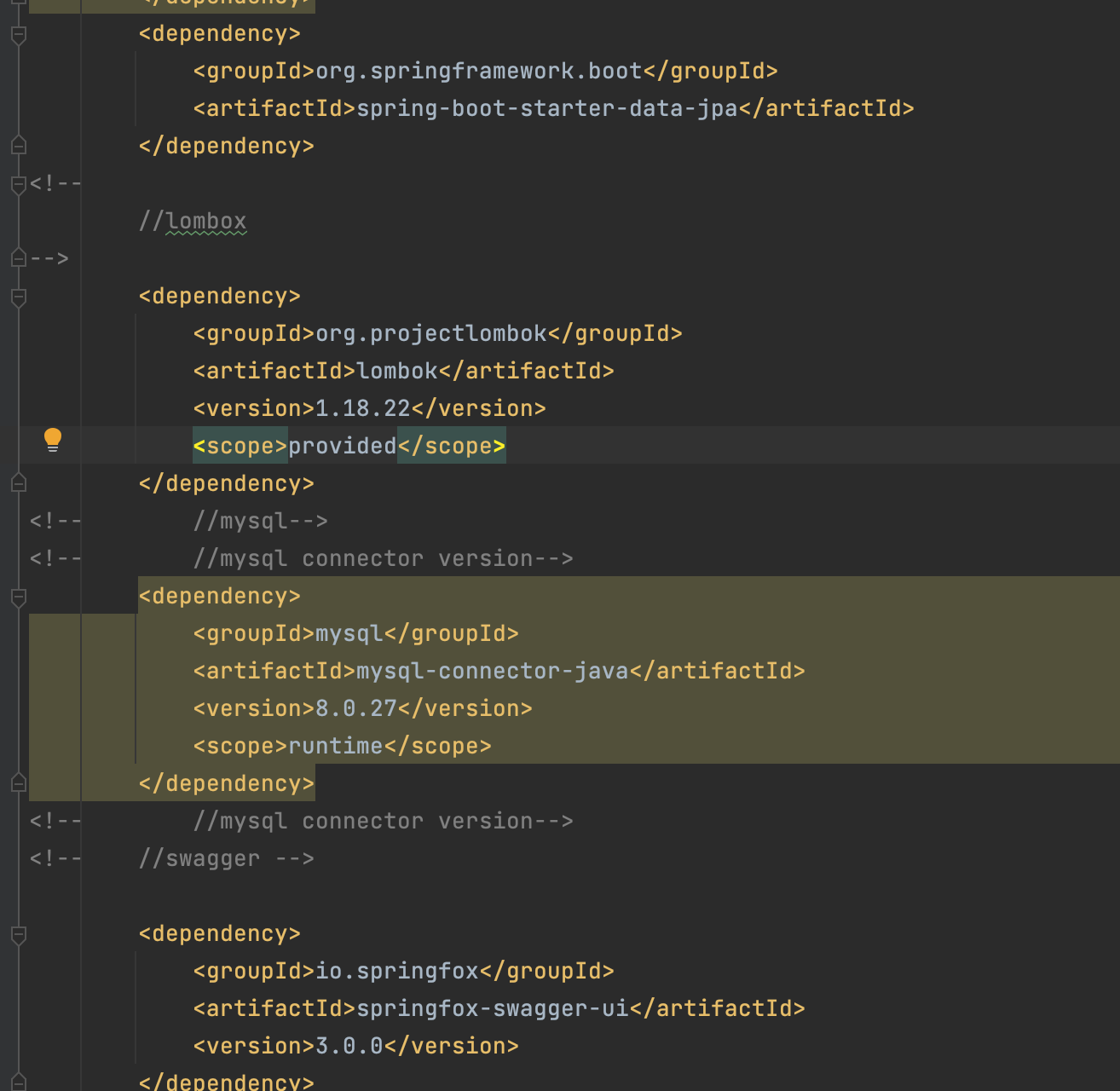
**Demo 1 : create a restful api project**

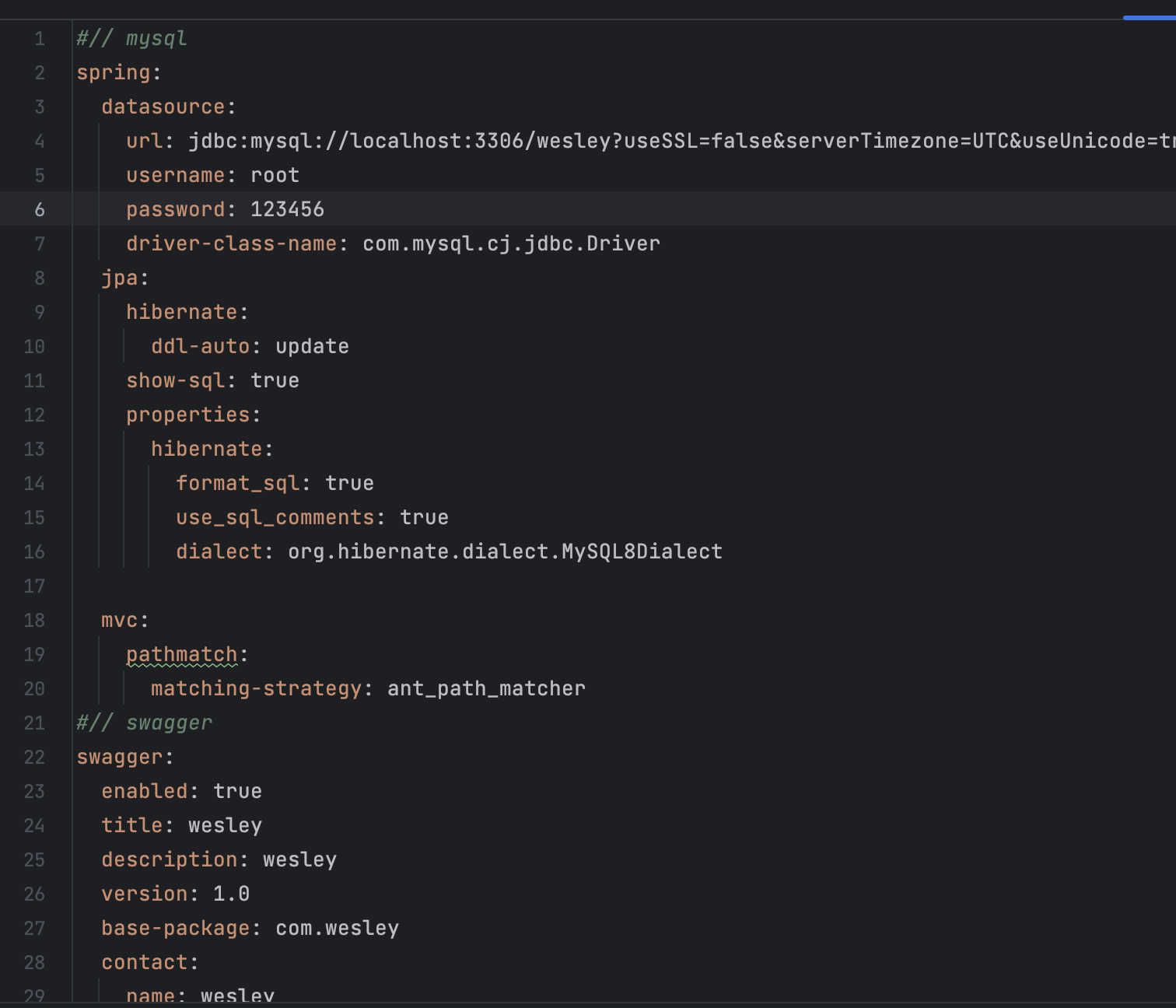
1) generate pom maven dependency -- lombok/mysql/jpa



2）mysql jdbc configuration in application.yml

3) JPA usage configuration

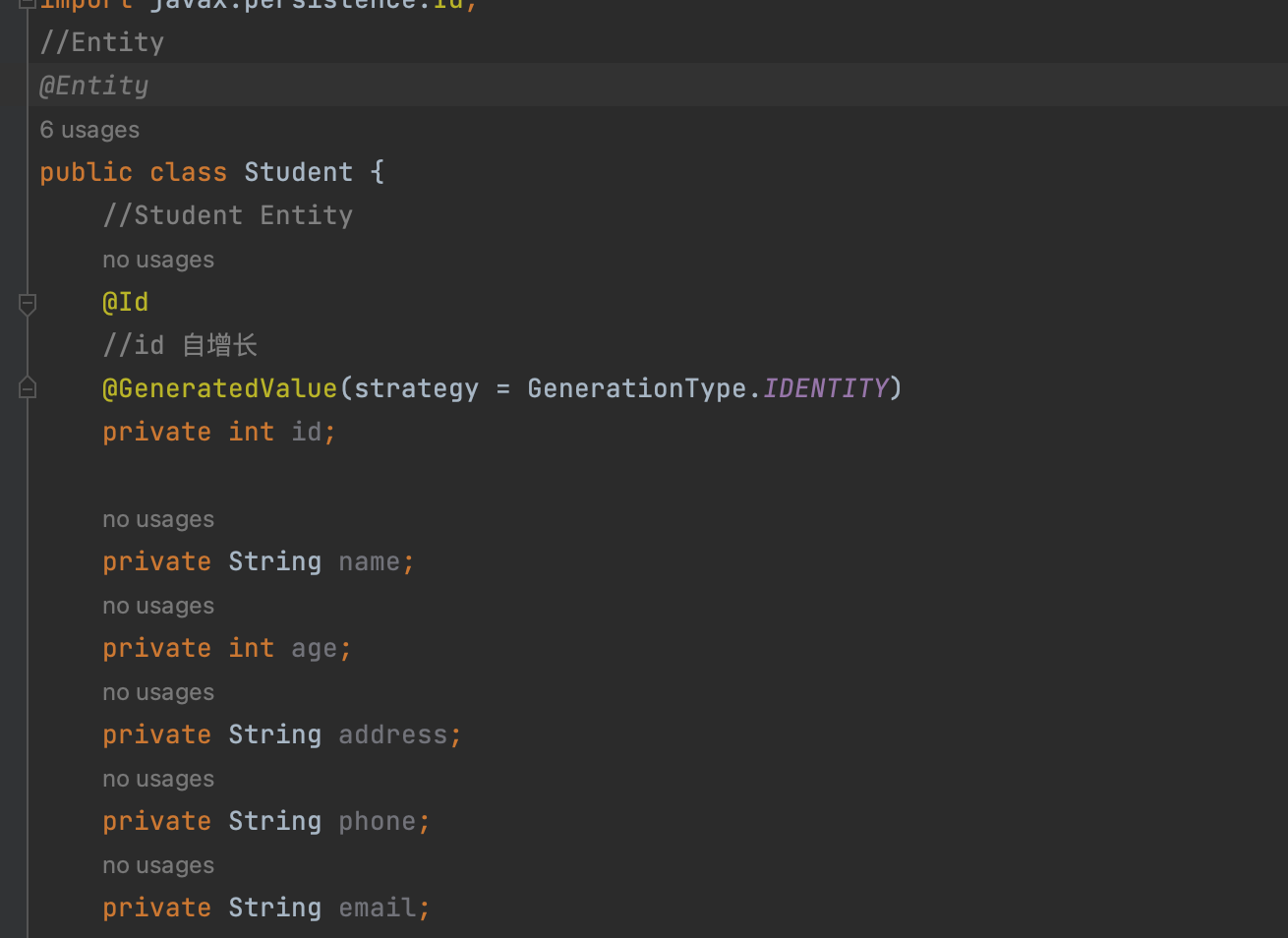
4）swagger config



5）Entity class auto generate next attribute

6) Entity @Entity annotation

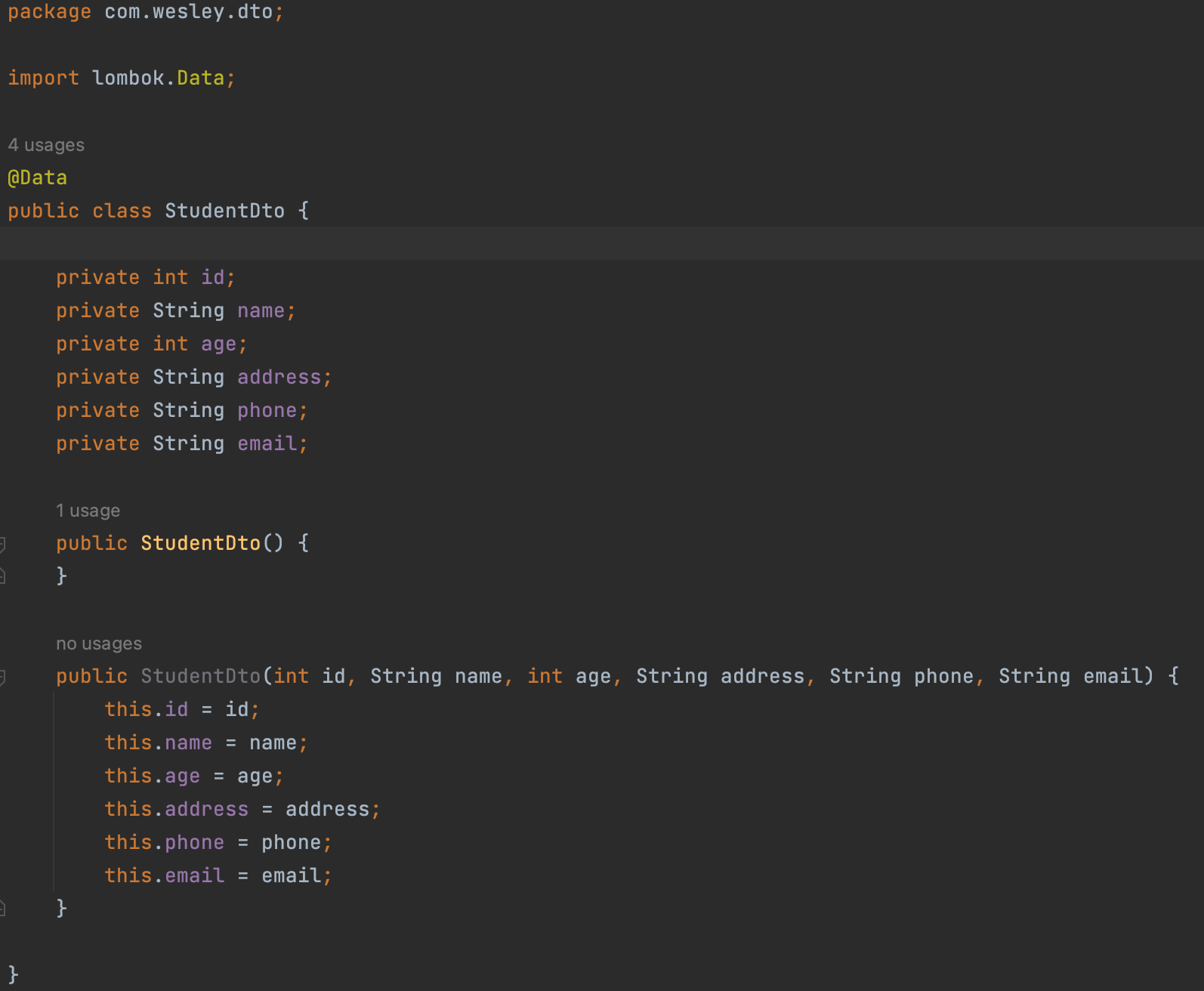
7）Id identity



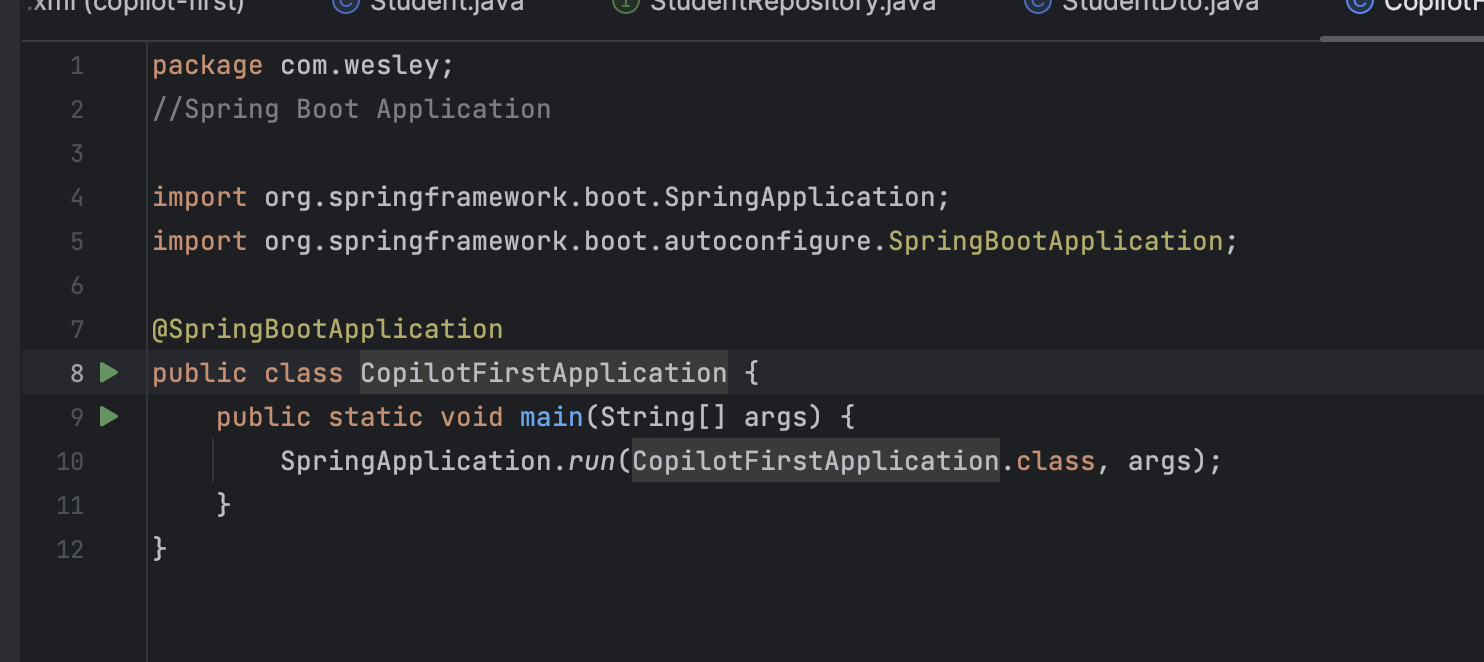
7）Create Repository auto extends CrudRepository



8）create DTO file， will auto match entity attribute， and generate contruct

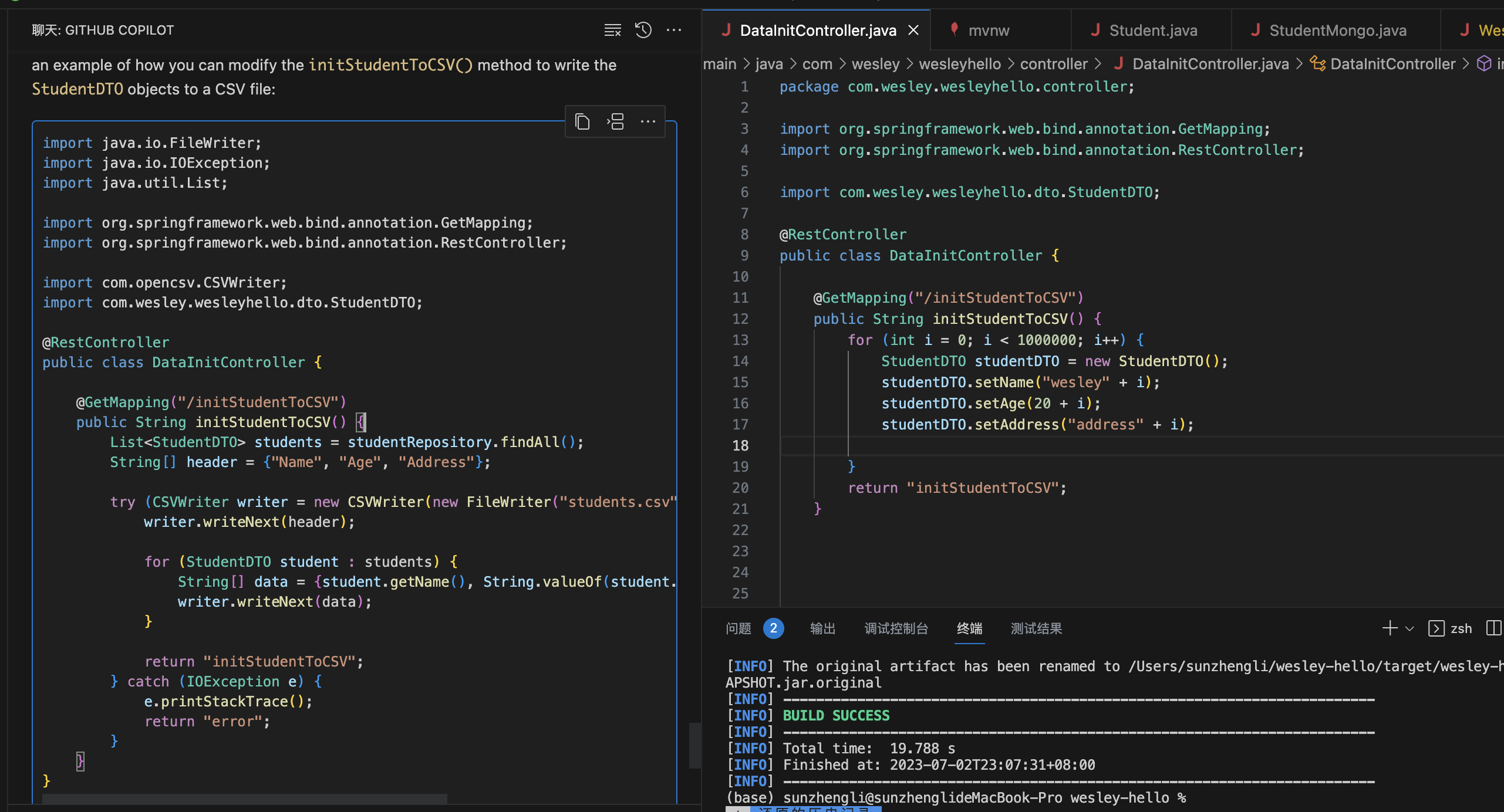


9） spring boot -- generate annotation and main method content

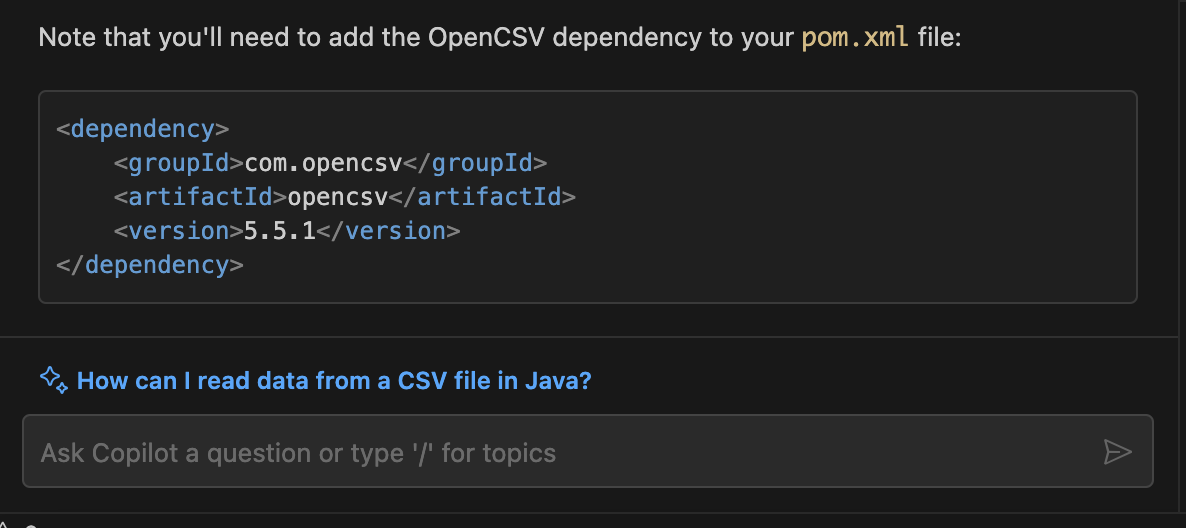


**Demo 2: Spring Batch project**

1) generate code for init data to csv code by existing code



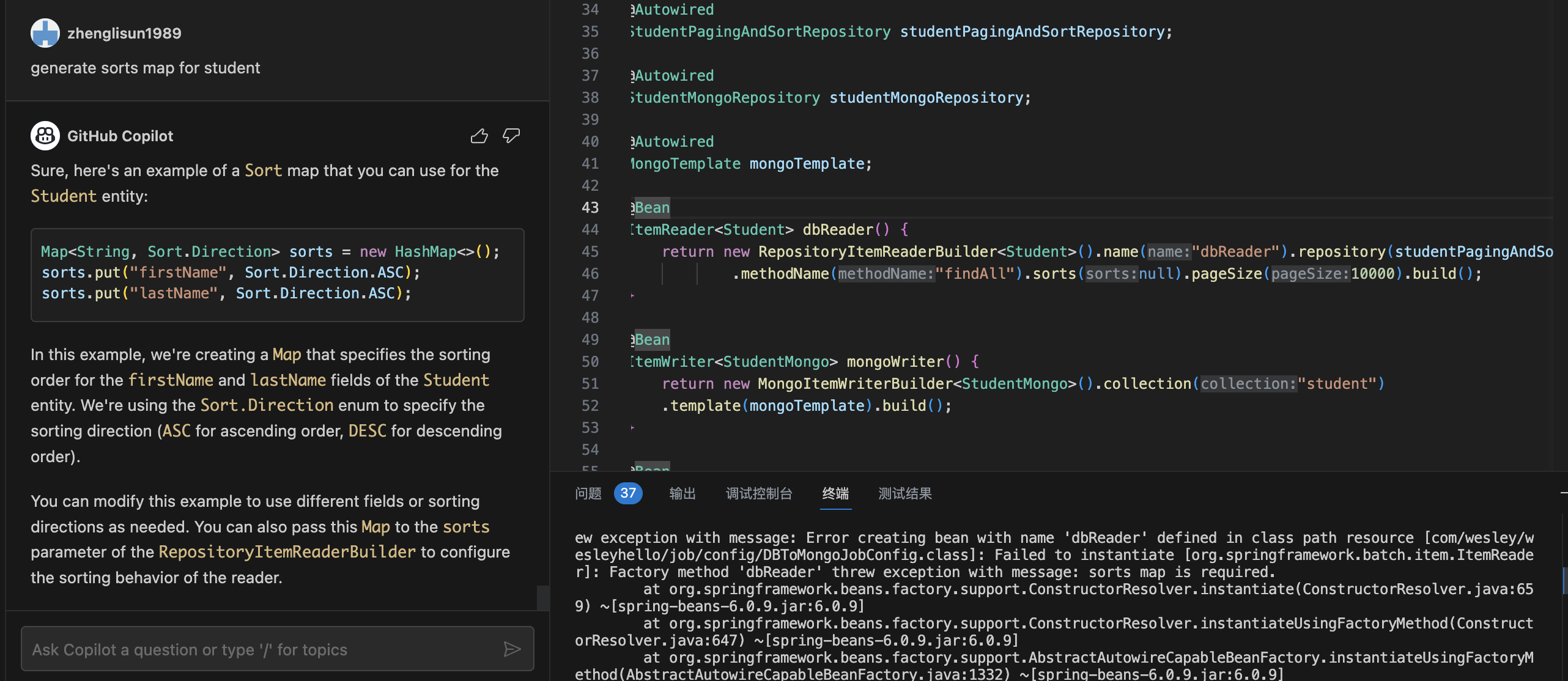
2) also has maven depency



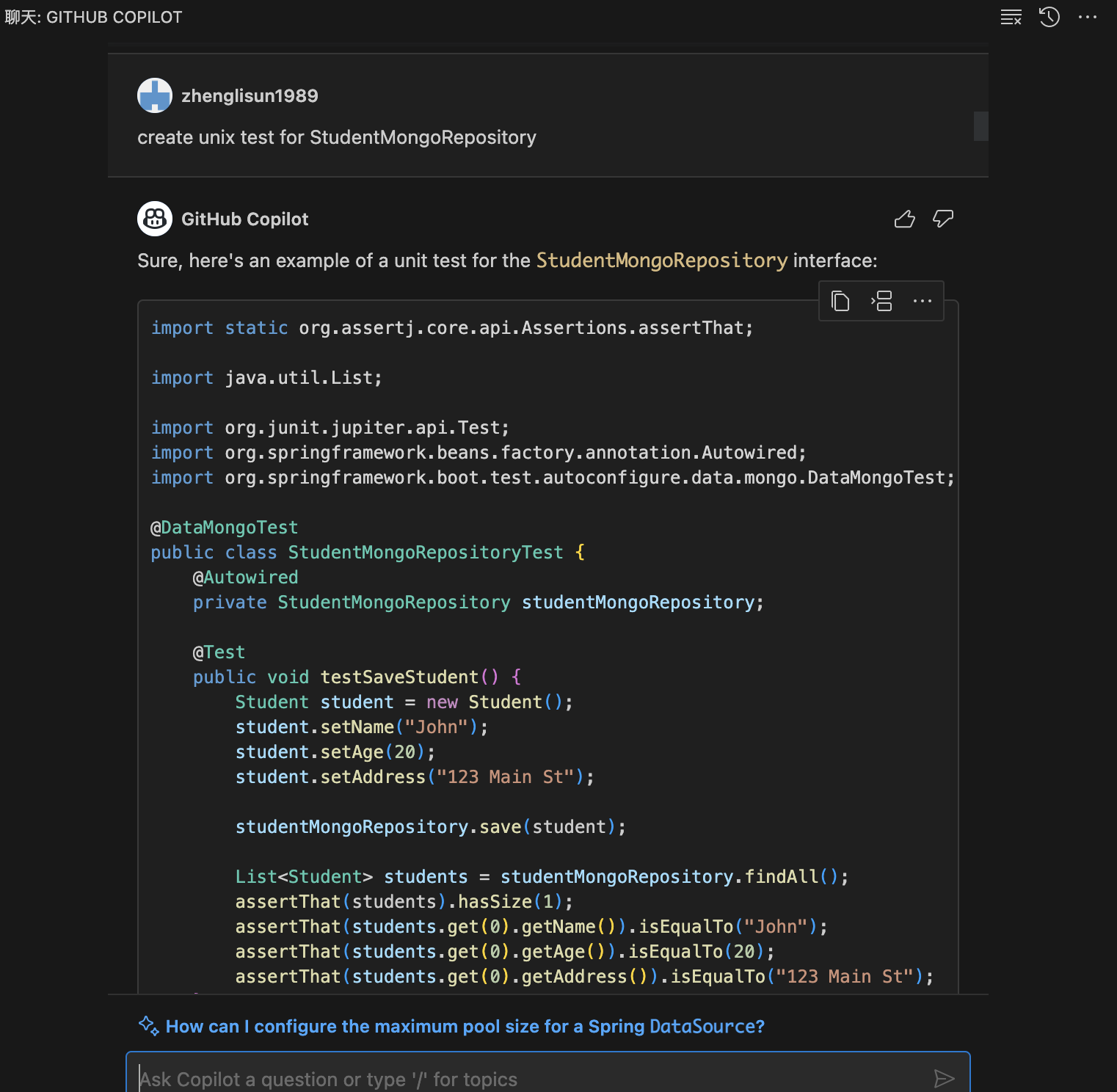
3). 结合上下文, 给出下一个提示



4). code run error, resolve it by copilot chat



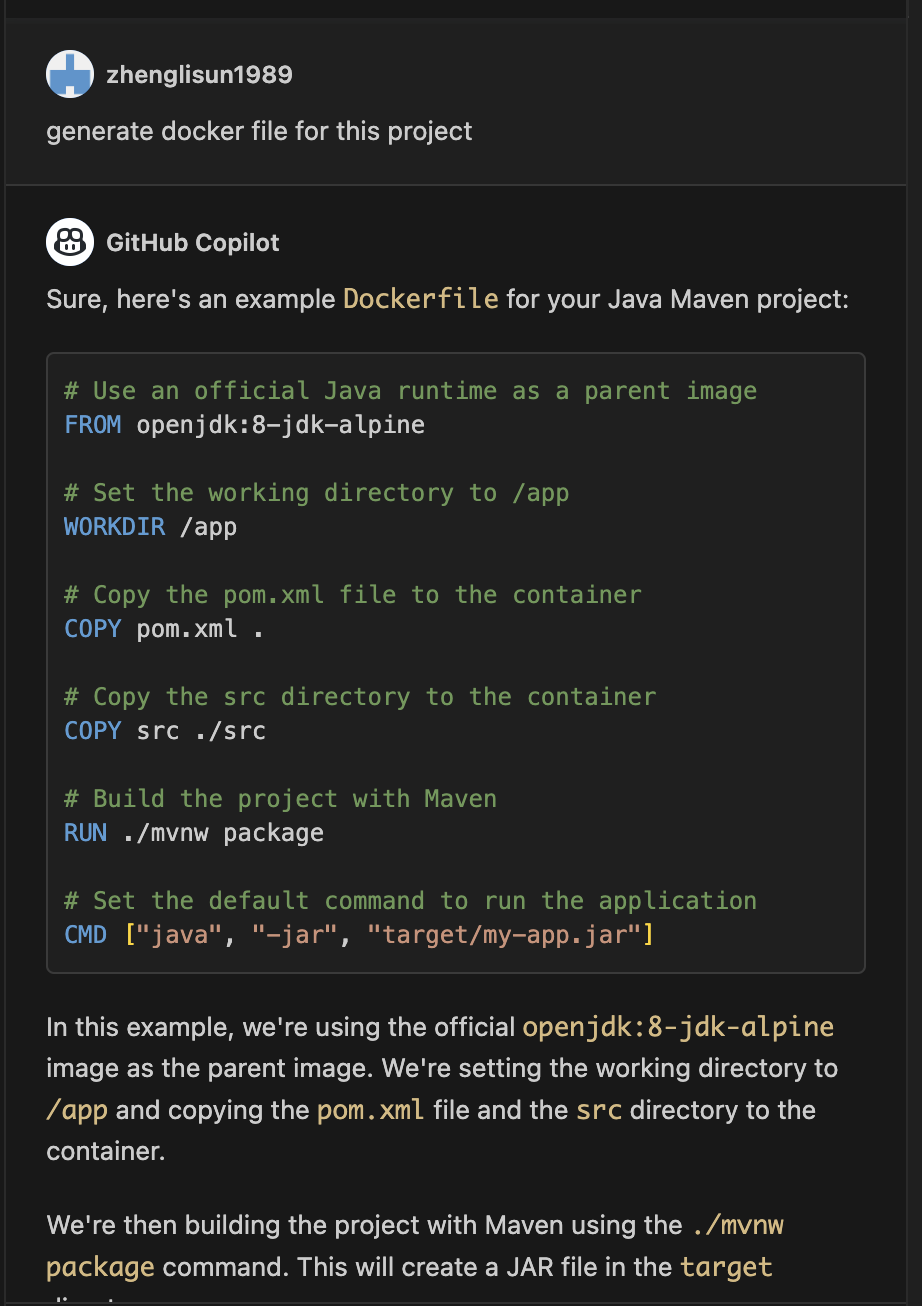
5) Unit test



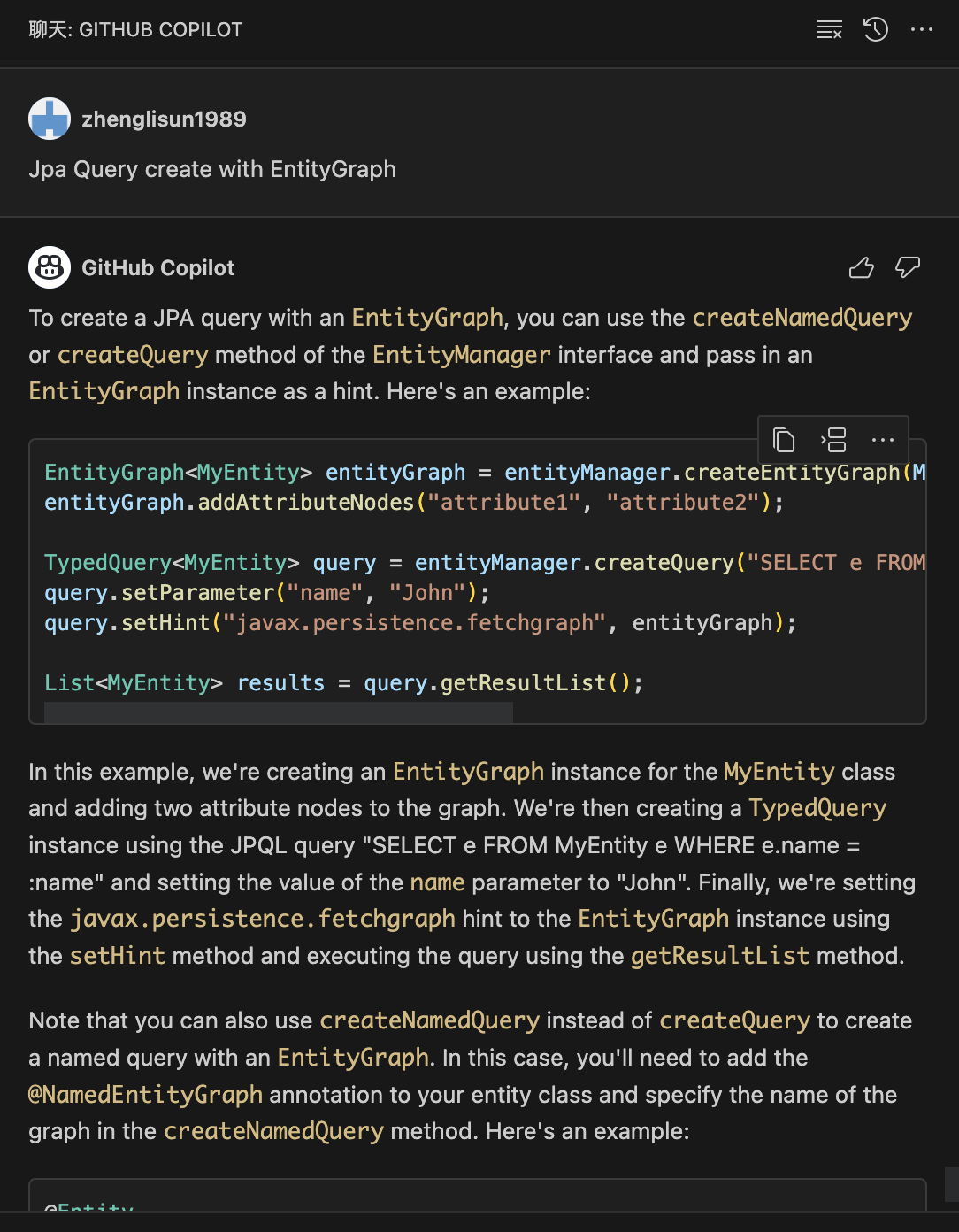
6) generate log level for each module

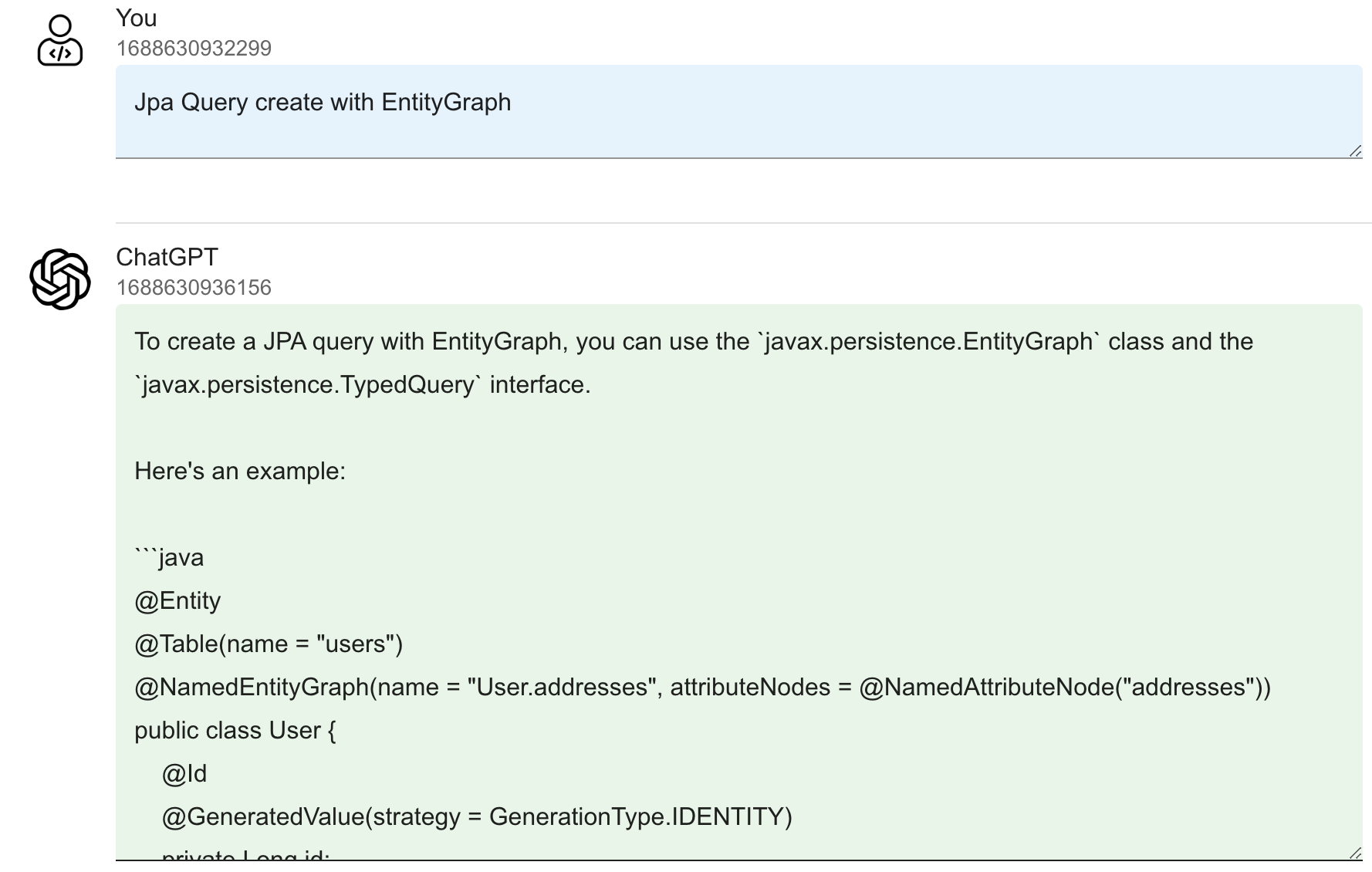


7) docker file generate



8) EntityGrapth





To create a JPA query with EntityGraph, you can use the javax.persistence.EntityGraph class and the javax.persistence.TypedQuery interface.

Here's an example:

@Entity

@Table(name = "users")

@NamedEntityGraph(name = "User.addresses", attributeNodes = @NamedAttributeNode("addresses"))

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "user", cascade = CascadeType.ALL)

private List<Address> addresses;

// getters and setters

}

@Entity

@Table(name = "addresses")

public class Address {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String street;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "user\_id")

private User user;

// getters and setters

}

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

@EntityGraph(value = "User.addresses", type = EntityGraph.EntityGraphType.LOAD)

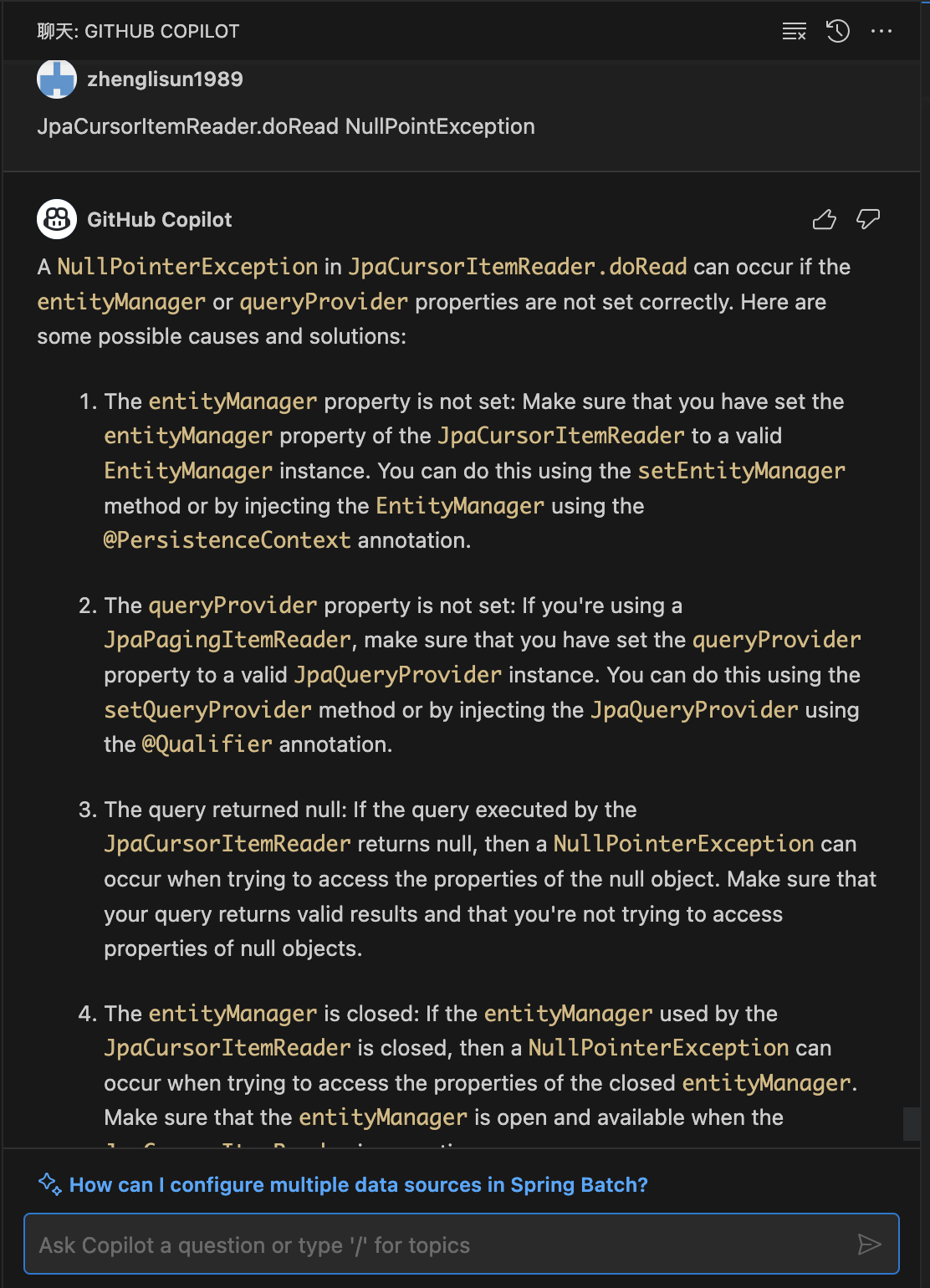
List<User> findAll();

}

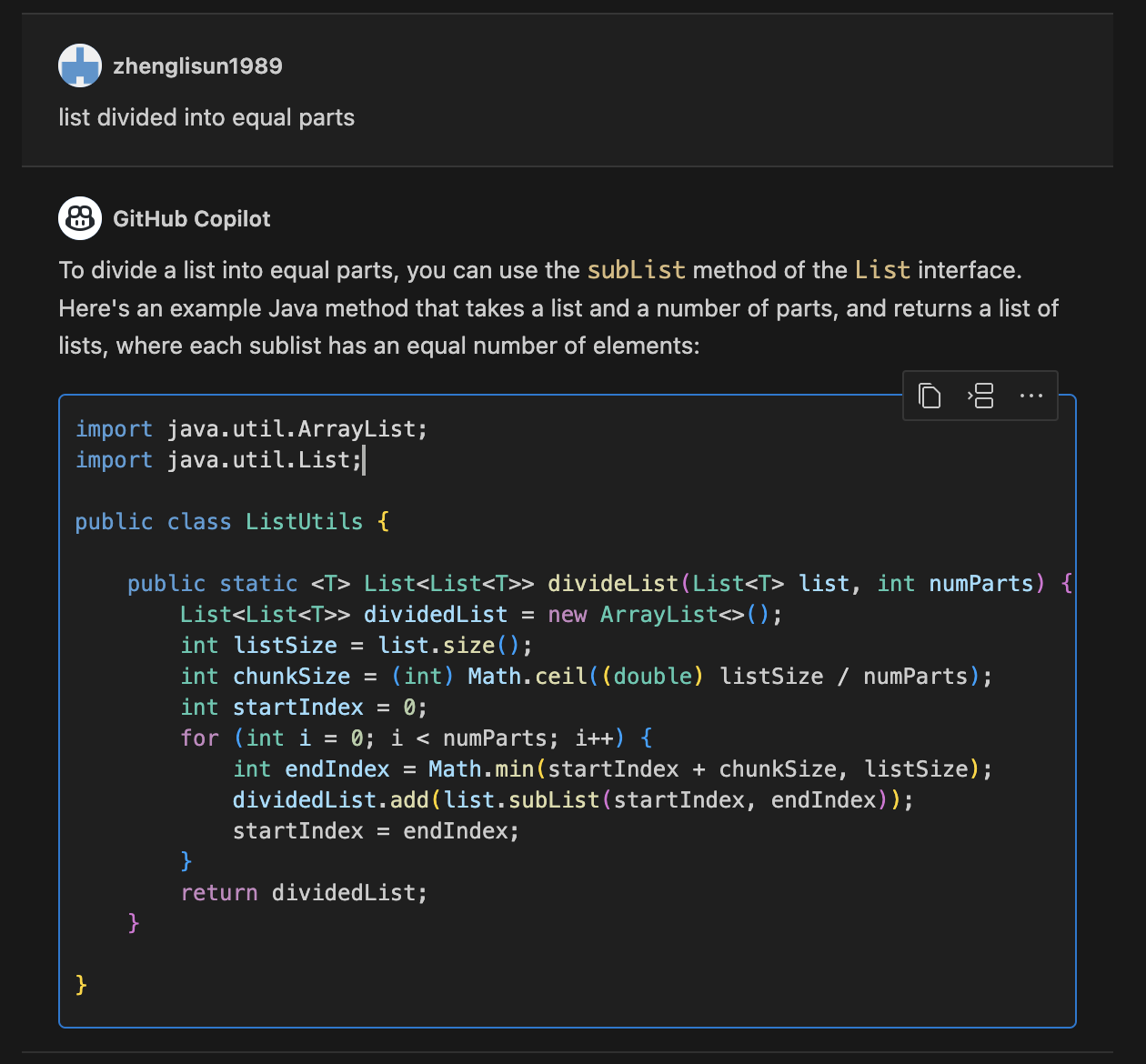
In the above example, we have two entities User and Address with a one-to-many relationship. The User entity is annotated with @NamedEntityGraph to define a named entity graph called "User.addresses" that eagerly loads the addresses. The UserRepository interface extends JpaRepository and uses the @EntityGraph annotation to specify the entity graph to be used for the findAll method.

Now, when you call the findAll method of the UserRepository, it will execute a JPA query that includes the join with the addresses table, fetching the addresses eagerly.

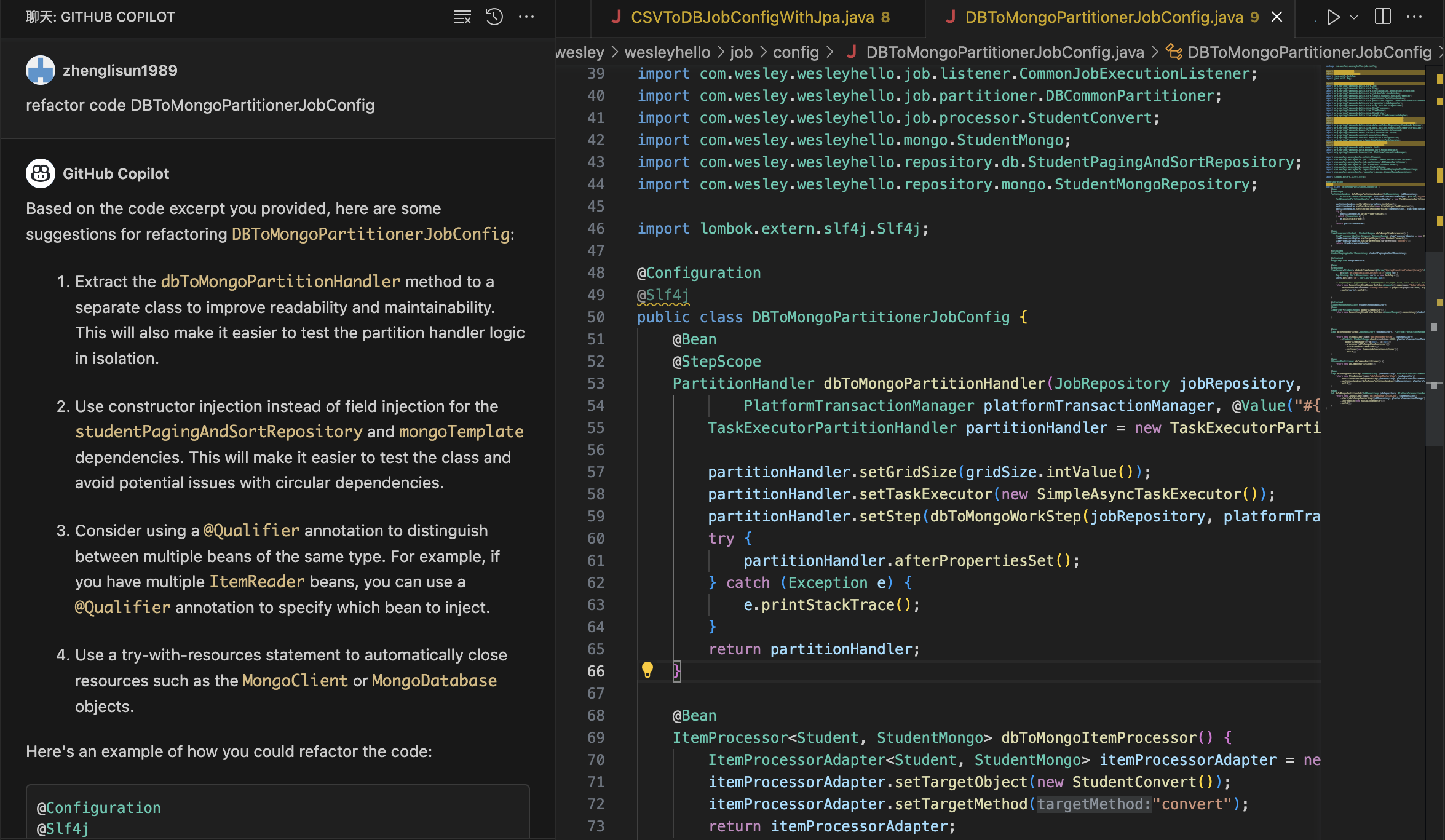
9) exception solution suggestion



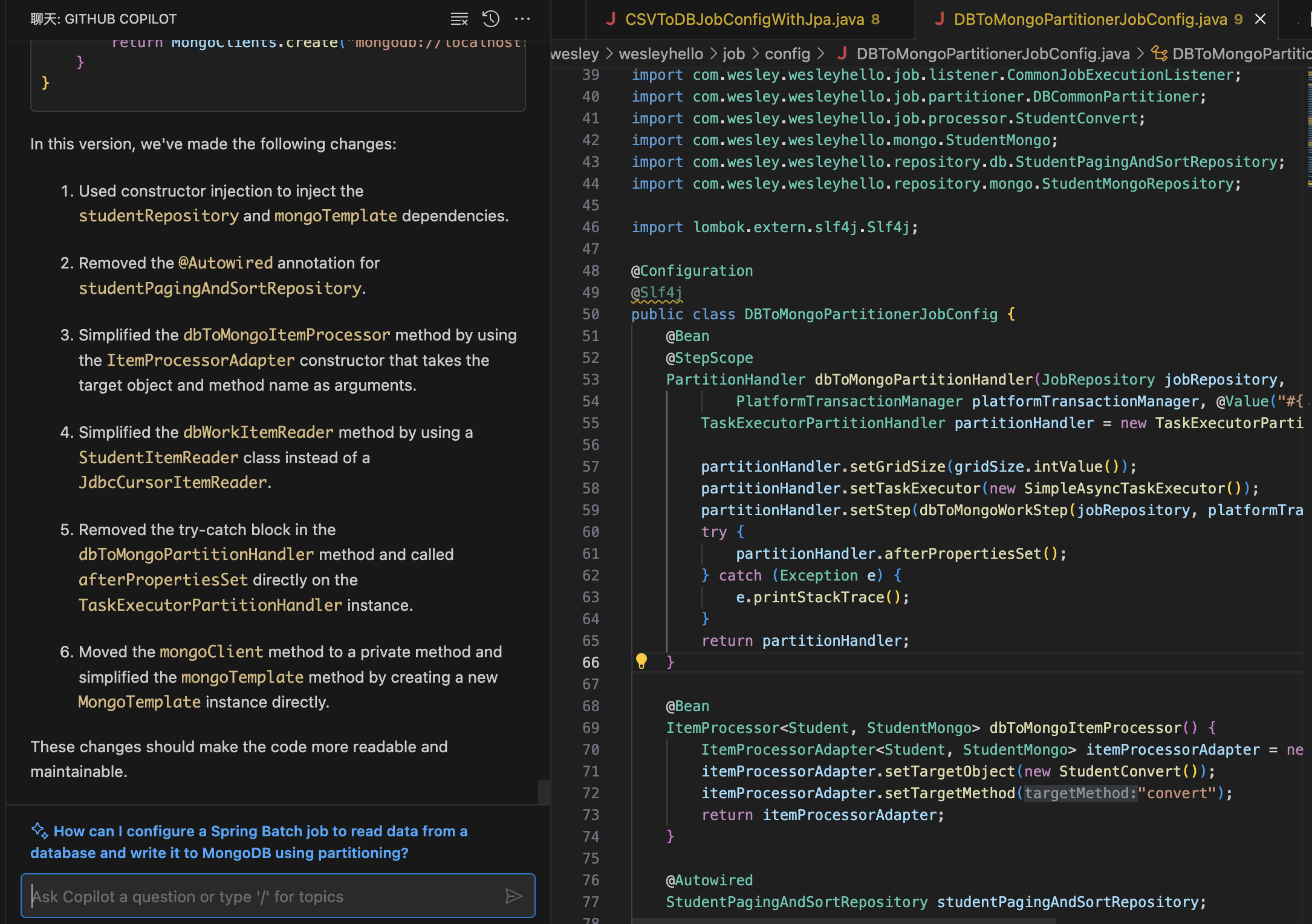
10) Algorithm



11) refactor code



12 clean code



3. Create Project structure

