SSL, java extenson pack, sqlTool

Viết table class -> tạo constructor abstrac trước(toString) -> tạo dbs, scrip -> netstat -ano, taskkill /pid yourid /f

====================================pom==========================================

<dependencies>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<!--

https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->

<dependency>

<groupId>org.mariadb.jdbc</groupId>

<artifactId>mariadb-java-client</artifactId>

<version>3.3.3</version>

</dependency>

<!--

https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<version>12.3.0.jre17-preview</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.10.1</version>

</dependency>

</dependencies>

=========================persistence=========================

<persistence-unit name=*"jpa-mssql"*>

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<properties>

<property name=*"jakarta.persistence.jdbc.driver"* value=*"com.microsoft.sqlserver.jdbc.SQLServerDriver"*/>

<property name=*"jakarta.persistence.jdbc.dialect"* value=*"org.hibernate.dialect.SQLServerDialect"*/>

<property name=*"hibernate.connection.url"* value=*"jdbc:sqlserver://localhost:1433;databaseName=NguyenMinhTien\_21044611;trustServerCertificate=true;encrypt=true;"*/>

<property name=*"hibernate.connection.username"* value=*"sa"*/>

<property name=*"hibernate.connection.password"* value=*"12345678"*/>

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

<property name=*"hibernate.show\_sql"* value=*"false"*/>

<property name=*"hibernate.format\_sql"* value=*"true"*/>

</properties>

</persistence-unit>

**=================================daoImpl==================================**

**private** EntityManager em;

**public** CourseImpl() **throws** RemoteException{

em = Persistence.*createEntityManagerFactory*("jpa-mssql").createEntityManager();

}

================================entity=====================================

@ManyToOne(fetch = FetchType.***LAZY***)

@JoinColumn(name = "DepartmentID")

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(

name = "CourseInstructor",

joinColumns = @JoinColumn(name = "CourseID"),

inverseJoinColumns = @JoinColumn(name = "PersonID")

)

@OneToOne

@JoinColumn(name = "InstructorID")

@OneToMany(mappedBy = "student")

**private** Set<StudentGrade> studentGrades;

@Inheritance(strategy = InheritanceType.***SINGLE\_TABLE***)

@DiscriminatorColumn(name = "Discriminator", discriminatorType = DiscriminatorType.***STRING***)

@ElementCollection

@CollectionTable(name = "books\_authors", joinColumns = @JoinColumn(name = "ISBN"))

@Column(name = "author", nullable = **false**)

**private** Set<String> authors;

=======================hàm=========================

@Override

**public** Map<String, Long> numberTitleCourseByNamePerson(String name) **throws** RemoteException {

Map<String, Long> map = **new** HashMap<>();

List<Object[]> ds = em.createQuery("SELECT c.title, count(\*) FROM Course c inner join c.studentGrades sg "

+ "where sg.student.id "

+ "in (select s.id from Student s where s.firstName like :name) "

+ "GROUP BY c.title", Object[].**class**)

.setParameter("name", "%"+name+"%")

.getResultList();

**for** (Object[] c : ds) {

map.put((String) c[0], (**long**) c[1]);

}

**return** map;

}

**public** Map<LocalDate, Integer> getTotalPriceOrder(LocalDate date) {

Map<LocalDate, Integer> map = **new** HashMap();

em.createQuery(

"SELECT o.orderDate, SUM(oi.quantity \* oi.listPrice) "

+ "FROM Order o JOIN o.orderItems oi "

+ "WHERE o.orderDate = :date "

+ "GROUP BY o.orderDate",

Object[].**class**).setParameter("date", date).getResultList()

.forEach(p -> map.put((LocalDate) p[0], ((Number) p[1]).intValue()));

**return** map;

}

@Override

**public** **boolean** updateCourse(**int** id , String title) **throws** RemoteException {

EntityTransaction tx = em.getTransaction();

Course course = em.find(Course.**class**, id);

course.setTitle(title);

**try** {

tx.begin();

em.merge(course);

tx.commit();

**return** **true**;

} **catch** (Exception e) {

tx.rollback();

e.printStackTrace();

}

**return** **false**;

}

==================================test================================

**class** StudentDAOTest {

**private** StudentService studentDAO;

@BeforeAll

**void** setUpBeforeClass() **throws** Exception {

studentDAO = **new** StudentImpl();

}

@AfterAll

**void** tearDownAfterClass() **throws** Exception {

studentDAO = **null**;

}

@Test

**void** testFindStudentsEnrolledIn() **throws** RemoteException {

List<Student> students = studentDAO.findStudentsEnrolledInCourse("po");

*assertEquals*(5, students.size());

Student student = students.stream().filter(s -> s.getId()==3).findFirst().orElse(**null**);

*assertNotNull*(student);

*assertEquals*("Peggy", student.getFirstName());

}

}

=================================server=================================

**public** **class** Server {

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

**try** (ServerSocket server = **new** ServerSocket(4611)){

System.***out***.println("connect server success");

**while** (**true**) {

Socket client = server.accept();

System.***out***.println("client connected");

System.***out***.println("client host: "+ client.getInetAddress().getHostName());

System.***out***.println("client port: "+ client.getPort());

Thread t = **new** Thread(**new** HandleClient(client));

t.start();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**class** HandleClient **implements** Runnable {

**private** Socket client;

**public** HandleClient(Socket client) {

**super**();

**this**.client = client;

}

@Override

**public** **void** run() {

// **TODO** Auto-generated method stub

**try** {

DataInputStream in = **new** DataInputStream(client.getInputStream());

ObjectOutputStream out = **new** ObjectOutputStream(client.getOutputStream());

**int** choice=0;

**while**(**true**) {

choice = in.readInt();

**switch** (choice) {

**case** 1: {

// dùng in để đọc dư liệu từ client

// tạo đối tượng food

**break**;

}

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

=====================================client==================================

**public** **class** Client {

@SuppressWarnings("unchecked")

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

**try** (Socket client = **new** Socket("DESKTOP-UFB2VOA", 4611); Scanner sc = **new** Scanner(System.***in***);) {

DataOutputStream out = **new** DataOutputStream(client.getOutputStream());

ObjectInputStream in = **new** ObjectInputStream(client.getInputStream());

System.***out***.println("connect to server");

**int** choice = 0;

**while** (**true**) {

System.***out***.println("\n1.");

System.***out***.println("\n.Nhập lựa chọn");

choice = sc.nextInt();

sc.nextLine();

out.writeInt(choice);

out.flush();

**switch** (choice) {

**case** 1: {

// dùng out để ghi dư liệu // dùng in để đọc dư liệu từ server

**break**;

}

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}