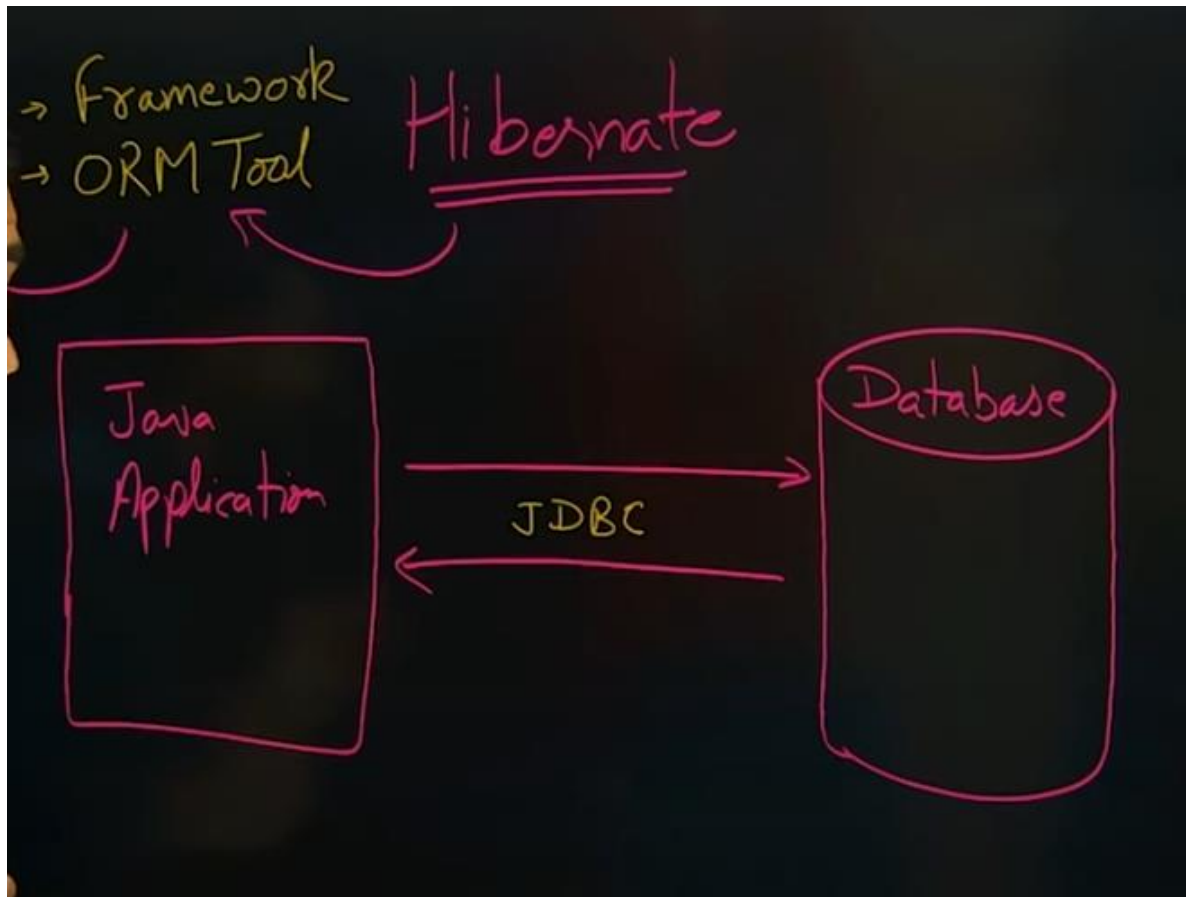
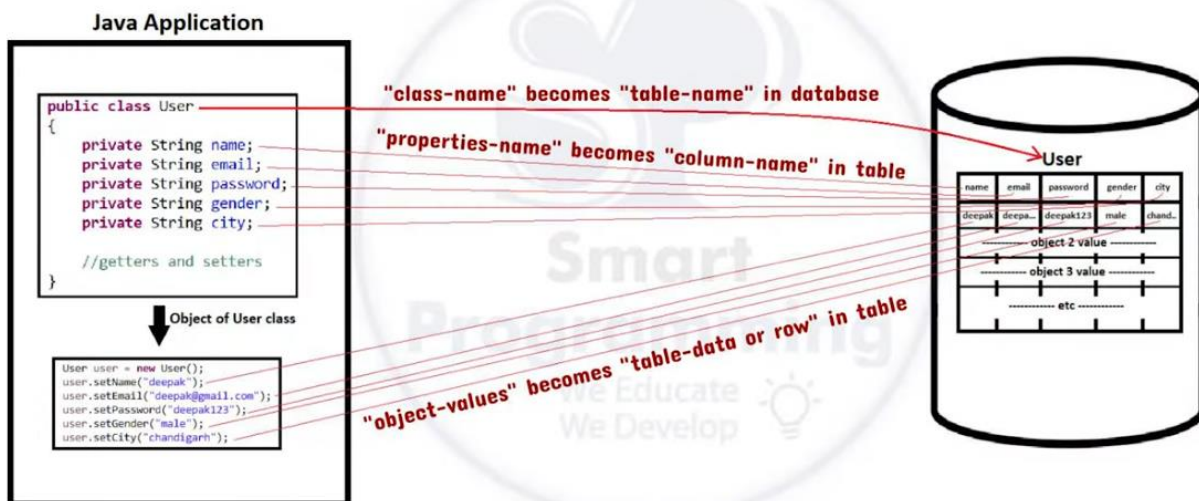


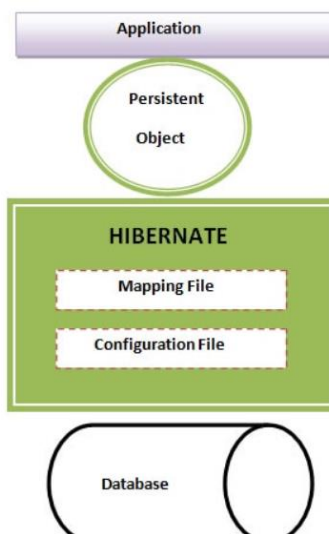
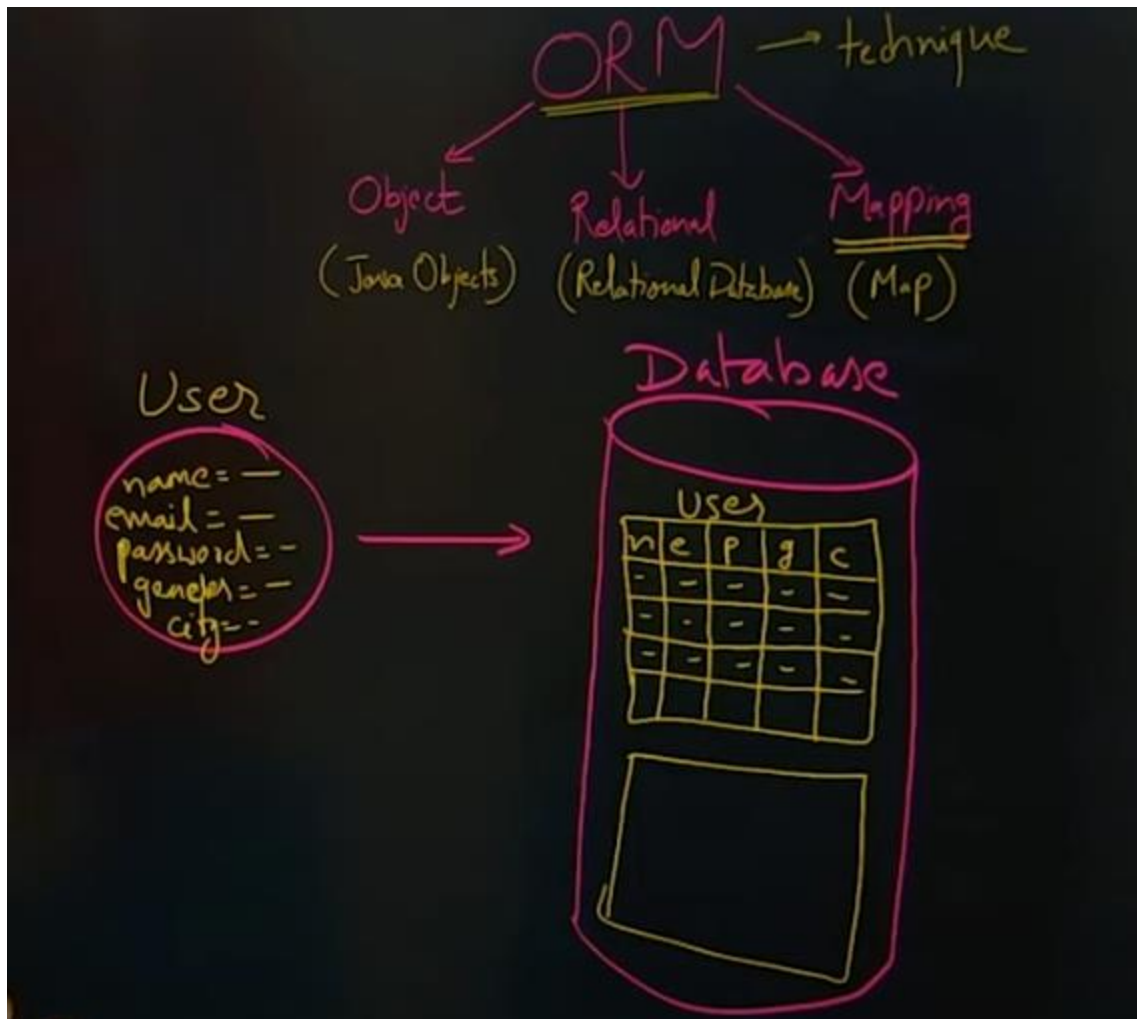
Hibernate



How Java Objects are Mapped with Database (ORM Technique)

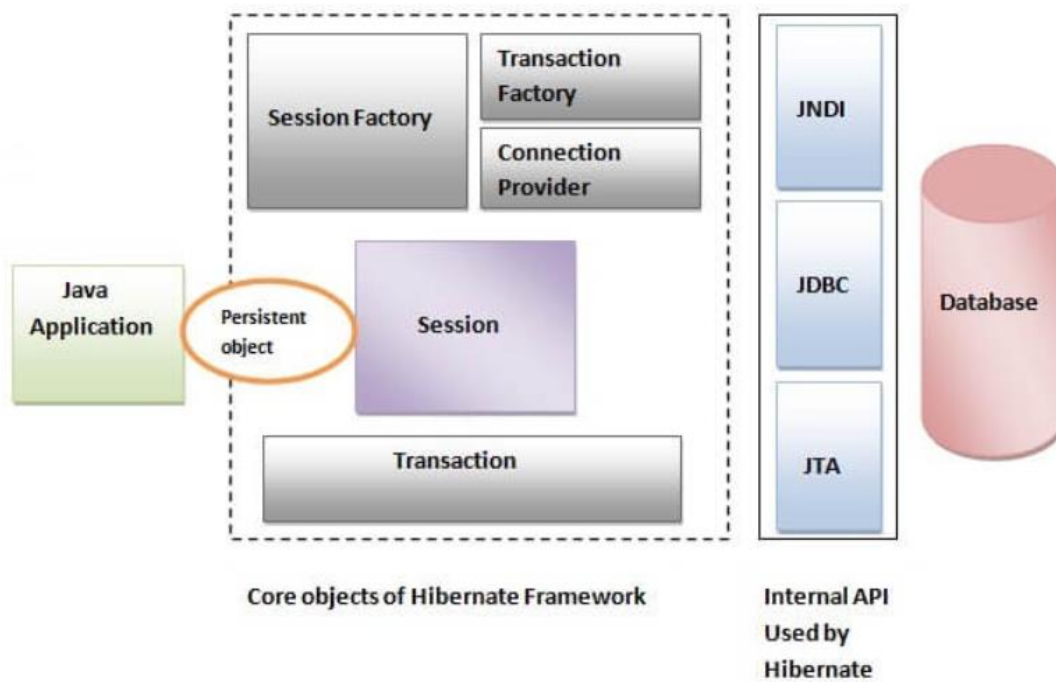


Hibernate



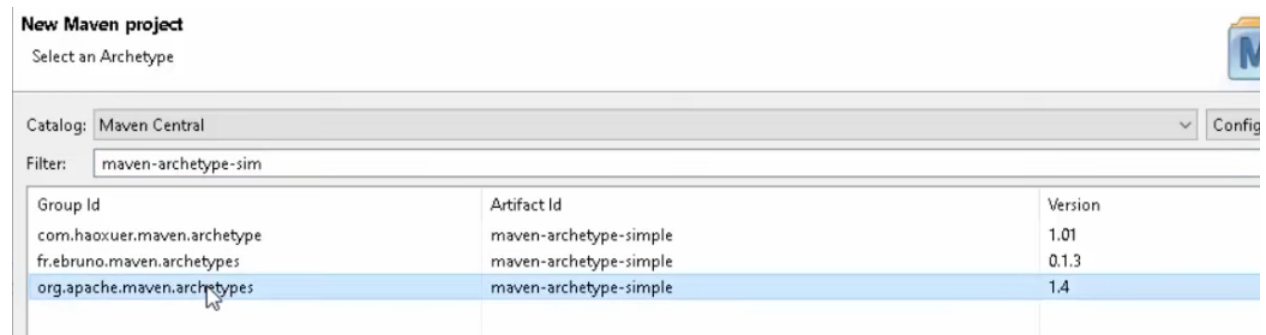
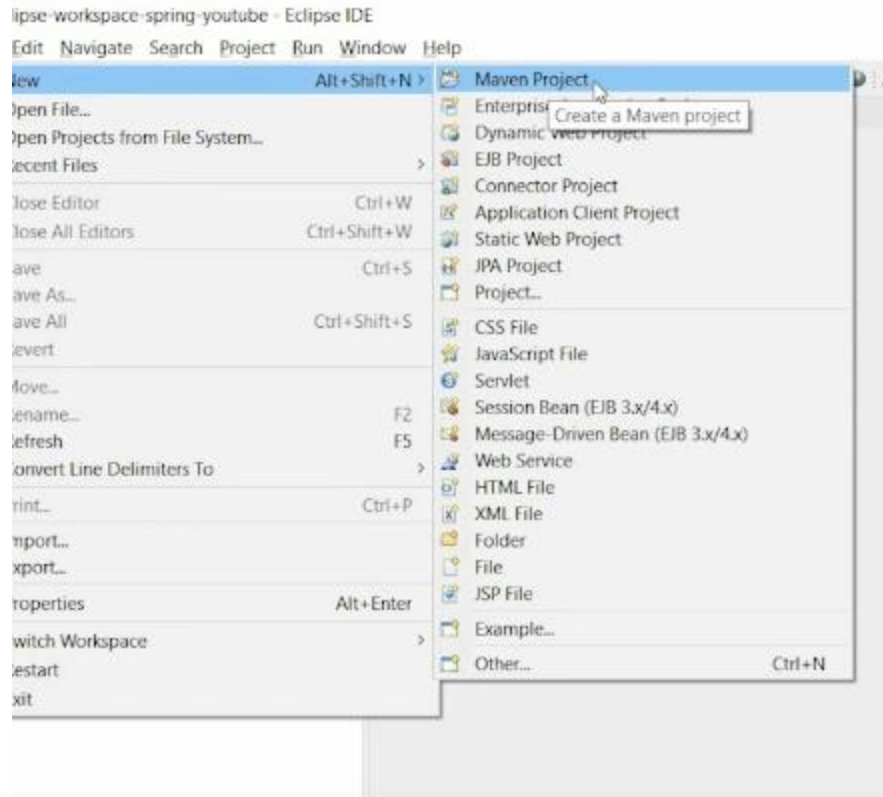
Hibernate

This is the high level architecture of Hibernate with mapping file and configuration file.

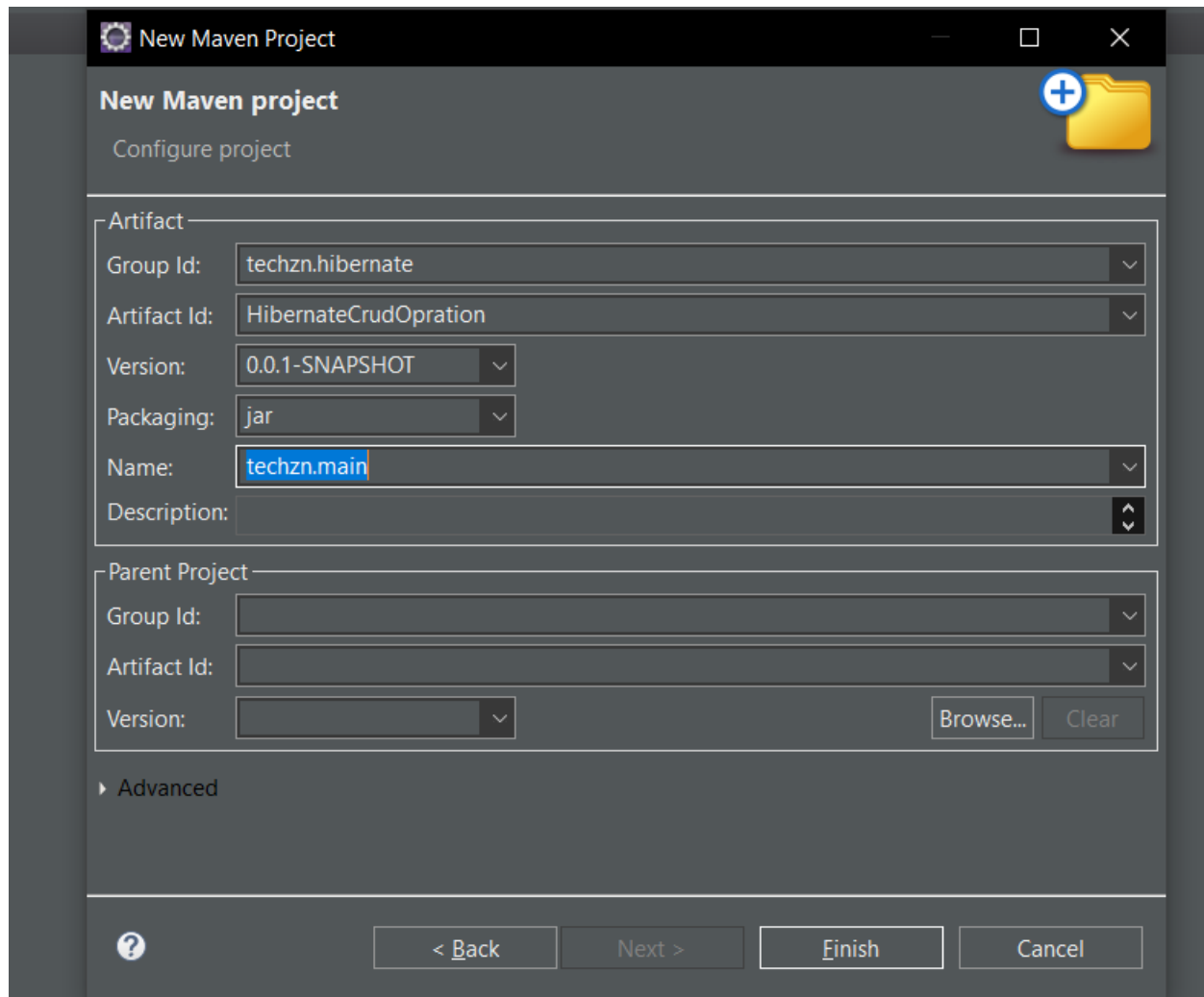


Maven Project u can use simple java project also

Hibernate



Hibernate



New Maven Project

New Maven project

Configure project

Artifact

Group Id: techzn.hibernate

Artifact Id: HibernateCrudOpration

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name: techzn.main

Description:

Parent Project

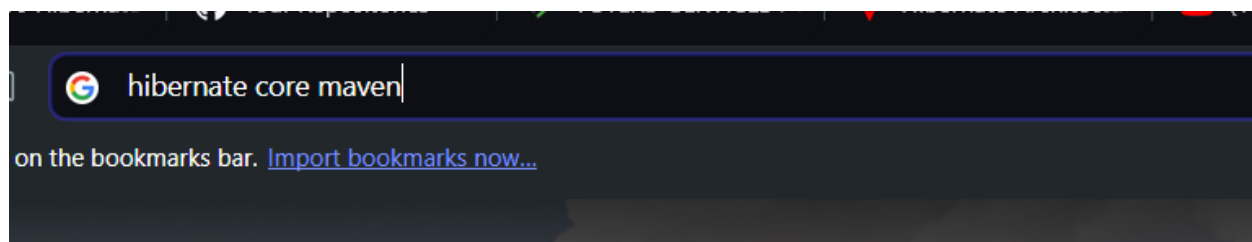
Group Id:

Artifact Id:

Version: Browse... Clear

Advanced

< Back Next > Finish Cancel



Add hibernate and mysql dependency in side pom.xml

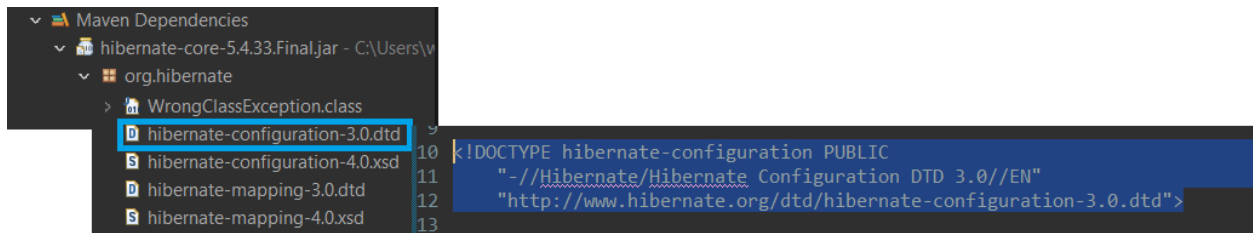
Create table name java class

Hibernate

After create config file name as

Hibernate.cnf.xml

And add this



If [configuration.cfg.xml](#) showing error goto Preference..

```
<?xml version="1.0" encoding="UTF-8"?>
```

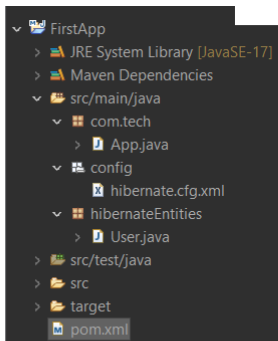
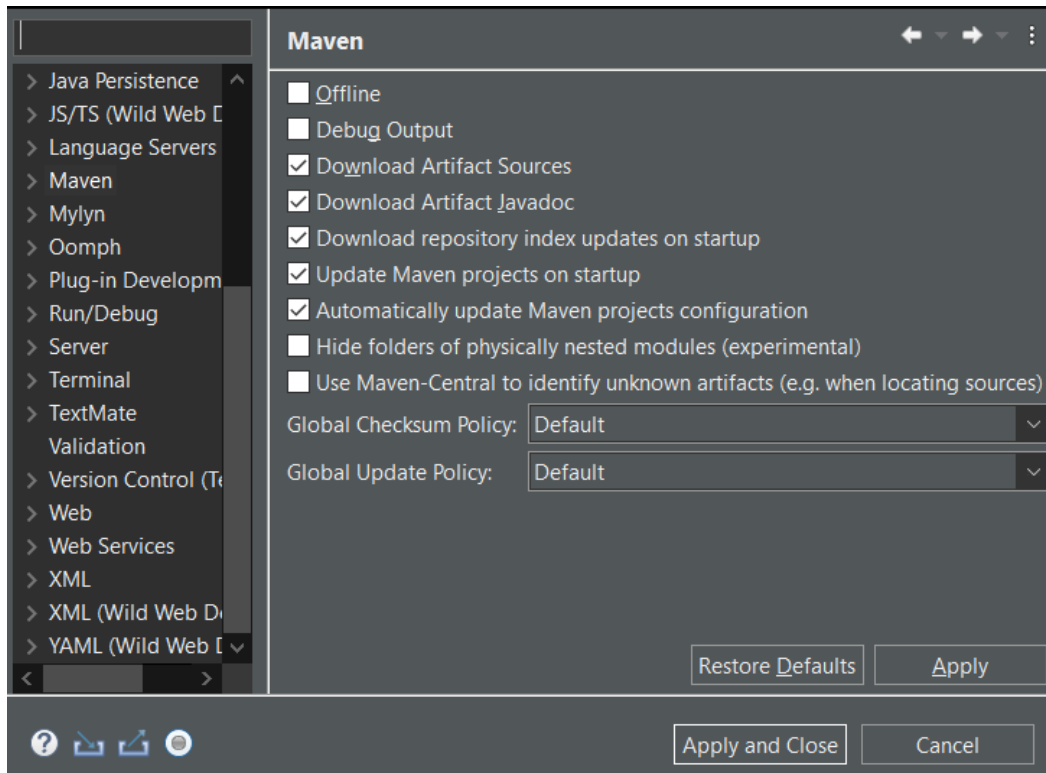
```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

```
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>

    <!-- Hibernate settings -->
    <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.hbm2ddl.auto">update</property>

    <!-- Mapping class -->
    <mapping class="hibernateEntities.User"/>
  </session-factory>
</hibernate-configuration>
```

Hibernate



```
1 package hibernateEntities;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Id;
5 import javax.persistence.Table;
6
7 @Entity
8 @Table(name = "user") // Maps to the "user" table
9 public class User {
10
11     @Id
12     private int id;
13     private String name;
14     private String email;
15     private String password;
16     private String city;
17
18     // Getters and Setters
19
20     public int getId() {
21         return id;
22     }
23
24     public void setId(int id) {
25         this.id = id;
26     }
27
28     public String getName() {
29         return name;
30     }
31
32     public void setName(String name) {
33         this.name = name;
34     }
35
36     public String getEmail() {
37         return email;
38     }
39
40     public void setEmail(String email) {
41         this.email = email;
42     }
43
44     public String getPassword() {
45         return password;
46     }
47
48     public void setPassword(String password) {
49         this.password = password;
50     }
51
52     public String getCity() {
53         return city;
54     }
55
56     public void setCity(String city) {
57         this.city = city;
58     }
59
60 }
```

```
1 package com.tech;
2
3 import org.hibernate.Session;
4 import org.hibernate.SessionFactory;
5 import org.hibernate.Transaction;
6 import org.hibernate.cfg.Configuration;
7 import hibernateEntities.User;
8
9 public class App {
10     public static void main(String[] args) {
11         User user = new User();
12         user.setId(1);
13         user.setName("Wasim");
14         user.setEmail("wasimkhan@gmail.com");
15         user.setPassword("123");
16         user.setCity("Katiyar");
17
18         // Load configuration and build session factory
19         Configuration cfg = new Configuration();
20         cfg.configure("config/hibernate.cfg.xml"); // Ensure the path is correct
21
22         SessionFactory sessionFactory = cfg.buildSessionFactory();
23         Session session = sessionFactory.openSession();
24         Transaction transaction = session.beginTransaction();
25
26         try {
27             session.save(user); // Save the User object to the database
28             transaction.commit(); // Commit the transaction
29             System.out.println("User saved successfully!");
30
31         } catch (Exception e) {
32             e.printStackTrace(); // Print the stack trace for easier debugging
33             transaction.rollback(); // Roll back the transaction in case of error
34             System.out.println("Transaction rolled back due to an error.");
35         } finally {
36             session.close(); // Close the session to release resources
37         }
38     }
39
40 }
```

Hibernate

CRUD Operation methods provided by Session are :

= Insert Operation :-

- > save()
- > persist()

= Select Operation :-

- > get()
- > load()

= Update Operation :-

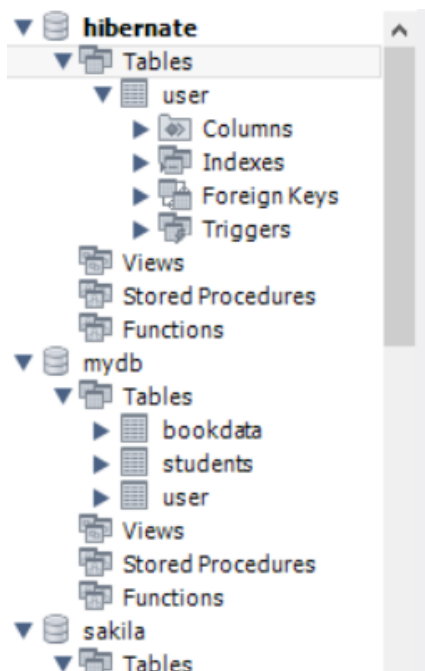
- > saveOrUpdate()
- > update()




= Delete Operation :-

- > delete()



```
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta
Hibernate: insert into user (city, email, name, password, id) values (?, ?, ?, ?, ?)
User saved successfully!
```



Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
	id	name	email	password	city
▶	1	Wasim	wasimkhanktr@gmail.com	123	Katihar
*	NULL	NULL	NULL	NULL	NULL