**JAVA FSD Phase 2**

**Day 8th (3 Nov 2022)**

Self Learning: SQL Training

Phase 2:

* JDBC
* Hibernate Framework

**SQL** (Structured Query Language)

**Query:** Statements that you write to interact with your database

**RDBMS:** Relational Database Management System

MySQL, MS Access, Oracle, Sybase

Create/Form relations between different entities/tables

**SQL Commands**

**DDL: Data Definition Language**

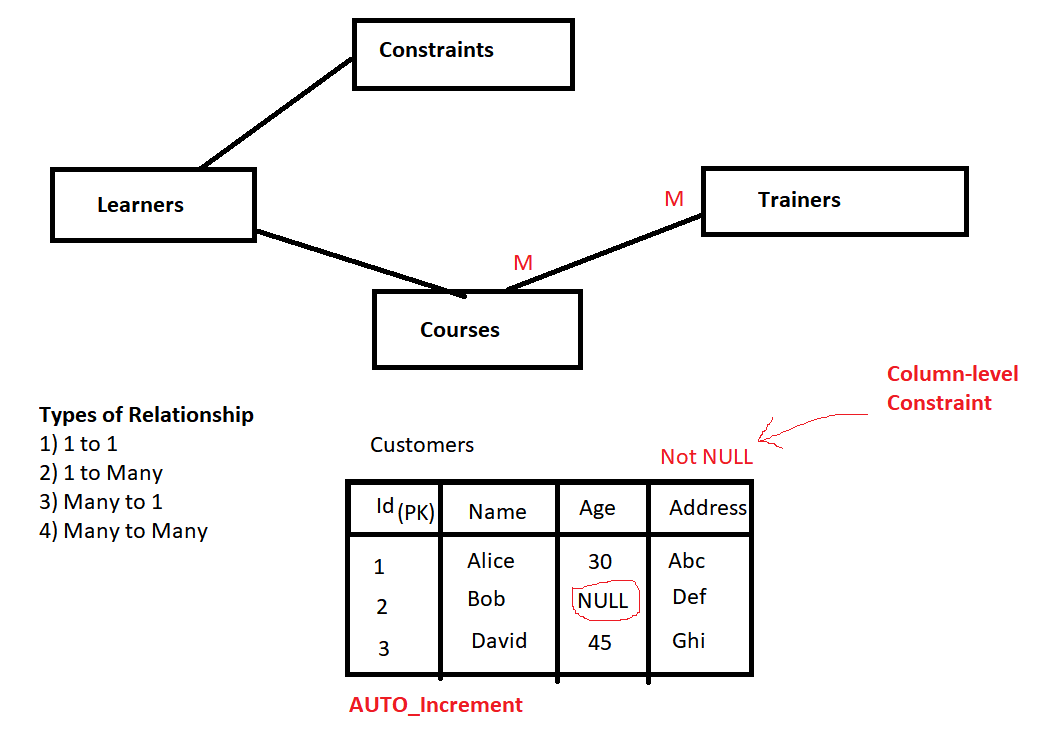
1. Create
2. Alter
3. Drop

**DML: Data Manipulation Language**

1. Select
2. Insert
3. Update
4. Delete

**DCL: Data Control Lang**

1. Grant: Give privilege to user
2. Revoke: Take back privilege from the user



**Syntax to write SQL Queries**

**DATABASE**

**Create Database**

CREATE DATABASE database\_name;

**Drop Database**

DROP DATABASE database\_name

**Use Database**

USE database\_name;

**TABLE**

1. View the records from the table

SELECT column1, column2, column3

FROM table\_name;

SELECT Id, name, age

FROM Customers;

SELECT \*

FROM Customers;

*\* indicates ALL*

SELECT \*

FROM Customers

WHERE age < 30;

*Where clause is used for specifying the condition*

*<,>, <=, >= !=, ==*

SELECT \*

FROM Customers

WHERE age < 30 AND|OR salary > 1200

SELECT \*

FROM Customers

WHERE name LIKE ‘A%’ % represents one or many characters

Patters used to match the String value using LIKE Operator

1. **Insert into the table**

INSERT INTO table\_name(column1, column2)

VALUES (value1, value2)

INSERT INTO table\_name

VALUES(value1,value2)

1. **Delete from the table**

DELETE FROM table\_name

DELETE FROM table\_name

WHERE age <30;

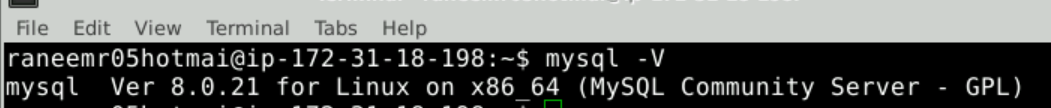
1. **Update from the table**

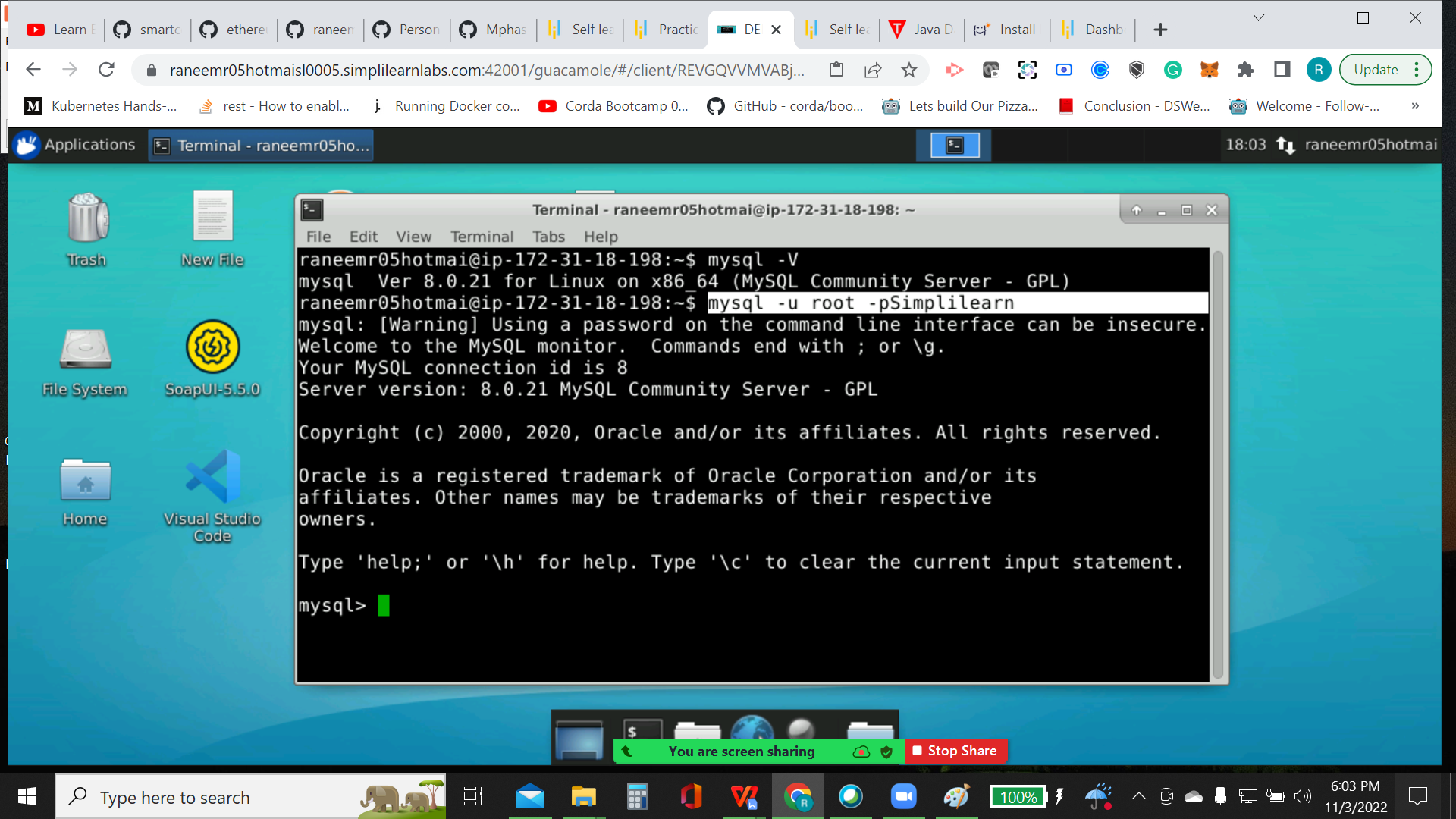
UPDATE table\_name

SET column1=value1, column2= value2

WHERE CONDITION id=1;

**Lab Environment**





**Show list of databases**

Show databases;

**Create database**

create database demo;

**Use Database**

Use demo

**Show Tables in Database**

Show tables;

**Create Table**

Create table employee(id int, name nvarchar(20), age int NOT NULL);

**JDBC**

* Establish connection with the DB
* Insert record to the DB
* View/Retrieve record from the DB
* Delete record
* Update record

Download the mysql connector jar file

* Download the jar file from <https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.21/mysql-connector-java-8.0.21.jar>

**In Eclipse**

* Create New Project
* Right click on the project folder --> Build Path --> Configure Build Path --> In Libraries tab --> click on Add External JARs and upload your jar file here
* Click on Apply and Close
* You can see jar file in your Referenced Libraries on the left

**Create JDBCDemo.java class (Establish Connection)**

package com.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBDemo {

public static final String DRIVER\_CLASS = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/demo";

public static final String USER = "root";

public static final String PASSWORD = "Simplilearn";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

System.out.println("Connecting to the database....");

//Register the Driver

Class.forName(DRIVER\_CLASS);

//Connect with the DB

// we need URL, Username and password

Connection con = DriverManager.getConnection(DB\_URL, USER, PASSWORD);

System.out.println("Connection established with the DB");

}

}

**DAY 9 (4 Nov)**

**CRUD operations**

**INSERT**

package com.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class DBDemo {

public static final String DRIVER\_CLASS = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/demo";

public static final String USER = "root";

public static final String PASSWORD = "Simplilearn";

public static final String QUERY ="Select \* from employee";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

System.out.println("Connecting to the database....");

//Register the Driver

Class.forName(DRIVER\_CLASS);

//Connect with the DB

// we need URL, Username and password

Connection con = DriverManager.getConnection(DB\_URL, USER, PASSWORD);

if(con!= null)

{

System.out.println("Connection established");

}

else

{

System.out.println("Connection not established");

}

Statement stmt = con.createStatement();

//Insert record

String insert = "INSERT INTO employee VALUES (10,'Scott',35)";

stmt.executeUpdate(insert);

System.out.println("Record inserted successfully");

}

}

**View (SELECT Statement)**

package com.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class DBDemo {

public static final String DRIVER\_CLASS = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/demo";

public static final String USER = "root";

public static final String PASSWORD = "Simplilearn";

public static final String QUERY ="Select \* from employee";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

System.out.println("Connecting to the database....");

//Register the Driver

Class.forName(DRIVER\_CLASS);

//Connect with the DB

// we need URL, Username and password

Connection con = DriverManager.getConnection(DB\_URL, USER, PASSWORD);

if(con!= null)

{

System.out.println("Connection established");

}

else

{

System.out.println("Connection not established");

}

Statement stmt = con.createStatement();

//Insert record

String insert = "INSERT INTO employee VALUES (5,'Mathew',30)";

int i = stmt.executeUpdate(insert);

if(i>0)

{

System.out.println("Record inserted successfully");

}

else

{

System.out.println("Record not inserted");

}

System.out.println("Retrieving records from the DB.....");

String select = "Select \* from employee";

ResultSet rs = stmt.executeQuery(select);

while(rs.next())

{

//Display values

System.out.println("Id is " + rs.getInt("id"));

System.out.println("Name is " + rs.getString("name"));

System.out.println("Age is " + rs.getInt("age"));

}

}

}

***PreparedStatement:*** It is a type of interface which provide set of methods which helps to do the operation on table like insert, del, update. It is also called as a pre-compiled query. Performance wise is faster than Statement.

"INSERT INTO employee VALUES (5,'Mathew',30)"

You will use variables so that you can take input from the user and insert it into the DB

“INSERT INTO employee VALUES (“+id+”,”’”+name+”’”……);

Dev spending a lot of time and effort in writing such query using Statement object.

package com.db;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class DBDemo {

public static final String DRIVER\_CLASS = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/demo";

public static final String USER = "root";

public static final String PASSWORD = "Simplilearn";

public static final String QUERY ="Select \* from employee";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

System.out.println("Connecting to the database....");

//Register the Driver

Class.forName(DRIVER\_CLASS);

//Connect with the DB

// we need URL, Username and password

Connection con = DriverManager.getConnection(DB\_URL, USER, PASSWORD);

if(con!= null)

{

System.out.println("Connection established");

}

else

{

System.out.println("Connection not established");

}

Statement stmt = con.createStatement();

//Insert record

String insert = "INSERT INTO employee VALUES (5,'Mathew',30)";

int i = stmt.executeUpdate(insert);

if(i>0)

{

System.out.println("Record inserted successfully");

}

else

{

System.out.println("Record not inserted");

}

PreparedStatement pstmt = con.prepareStatement("INSERT INTO employee values(?,?,?)");

pstmt.setInt(1, 7);

pstmt.setString(2, "William");

pstmt.setInt(3, 44);

int j = pstmt.executeUpdate();

System.out.println(j + "records inserted using PS");

System.out.println("Retrieving records from the DB.....");

String select = "Select \* from employee";

ResultSet rs = stmt.executeQuery(select);

while(rs.next())

{

//Display values

System.out.println("Id is " + rs.getInt("id"));

System.out.println("Name is " + rs.getString("name"));

System.out.println("Age is " + rs.getInt("age"));

}

}

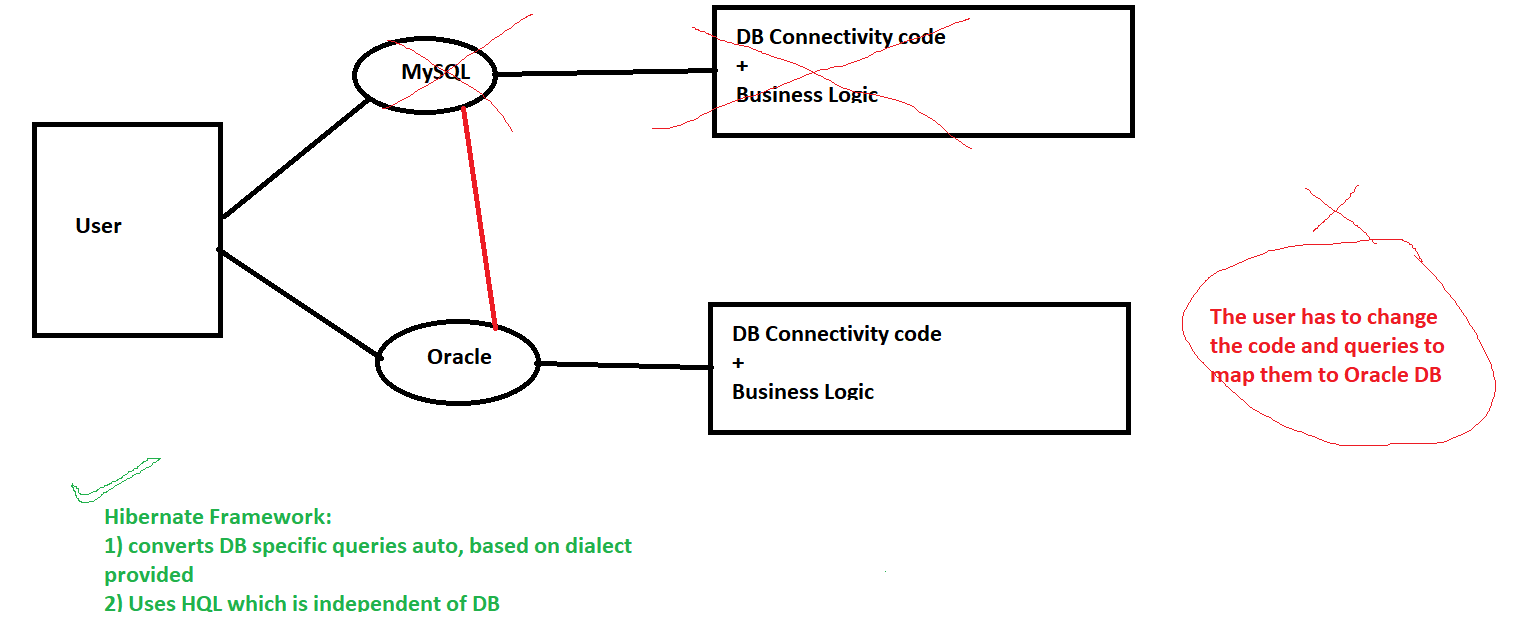
}

**JDBC TASK**

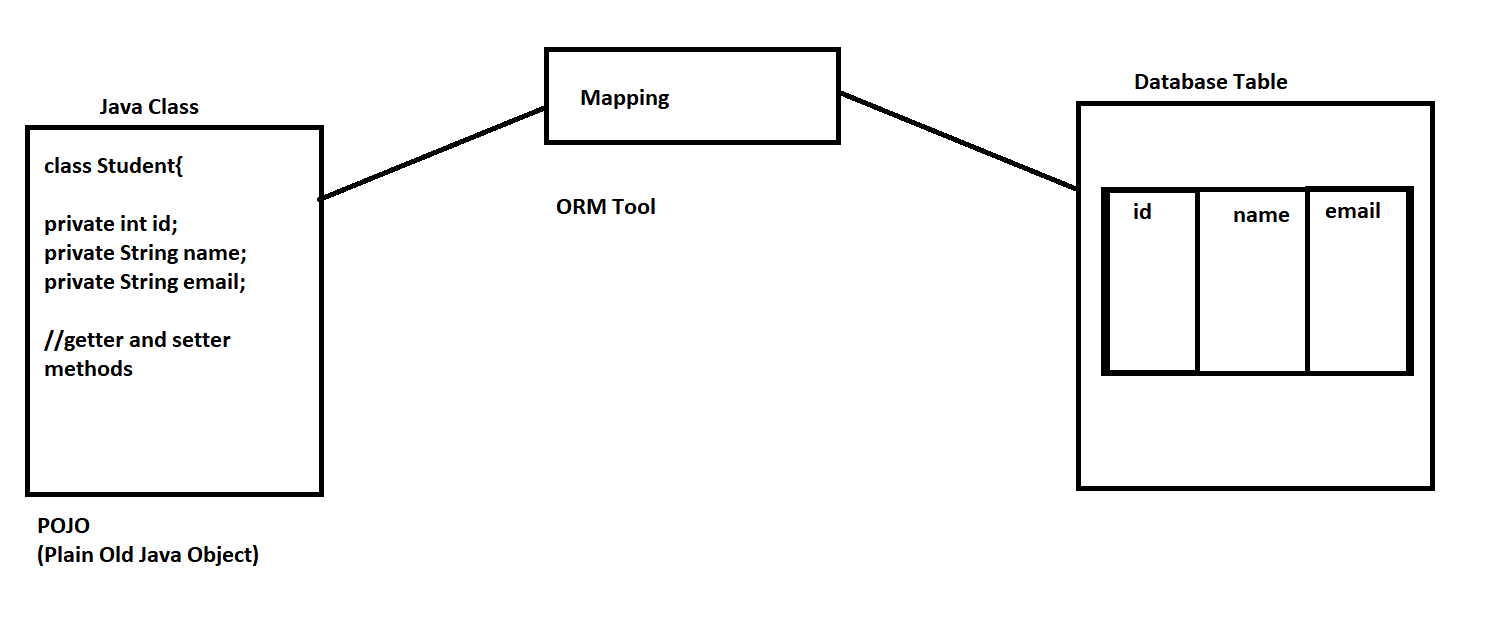
A JDBC program

1. Establish the connection
2. Create a database
3. Create a table (customers\_id, name, address, country, city, phone, email)
4. Insert 10 records (use the PS)
5. Update 2 records (update address and phone for 2 records)
6. Delete 1 record from the table (Delete record for the customer whose email is NULL)
7. View the records from the table
8. Drop the database (optional)

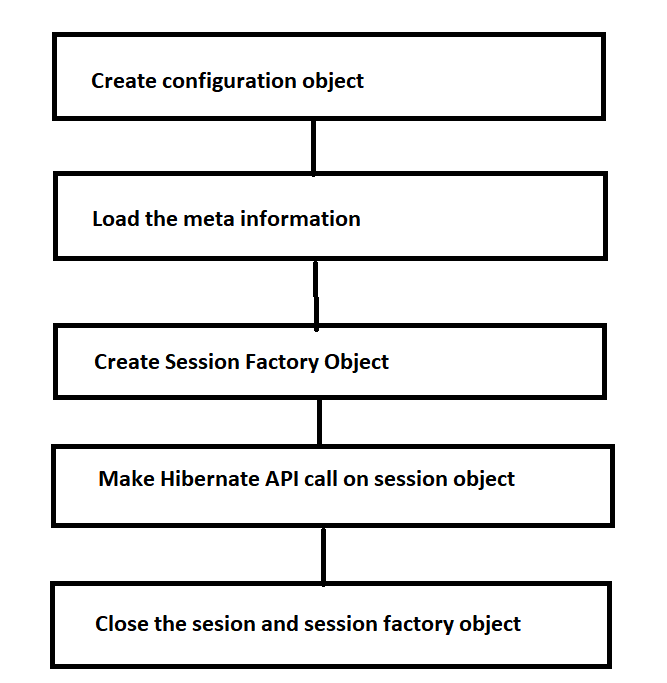
**HIBERNATE FRAMEWORK**



**ORM**



**Hibernate Programming Model**



**Setting up Project with Hibernate**

1. Create a java project and add required jars
2. Create the POJO class in src folder
3. Create hibernate.cfg.xml
4. Use RDBMS dialect (org.hibernate.dialect.MySQL8Dialect - Not Required)
5. Create the ClassName.hbm.xml (mapping file) -- > Used <mapping> tag in cfg.xml file and added annotations in Class
6. Create a class that stores and retrieve the data from the PO

**Day 10th (7th Nov 2022)**

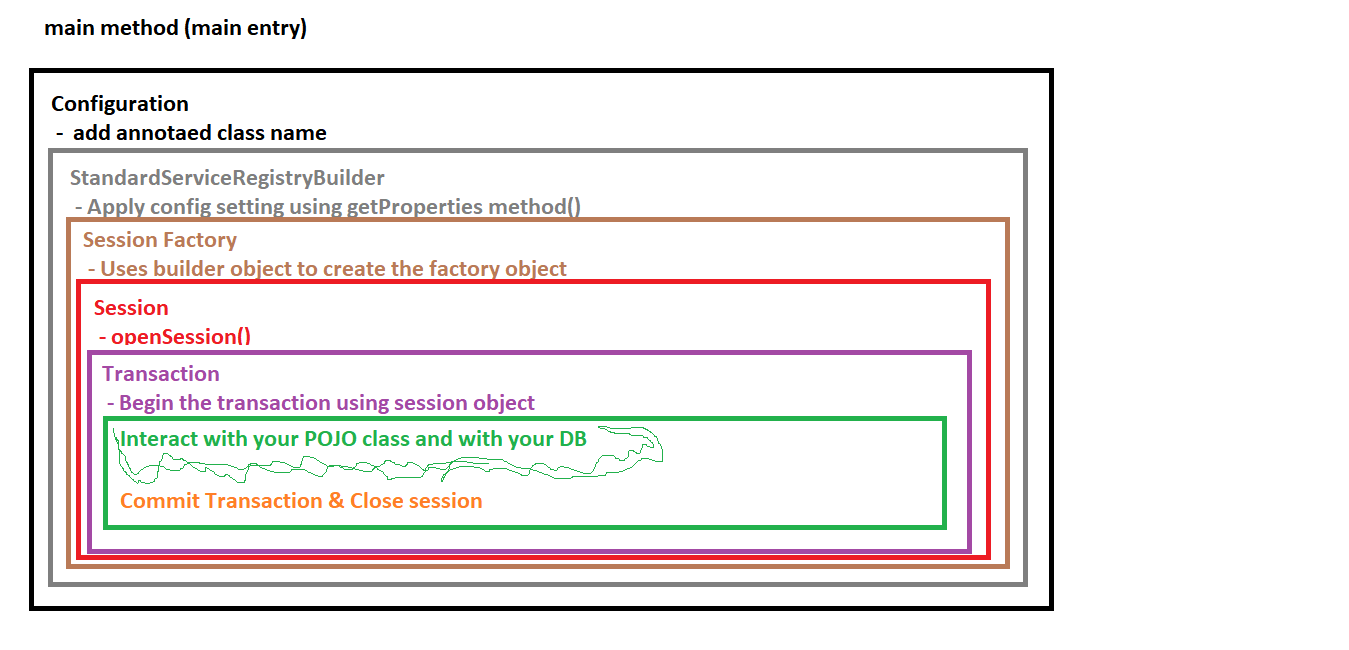
**Download hibernate folder from this URL**

<https://sourceforge.net/projects/hibernate/files/>

Add these jars into your Eclipse Project folder --> Build Path --> Configure Build Path --> Libraries--> Add External Libraries --> Select the jar files and Click Apply and Close

*hibernate-search-5.8.0.Final/dist/lib/provided/*

*hibernate-search-5.8.0.Final/dist/lib/required/*



**POJO Class**

package com.Student;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

//POJO Class - Plain Old Java Object Class

//Annotation

@Entity

@Table(name="Students")

public class Student {

@Id

private int id;

@Column(name="StudentName")

private String name;

private String email;

private int age;

Student()

{

}

public Student(int id, String name, String email, int age) {

super();

this.id = id;

this.name = name;

this.email = email;

this.age = age;

}

public Student(int id, String name)

{

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

@Override

public String toString() {

return "Student [id=" + id + ", name=" + name + ", email=" + email + ", age=" + age + "]";}

}

**Hibernate.cfg.xml**

<!--

~ Hibernate Search, full-text search for your domain model

~

~ License: GNU Lesser General Public License (LGPL), version 2.1 or later

~ See the lgpl.txt file in the root directory or <http://www.gnu.org/licenses/lgpl-2.1.html>.

-->

<hibernate-configuration>

<session-factory>

<!-- Database Settings -->

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/demo</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">Simplilearn</property>

<property name="connection.pool\_size">3</property>

<!-- SQL Dialect -->

<!-- <property name="dialect">org.hibernate.dialect.MYSQLDialect</property>-->

<property name ="current\_session\_context\_class">thread</property>

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

<property name="format\_sql">true</property>

<!-- Drop and Create table -->

<property name="hibernate.hbm2ddl.auto">update</property>

<!-- Mapping class -->

<mapping class="com.Student.Student"/>

</session-factory>

</hibernate-configuration>

**Student Main class(Main Method)**

package com.control;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import com.Student.Student;

public class StudentMain {

public static void main(String[] args) {

try

{

//1. Create configuration object

Configuration configuration = new Configuration().configure();

configuration.addAnnotatedClass(com.Student.Student.class);

//2. Create an object to get the metadata and apply it to the Main

StandardServiceRegistryBuilder builder =

new StandardServiceRegistryBuilder().applySettings(configuration.getProperties());

//3. Session Factory

SessionFactory factory = configuration.buildSessionFactory(builder.build());

//4. Start the session

Session session = factory.openSession();

//5. Begin the Transaction

Transaction transaction = session.beginTransaction();

//6. Creating POJO Class objects and interact with DB

Student s1 = new Student(101,"Alice","alice@example.com",25);

Student s2 = new Student(102,"Bob","abc@example.com", 15);

Student s3 = new Student(103,"David","david@demo.com",35);

System.out.println("Student data inserted");

session.save(s1);

session.save(s2);

session.save(s3);

//View the record from the table

Student s = session.get(Student.class, 102);

System.out.println("Details are: " + s);

//Update the record

s.setAge(30);

s.setEmail("bob@demo.com");

//Delete the record

session.delete(s);

//7. Commit the transaction

transaction.commit();

//8. Close the session

session.close();

}

catch(Exception e)

{

e.getMessage();

}

}

}

**Hibernate Task**

1. Create a new class for Customers
2. Insert 10 values to the Customers
3. Auto Generate the IDs for the customer ID

@GeneratedValue(strategy=GenerationType.IDENTITY)

private int id;

1. Display the records
2. Update 3 values
3. Delete 2 records