilities than those offered by EJB2.

In early 2003, the Hibernate development team began Hibernate2 releases. In 2005, Hibernate version 3.0 was released. In December 2018, Hibernate ORM 5.4.0 Final was released

The JDBC API provides classes, such as the ResultSet and the PreparedStatement, which allows the tabular data to be consumed and transformed into components that are compatible with Java's object-oriented constructs. However, this transformation process from JDBC results to object-oriented components requires a great deal of boiler-plate code, and it is tedious, cumbersome and error-prone.

```
1  import java.sql.*;
2
3  public class AddGameSummary {
4      public static void main(String[] args) throws Exception {
5
6          String dburl = "jdbc:mysql://localhost:3306/rps";
7          Class.forName("com.mysql.jdbc.Driver");
8          Connection con =
9                  DriverManager.getConnection(dburl, "name", "pass");
10
11         String INSERT_SQL =
12             "INSERT  INTO  GS  (cc,  sc,  r,  d)  " +
13             "VALUES  (?,  ?,  ?)";
14         PreparedStatement ps = con.prepareStatement(INSERT_SQL);
15         ps.setString(1, "rock");
16         ps.setString(2, "paper");
17         ps.setString(3, "win");
18         ps.executeUpdate();
19         ps.close();
20     }
21 }
```

Hibernate API then lets developers decorate their Java code with minimally obtrusive JPA annotations. Said annotations describe how a given Java class and its variables map to a given table and its columns within a database. The Hibernate framework then takes care of all the tedious and error-prone coding required to convert between the object-oriented Java code and the backend database.

```
@Entity
public class Tent {

  private Long id;
  private String title;
  private Date date;

  @Id
  @GeneratedValue
  public Long getId() { return id; }
  private void setId(Long id) {this.id = id;}

  public Date getDate() {return date;}
  public void setDate(Date date) {
    this.date = date;
  }

  public String getTitle() {return title;}
  public void setTitle(String title) {
    this.title = title;
  }

}
```

Procedure to create maven project in hibernate

Step 1: Open Google Type the Spring tool eclipse ide for spring tool

(Spring tool 4 for visual studio code)

Select 4.18.0window 64 download version ,setup into mechine

Step 2: download mysql workbench 8.0 database software

from mysql community server website

(https://dev.mysql.com/downloads/) and setup mysql into mechine

Step 3: open the spring toolsuite4.exe for creating maven project

File>new>others>maven>next> sample project/

maven-architype-quickstart

give the =>Groupid:com.klu

=> Artifact:projectname Click finish

Step 4: open the pom.xml from project folder

Double click on pom.xml file

Add the required <dependency> source code in between <dependencies> tag from websource

Ex: <dependencies>
<dependency>

```
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<version>8.0.33</version>
</dependency>
</dependencies>
```

Add the all following decencies from maven repository

Using websource "Maven Repository: Search/Browse/Explore (mvnrepository.com)" source

1) **Java Transaction API dependency**
2) **Hibernate ORM Hibernate Core Dependency**
3) **MySQL Connector Java dependency**
4) **JAXB Runtime dependency**

After added the dependency update the project

(Put the cursor &right click) **Project>maven>update project>***select* **project>***tick the* **force update of snapshot>***click* **ok**

Step 5: create the hibernate.cfg.xml

Put the cursor on <src/main/java>right click >new>other>

select hibernate configuration file it will be added to your project

open the configaration file add the following code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory name="sf1">
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password"><database name></property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/amar</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
         <property name="hbm2ddl.auto">update</property>
           <mapping class="project.classname" />
    </session-factory>
</hibernate-configuration>
```

Step 6: Create the class program in java (ex:studentklu.java)

```
package sunny;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Studentklu {
 @Id
        int a;
         String b;
           public int getA() {
                    return a;
                    }
          public void setA(int a) {
                    this.a = a;
                    }
                    public String getB() {
                            return b;
                    }
                    public void setB(String string) {
                            this.b = string;
                            }
                    @Override
                    public String toString() {
                            return "Studentklu [a=" + a + ", b=" + b + "]";
                        }
                             }
```

Step 7 : Create the main java program (ex:mainpr.java)

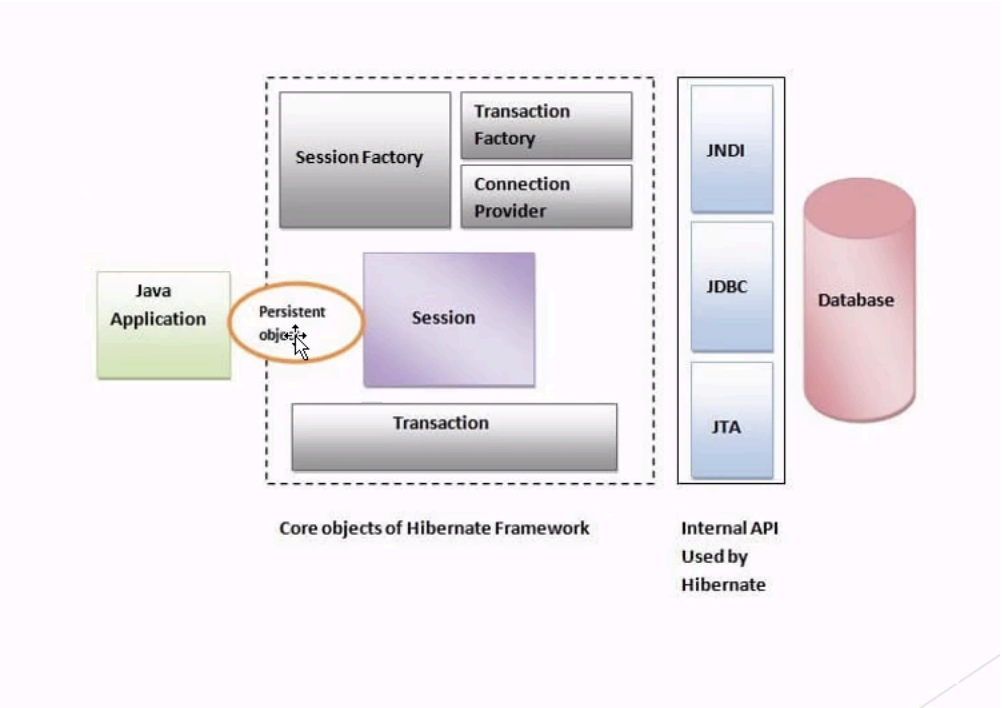
```
package sunny;
mport org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
public class Main_prog
{

public static void main(String[] args)
{
try {

// Create Configuration
Configuration configuration = new Configuration();
configuration.configure("hibernate.cfg.xml");
configuration.addAnnotatedClass(Studentklu.class);
 // Create Session Factory
org.hibernate.SessionFactory sessionFactory = configuration.buildSessionFactory();
// Initialize Session Object
Session session = sessionFactory.openSession();
Transaction t;
 Studentklu song1 = new Studentklu();
 song1.setA(1);
 song1.setB("rose");
  song1.setA(2);
  song1.setB("lilly");
  session.beginTransaction();
  // Here we have used
```

```
// save() method of JPA
session.save(song1);
session.getTransaction().commit();
}finally{
 }
    }
     }
```

Step 8: **Open the mysql workbench data base software for storing the data.**

Step 9: **Run the main java program the result will be seen in database table**

Step 10: **exit from ide**



Java application :contains entity ex:employe

Session transaction under the session only which will do **commit and rollback** operation

Session object will be used to call the API like save,update,rollback

After started session factory all these things were done

Transaction factory is an optional

Session factory is an mandatory

Conection provider provides the connection services

Jndi –java naming dictionary interface

Jdbc –java data base connectivity

Jta –java transaction Api

6 Create from new Hibernate.cfg.xml

Hibernate session factory

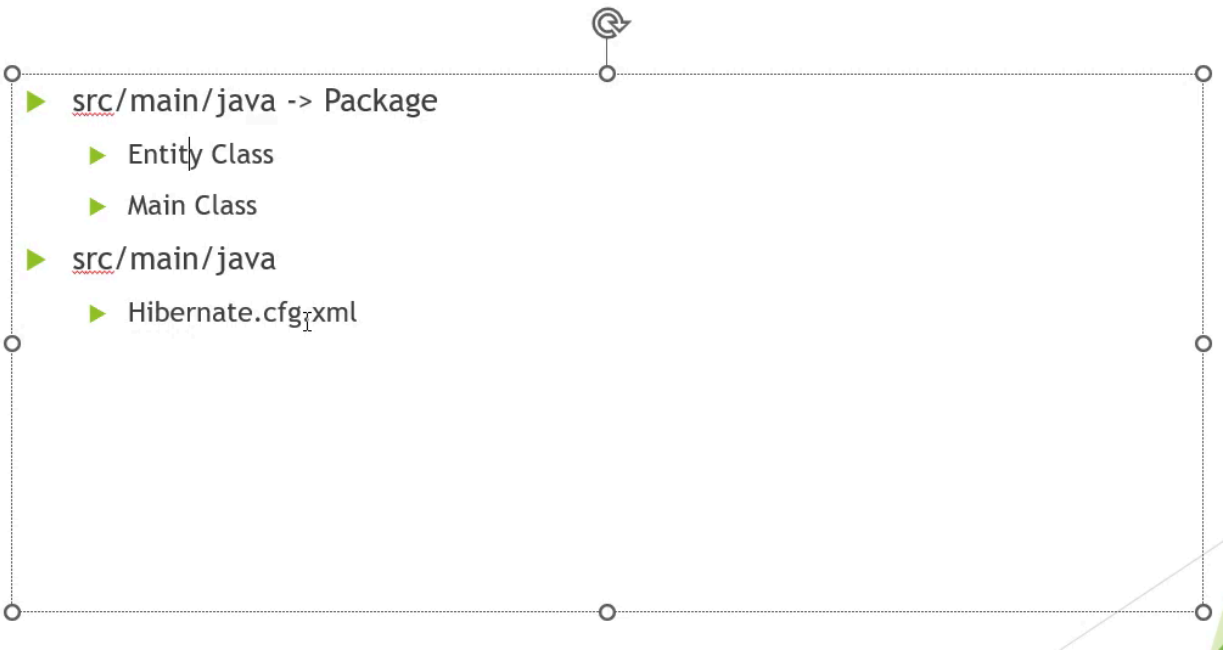
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>

    <property name="hbm2ddl.auto">update</property>
    <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/jfsd</property>
    <property name="connection.username">root</property>
    <property name="connection.password">root</property>
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>

    <mapping class="com.klu.hibernate.Employee"/>

</session-factory>
</hibernate-configuration>
```

Name=Hbm2.ddl.auto will reuse the same data/table /database to perform different tasks

Dialect: dialect will use to write the query for api which were calling (hibernate take care of all querries)

Whatever request asked by api that corresponding querry will be write by dialect

Files Required



7 In src/main/java >create hibernate.cfg.xml file