**State Street Java Test**

**Bonus Points:**

**Q. What if we want to record course scores? What possible changes need to be made? Explain briefly.**

**A.** To record course scores, we need to create another bean “student\_course” which will have 3 columns named studientId, courseId and scores. We will need to establish mapping relationship between the 3 beans now.

student and student\_course: one to many relationship

student\_course and student: many to one relationship

Similary between course and student\_course.

So each row in student\_course table will look like following:

|  |  |  |
| --- | --- | --- |
| **studientId** | **courseId** | **scores** |
| 101 | 1001 | 98 |
| 101 | 1002 | 95 |
| 101 | 1003 | 99 |
| 102 | 1001 | 95 |
| 102 | 1002 | 97 |
| 102 | 1003 | 90 |
| 103 | 1001 | 91 |
| 103 | 1002 | 94 |
| 103 | 1003 | 98 |

**Q. How to find all students who don’t register for a given course?**

**A.** In “course” table we have a Set<Student> which tells us all the students who have registered for courses. In the “student” table we have the data of all the students.

Repository we can get all students and from set all students under the specific course (as parameter). Hence, we can evaluate the students who have not registered for this course.

**Q. Show best practice(s) when you can solve the problems with multiple approaches. If possible comment why one approach is better than the others.**

**A.** The same implementation can be achieved through Spring MVC as well. The difference would be to use DAO classes where all the transactions and interaction with database would occur. There would be a DAO class where our session object would be created and all other DAO classes such as StudentDAO, CourseDAO will get the session from DAO and create query in a begin() and commit() statements(transactions).

The MVC approach is very detailed as it demonstrates how the Model View and Controller all interact with each other to achieve the goal. With Spring Boot the functionality of DAO is inbuilt and we use repositories to carry out any of the CRUD operations on our data. To find a student by ID we s\just use the inbuilt method of boot findById(id) to fetch the student with a specific ID.

**Spring BOOT better than Spring MVC**

1. Spring BOOT could be a better option than Spring MVC as Spring Boot sits on top of Spring MVC and it takes care of all the configurations, jars required to develop web application.
2. Whereas Spring MVC has a lot of configurations to do like view resolvers, dispatcher servlet, web jars(for dependencies). Spring boot is the solution for this.
3. Spring Boot provides us starters to build our web applications easily. Starters such as tomcat server, validations, bindings.

**Q. We love Hibernate and also hate it!! Would love to hear your opinion on best practices.**

**A.** Best practices of hibernate are as follows:

1. Use String constants for every element used in our business code**.** Substitute those where required in classes wherever required. This makes code easier to understand and debug. If we need to change the string value which applies at several places we just need to make change to the String constant only.

2. JPA clients such as hibernate needs to add a dependency in the pom.xml and it will take care of all the configurations required to achieve the desired result such as using sql database in our project.

3. Annotations is one of the most important practice used as it gives a good understanding of each entity such as @Id, @Column, @Autowired, @Repository. Hibernate also understands the implementation of getter and setter methods to get and set values of an attribute.

4. Using bind parameters(:) provides benefit when we execute the query. Hence easier to understand.