What is Spring JPA or Hibernate?

Spring JPA or Hibernate is an Object Relational Mapping Software for Java.

You can refer to following link if you want to know about what is ORM (Object Relational Mapping Software).

http://www.cppcourses.com/ormhandbook.pdf

In this Tutorial we will create a Spring Boot Application which will run through command line and will connect to a MySQL Server Database using Spring JPA or Hibernate.

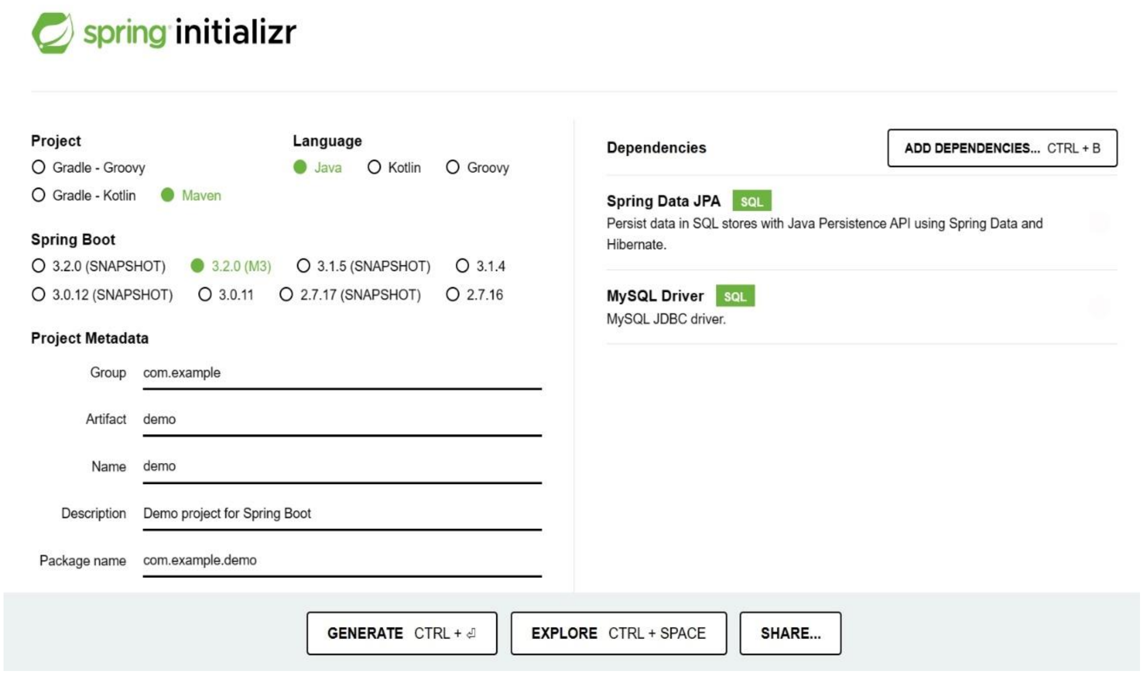
Following are the MySQL scripts to create user, database and table

```
-- Drop user first if they exist
DROP USER if exists 'springstudent'@'%' ;

-- Now create user with prop privileges
CREATE USER 'springstudent'@'%' IDENTIFIED BY 'springstudent';

GRANT ALL PRIVILEGES ON * . * TO 'springstudent'@'%';
```

```
CREATE DATABASE  IF NOT EXISTS `student_tracker`;
USE `student_tracker`;

--
-- Table structure for table `student`
--

DROP TABLE IF EXISTS `student`;

CREATE TABLE `student` (
  `id` int NOT NULL AUTO_INCREMENT,
  `first_name`varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

Go to https://start.spring.io

and create a demo project as shown in the following screenshot.


spring initializr

Project
○ Gradle - Groovy
○ Gradle - Kotlin
● Maven

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 3.2.0 (SNAPSHOT) ● 3.2.0 (M3) ○ 3.1.5 (SNAPSHOT) ○ 3.1.4
○ 3.0.12 (SNAPSHOT) ○ 3.0.11 ○ 2.7.17 (SNAPSHOT) ○ 2.7.16

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo

Dependencies

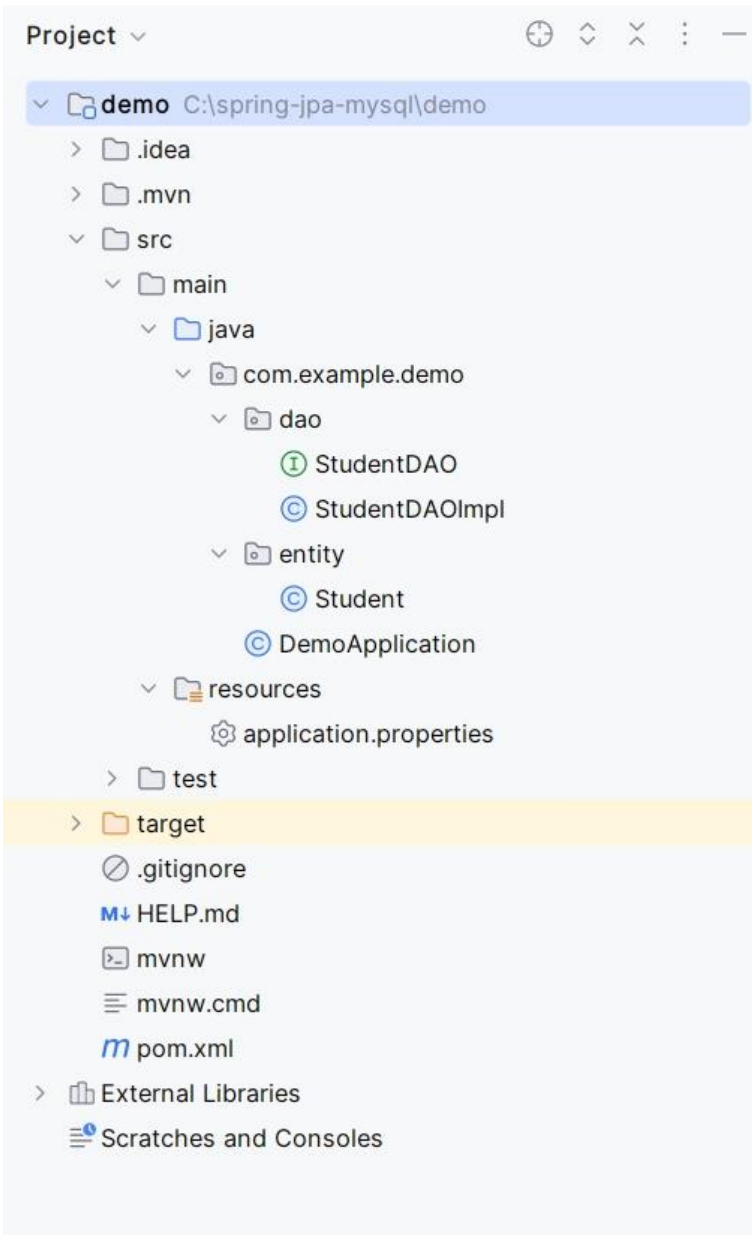
Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

Click on Generate and demo.zip will be available for download

extract demo.zip in a folder and open demo project in IntelliJ IDEA Community Edition.

After opening the project in IDE create the following project structure



```
Project
demo C:\spring-jpa-mysql\demo
  .idea
  .mvn
  src
    main
      java
        com.example.demo
          dao
            StudentDAO
            StudentDAOImpl
          entity
            Student
          DemoApplication
      resources
        application.properties
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

Here dao is a package

entity is a package

StudentDAO is an interface (the class that implements the interface should contain definitions of all methods in the interface)

StudentDAOImpl is a class

Student is a class

DemoApplication is a class

application.properties is a property file that contains all configuration of the project

Following is the code for all these classes and interface

Code for StudentDAO Interface

```java
package com.example.demo.dao;

import com.example.demo.entity.Student;

import java.util.List;

public interface StudentDAO {

    void save(Student theStudent);

    Student findById(Integer id);

    List<Student> findAll();

    List<Student> findByLastName(String theLastName);

    void update(Student theStudent);

    void delete(Integer id);

    int deleteAll();
}
```

Code for StudentDAOImpl

```java
package com.example.demo.dao;

import com.example.demo.entity.Student;
import jakarta.persistence.EntityManager;
import jakarta.persistence.TypedQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Repository
public class StudentDAOImpl implements StudentDAO {

    // define field for entity manager
    private EntityManager entityManager;

    // inject entity manager using constructor injection
    @Autowired
    public StudentDAOImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    // implement save method
    @Override
    @Transactional
    public void save(Student theStudent) {
        entityManager.persist(theStudent);
    }

    @Override
    public Student findById(Integer id) {
        return entityManager.find(Student.class, id);
    }

    @Override
    public List<Student> findAll() {
        // create query
        TypedQuery<Student> theQuery = entityManager.createQuery("FROM
Student", Student.class);

        // return query results
        return theQuery.getResultList();
    }

    @Override
    public List<Student> findByLastName(String theLastName) {
        // create query
        TypedQuery<Student> theQuery = entityManager.createQuery(
                                                "FROM Student WHERE
lastName=:theData", Student.class);

        // set query parameters
        theQuery.setParameter("theData", theLastName);

        // return query results
        return theQuery.getResultList();
    }

    @Override
    @Transactional
    public void update(Student theStudent) {
        entityManager.merge(theStudent);
    }

    @Override
    @Transactional
    public void delete(Integer id) {

        // retrieve the student
        Student theStudent = entityManager.find(Student.class, id);

        // delete the student
        entityManager.remove(theStudent);
    }

    @Override
    @Transactional
    public int deleteAll() {

        int numRowsDeleted = entityManager.createQuery("DELETE FROM
Student").executeUpdate();

        return numRowsDeleted;
    }
}
```

Code for Student Class

```
package com.example.demo.entity;
import jakarta.persistence.*;

@Entity
@Table(name="student")
public class Student {

    // define fields
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    @Column(name="email")
    private String email;

    // define constructors
    public Student() {

    }

    public Student(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    // define getters/setters

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }


    // define toString() method

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

Code for DemoApplication

```
package com.example.demo;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import com.example.demo.dao.StudentDAO;
import java.util.List;
import com.example.demo.entity.Student;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
       SpringApplication.run(DemoApplication.class, args);
    }
    @Bean
    public CommandLineRunner commandLineRunner(StudentDAO studentDAO) {

       return runner -> {
          // createStudent(studentDAO);

          createMultipleStudents(studentDAO);

          // readStudent(studentDAO);

          // queryForStudents(studentDAO);

          // queryForStudentsByLastName(studentDAO);

          // updateStudent(studentDAO);

          // deleteStudent(studentDAO);
```

```java
        // deleteAllStudents(studentDAO);
    };
}

private void deleteAllStudents(StudentDAO studentDAO) {

    System.out.println("Deleting all students");
    int numRowsDeleted = studentDAO.deleteAll();
    System.out.println("Deleted row count: " + numRowsDeleted);
}

private void deleteStudent(StudentDAO studentDAO) {

    int studentId = 3;
    System.out.println("Deleting student id: " + studentId);
    studentDAO.delete(studentId);
}

private void updateStudent(StudentDAO studentDAO) {

    // retrieve student based on the id: primary key
    int studentId = 1;
    System.out.println("Getting student with id: " + studentId);
    Student myStudent = studentDAO.findById(studentId);

    // change first name to "John"
    System.out.println("Updating student ...");
    myStudent.setFirstName("John");

    // update the student
    studentDAO.update(myStudent);

    // display the updated student
    System.out.println("Updated student: " + myStudent);
}

private void queryForStudentsByLastName(StudentDAO studentDAO) {

    // get a list of students
    List<Student> theStudents = studentDAO.findByLastName("Doe");

    // display list of students
    for (Student tempStudent : theStudents) {
        System.out.println(tempStudent);
    }
}

private void queryForStudents(StudentDAO studentDAO) {

    // get a list of students
    List<Student> theStudents = studentDAO.findAll();

    // display list of students
    for (Student tempStudent : theStudents) {
        System.out.println(tempStudent);
    }
}

private void readStudent(StudentDAO studentDAO) {

    // create  a student object
```

```
        System.out.println("Creating new student object ...");
        Student tempStudent = new Student("Daffy", "Duck",
"daffy@abcdefdf.com");

        // save the student
        System.out.println("Saving the student ...");
        studentDAO.save(tempStudent);

        // display id of the saved student
        int theId = tempStudent.getId();
        System.out.println("Saved student. Generated id: " + theId);

        // retrieve student based on the id: primary key
        System.out.println("Retrieving student with id: " + theId);
        Student myStudent = studentDAO.findById(theId);

        // display student
        System.out.println("Found the student: " + myStudent);
    }

    private void createMultipleStudents(StudentDAO studentDAO) {

        // create multiple students
        System.out.println("Creating 3 student objects ...");
        Student tempStudent1 = new Student("John", "Doe",
"john@abcdefcdrt.com");
        Student tempStudent2 = new Student("Mary", "Public",
"mary@abcdefsdsd.com");
        Student tempStudent3 = new Student("Bonita", "Applebum",
"bonita@jhjhj.com");

        // save the student objects
        System.out.println("Saving the students ...");
        studentDAO.save(tempStudent1);
        studentDAO.save(tempStudent2);
        studentDAO.save(tempStudent3);
    }

    private void createStudent(StudentDAO studentDAO) {

        // create the student object
        System.out.println("Creating new student object ...");
        Student tempStudent = new Student("Paul", "Doe",
"paul@luv2code.com");

        // save the student object
        System.out.println("Saving the student ...");
        studentDAO.save(tempStudent);

        // display id of the saved student
        System.out.println("Saved student. Generated id: " +
tempStudent.getId());
    }
}
```

Code for application.properties

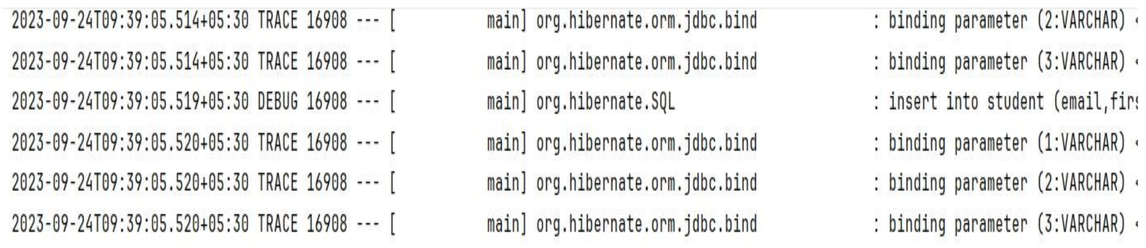
```
spring.datasource.url=jdbc:mysql://localhost:3306/student_tracker
spring.datasource.username=springstudent
spring.datasource.password=springstudent
```

```
# Turn off the Spring Boot banner
spring.main.banner-mode=off

# Reduce logging level. Set logging level to warn
logging.level.root=warn

# Add logging configs to display SQL statements
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.orm.jdbc.bind=trace

# Configure JPA/Hibernate to auto create the tables
# the "update" config will keep existing data in the table
spring.jpa.hibernate.ddl-auto=update
```

After creating structure of the project and writing code for interfaces and classes run the project

as right click on DemoApplication and select Run 'DemoApplication.main()'

You will get the following output



```
2023-09-24T09:39:05.514+05:30 TRACE 16908 --- [           main] org.hibernate.orm.jdbc.bind              : binding parameter (2:VARCHAR)
2023-09-24T09:39:05.514+05:30 TRACE 16908 --- [           main] org.hibernate.orm.jdbc.bind              : binding parameter (3:VARCHAR)
2023-09-24T09:39:05.519+05:30 DEBUG 16908 --- [           main] org.hibernate.SQL                        : insert into student (email,fir
2023-09-24T09:39:05.520+05:30 TRACE 16908 --- [           main] org.hibernate.orm.jdbc.bind              : binding parameter (1:VARCHAR)
2023-09-24T09:39:05.520+05:30 TRACE 16908 --- [           main] org.hibernate.orm.jdbc.bind              : binding parameter (2:VARCHAR)
2023-09-24T09:39:05.520+05:30 TRACE 16908 --- [           main] org.hibernate.orm.jdbc.bind              : binding parameter (3:VARCHAR)
```