2021_25_IV_Rest API G2 CA2_Slot 3

Test Summary

- No. of Sections: 2
- No. of Questions: 10
- Total Duration: 90 min

Section 1 - MCQ

Section Summary

- No. of Questions: 9
- Duration: 15 min

Additional Instructions:

None

Q1. Which of the following is a valid JPQL function?

- LENGTH
- MAX
- SUM
- BOTH LENGTH AND SUM

Q2. Which JPA annotation is used to create a one to one relationship between two entities?

- OneToOne
- oneToone
- One-To-One
- 1-To-1

Q3. Which method is used to execute a named query in JPA?

- executeQuery()
- getResultList()
- getSingleResult()
- All of the above

Q4. What is a One-to-Many relationship in JPA?

A relationship between two entities where one entity can have multiple instances of the other entity

A relationship between two entities where one entity can have only one instance of the other entity

A relationship between three entities where one entity can have multiple instances of the other two entities

A relationship between two entities where both entities can have multiple instances of each other

Q5. In a One-to-Many relationship, which entity is typically the owning entity?

The entity with the @ManyToOne annotation

The entity with the @OneToMany annotation

Both entities are considered owning entities

None of the entities are considered owning entities

Q6. How do you define a bidirectional relationship in JPA?

By using the @ManyToOne annotation in one entity and the @OneToMany annotation in the other entity.

By using the @OneToOne annotation in both entities

By using the @ManyToMany annotation in both entities.

None of the above

Q7. What is the purpose of the EntityManager in JPA?

To manage entity instances during runtime

To define the structure of the database schema

To map entity classes to database tables

To generate SQL queries automatically

Q8. Which of the following is not a valid JPQL query?

SELECT e FROM Employee e

SELECT e.name FROM Employee e

SELECT name FROM Employee

SELECT COUNT(e) FROM Employee e

Q9. Which annotation is used to specify a named query in JPA?

@NamedQuery

@Named

@NamedPersonQuery

@NamedNativeQuery

Section 2 - Coding

Section Summary

- No. of Questions: 1
- Duration: 75 min

Additional Instructions:

None

Q1. **Overview:**
Get the employee details using JPQL Query operations.

Functional Requirements:
Create 4 folders inside the **WORKSPACE/springapp/src/main/java/com/example/springapp**
1. Controller
2. Model
3. Repository
4. Service

Inside the controller, create a Java file named “ApiController.java”
Inside the model, create a Java file named “Employee.java”
Your table name should be **employee**.

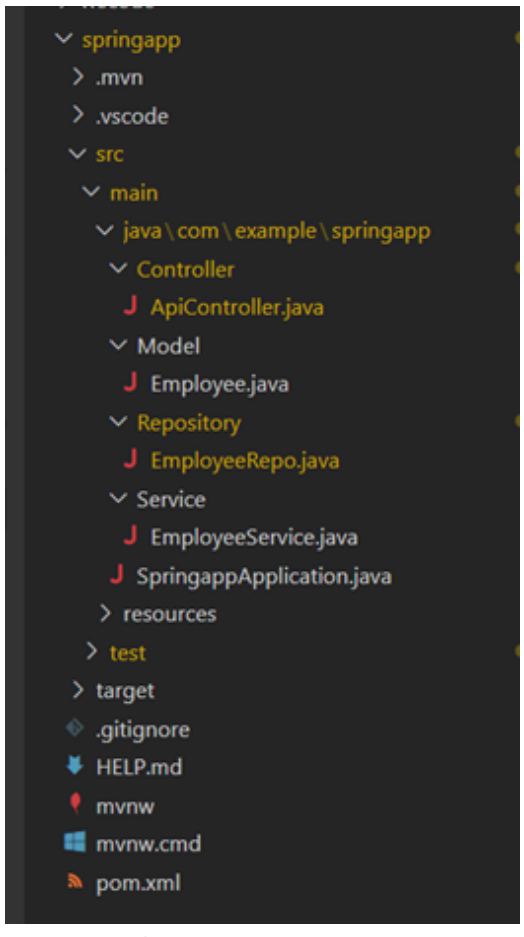
Create 9 variables

- 1. id - int
- 2. name- string
- 3. phoneNumber- long
- 4. address- string
- 5. email- string
- 6. jobtitle - string
- 7.department - string
- 8.salary-long
- 9.hiredate-string

as well as create getters and setters and constructors for the corresponding variables.

- Inside the Repository, create a Java file named "EmployeeRepo.java"
- Inside the service, create a Java file named "EmployeeService.java"

The project structure looks like this image



Core Platform
OpenJDK 11

API:
Question 1 (16 Marks)
POST - "/" --> true/false
GET - "/" --> List of employee object

Question 2 (16 Marks)
GET - "/{id}" --> Employee object
GET - “/findBy/{name}” --> Employee object

Note:
Copy and paste it into the application.properties file

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost/bike?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=examly
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql= true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

API endpoint:
8080

Platform Guidelines:
To run the command use Terminal in the platform.

Spring Boot:
Navigate to the springapp directory => **cd springapp**
To start/run the application '**mvn spring-boot:run**'
Click on the Run Test Case button to pass all the test cases

Answer Key & Solution

Section 1 - MCQ

Q1 BOTH LENGTH AND SUM

Solution

No Solution

Q2 OneToOne

Solution

No Solution

Q3 All of the above

Solution

No Solution

Q4 A relationship between two entities where one entity can have multiple instances of the other entity

Solution

No Solution

Q5 The entity with the @ManyToOne annotation

Solution

No Solution

Q6 By using the @ManyToOne annotation in one entity and the @OneToMany annotation in the other entity.

Solution

No Solution

Q7 To manage entity instances during runtime

Solution

No Solution

Q8 SELECT name FROM Employee

Solution

No Solution

Q9 @NamedQuery

Solution

No Solution

Section 2 - Coding

Q1 Solution cannot be displayed