Spring Data JPA

Question **1**

Correct

Mark 1.00 out of 1.00

Question text

@GenerateValue can be used with non primary key attributes? State True or False.

Select one:

True

False

Feedback

The correct answer is 'True'.

Comments

Question **2**

Correct

Mark 1.00 out of 1.00

Question text

Which of the following is the name of hibernate configuration xml file?

Select one:

None of the listed options

hibernate.cfg.xml

hibernate.configuration.xml

hibernate.config.xml

hibernate.xml

Feedback

The correct answer is: hibernate.cfg.xml

Comments

Question **3**

Correct

Mark 1.00 out of 1.00

Question text

John is a freelance developer he got a requirement to develop an "Pizza order Delivery System" Application and client suggested to use mysql database and Spring data JPA. He started setting configuration in maven POM.xml for a spring boot application. Which of the following spring boot dependencies must be added?

Select one:

None of the listed options

spring-boot-starter-data-jpa  
mysql-connector

spring-boot-starter-data-jpa  
hibernate mysql-connector

spring-boot-starter-data-jpa

Feedback

The correct answer is: spring-boot-starter-data-jpa  
mysql-connector

Comments

Question **4**

Correct

Mark 1.00 out of 1.00

Question text

John is an application developer and he is currently working with DAO/Repository layer of web service called EmployeeService which is developed using Spring Boot Data JPA. Following is the code of Employee Entity.  
  
Choose the correct code snippet to define the @Query to fetch the details of the employee's whose salary falls between a given range.  
  
@Data  
@Entity  
public class Employee {  
  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
  
@Column(name = "first\_name")  
private String firstName;  
  
@Column(name = "last\_name")  
private String lastName;  
  
@Column(name="date\_of\_birth")  
private LocalDate dateOfBirth;  
  
@Column(name="date\_of\_join")  
private LocalDate dateOfJoin;  
  
@Column(name = "salary")  
private double salary;  
}

Select one:

@Query("select e from Employee e where e.salary between ?1 to ?2")  
public List<Employee> getAllEmployeeBySalaryBetween(double minSalary ,double maxSalary);

@Query("select e from Employee e where e.salary between ?1 and ?2")  
public List<Employee> getAllEmployeeBySalaryBetween(double minSalary ,double maxSalary);

None of the listed options

@Query("select e from Employee e where e.salary >=?1 and <=?2")  
public List<Employee> getAllEmployeeBySalaryBetween(double minSalary ,double maxSalary);

Feedback

The correct answer is: @Query("select e from Employee e where e.salary between ?1 and ?2")  
public List<Employee> getAllEmployeeBySalaryBetween(double minSalary ,double maxSalary);

Comments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3** | **21/02/22, 14:25** | **Attempt finished** | **Correct** | **1.00** |

Question **5**

Incorrect

Mark 0.00 out of 1.00

Question text

Which repository is used in order to provide basic DML functionalities ?

Select one:

BasicRepository

None of the listed options

CRUDRepository

DMLRepository

Feedback

The correct answer is: CRUDRepository

Comments

Incorrect

Mark 0.00 out of 1.00

Question text

You are tasked to develop a Spring Data JPA application to manage the employees of an organization. You are working with Spring Boot Framework in order to implement the employee application. Choose the correct code snippet to define the spring boot application properties such as database url and database driver from below.

Select one:

database.url=jdbc:mysql://localhost:3306/test  
database.driverClassName=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/test  
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

spring.database.url=jdbc:mysql://localhost:3306/test  
spring.database.driverClassName=com.mysql.cj.jdbc.Driver

None of the listed options

Feedback

The correct answer is: spring.datasource.url=jdbc:mysql://localhost:3306/test  
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

Comments

Question **7**

Incorrect

Mark 0.00 out of 1.00

Question text

You are tasked to develop a Hibernate application to manage the employees of an organization. Choose the correct code snippet to define the SessionFactory in hibernate.cfg.xml from below.

Select one:

<hibernate-configuration>  
<session-factory>  
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
<property name="hibernate.connection.url">jdbc:mysql://localhost/test</property>  
<property name="hibernate.connection.username">root</property>  
<property name="hibernate.connection.password">mysql</property>  
<mapping resource="com/example/hibernate/Employee.hbm.xml" />  
</session-factory>  
</hibernate-configuration>

None of the listed options

<hibernate-configuration>  
<session-factory>  
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
<property name="hibernate.connection.username">root</property>  
<property name="hibernate.connection.password">mysql</property>  
<mapping resource="com/example/hibernate/Employee.hbm.xml" />  
</session-factory>  
</hibernate-configuration>

<hibernate-configuration>  
<session-factory>  
<property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>  
<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
<property name="hibernate.connection.url">jdbc:mysql://localhost/test</property>  
<property name="hibernate.connection.username">root</property>  
<property name="hibernate.connection.password">mysql</property>  
</session-factory>  
</hibernate-configuration>

<hibernate-configuration>  
<session-factory>  
<property name="dialect">org.dialect.MySQLDialect</property>  
<property name="connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
<property name="connection.url">jdbc:mysql://localhost/test</property>  
<property name="connection.username">root</property>  
<property name="connection.password">mysql</property>  
<mapping resource="com/example/hibernate/Employee.hbm.xml" /></session-factory>  
</hibernate-configuration>

Feedback

The correct answer is: <hibernate-configuration>  
<session-factory>  
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
<property name="hibernate.connection.url">jdbc:mysql://localhost/test</property>  
<property name="hibernate.connection.username">root</property>  
<property name="hibernate.connection.password">mysql</property>  
<mapping resource="com/example/hibernate/Employee.hbm.xml" />  
</session-factory>  
</hibernate-configuration>

Comments

Question **8**

Correct

Mark 1.00 out of 1.00

Question text

The Session is Single Threaded Object. State True or False.

Select one:

True

False

Feedback

The correct answer is 'True'.

Comments

Question **9**

Incorrect

Mark 0.00 out of 1.00

Question text

John is working on simple requirement fetch the details of employees whose salary is between the range of 20000 and 60000 using jpa query methods for an Entity(eno,name,address,sal,dateOfJoin,exp).  
Choose the correct customized Spring Data Jpa Query generation method to be added to the repository.  
@Repository  
public interface EmployeeRepository extends JpaRepository<Long, Emp> {}

Select one:

List<Emp> findBySalBetween(int sal1,int sal2);

List<Emp> findBySalGraterThanAndLessThan(int sal1,int sal2)

List<Emp> findBySalBetweenRange(int sal1,int sal2);

List<Emp> findBySalIn(int sal1,int sal2);

None of the listed options

Feedback

The correct answer is: List<Emp> findBySalBetween(int sal1,int sal2);

Comments

Question **10**

Correct

Mark 1.00 out of 1.00

Question text

How can you configure JPQL query for a query method in a repository in Spring Data JPA?

Select one:

Option1 and Option2

Option 1: Using @Query annotation

Option 3: using @GenerateQuery

None of the listed options

Option 2: Using @NamedQuery annotation

Feedback

The correct answer is: Option 1: Using @Query annotation

Comments

Question **11**

Correct

Mark 1.00 out of 1.00

Question text

John is working on simple requirement fetch the details of the employee who draws the highest salary using JpaRepository methods for an Entity(eno,name,address,sal,dateOfJoin,exp).  
Choose the correct statement for the given requirement.

Select one:

employeeRepository.findAll(Sort.by("sal").descending()).findFirst();

None of the listed options

employeeRepository.findAll(Sort.by("sal").descending()).stream().findFirst().get();

employeeRepository.findAll(Sort.by("sal").descending()).stream().findLast().get();

Feedback

The correct answer is: employeeRepository.findAll(Sort.by("sal").descending()).stream().findFirst().get();

Comments

Question **12**

Incorrect

Mark 0.00 out of 1.00

Question text

What is the default FetchType for @ManyToMany Association?

Select one:

EAGER

LAZY

None of the listed options

Feedback

The correct answer is: LAZY

Comments

Question **13**

Correct

Mark 1.00 out of 1.00

Question text

John is an application developer and he is currently working with DAO/Repository layer of web service called EmployeeService which is developed using Spring Boot Data JPA. Following is the code of Employee Entity.  
Requirement is to fetch the employee details based on their employee id and date of joining.  
Choose the correct code snippet to define the Query method from below.  
  
@Data  
@Entity  
public class Employee {  
  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
  
@Column(name = "first\_name")  
private String firstName;  
  
@Column(name = "last\_name")  
private String lastName;  
  
@Column(name="date\_of\_birth")  
private LocalDate dateOfBirth;  
  
@Column(name="date\_of\_join")  
private LocalDate dateOfJoin;  
  
@Column(name = "salary")  
private double salary;  
}

Select one:

None of the listed options

All of the listed options

@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer>  
{  
public Employee findByEmployeeIdAndDateOfJoinBetween(int id, LocalDate startDate, LocalDate endDate);}

@Repository public interface EmployeeRepository extends JpaRepository<Employee, Integer>  
{  
public Employee findByIdAndDateOfJoinBetween(int id, LocalDate startDate, LocalDate endDate);  
}

@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer>{  
  
public List<Employee> findByEmployeeIdAndDateOfJoinBetween(int id, LocalDate startDate, LocalDate endDate);}

Feedback

The correct answer is: @Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer>  
{  
public Employee findByEmployeeIdAndDateOfJoinBetween(int id, LocalDate startDate, LocalDate endDate);}

Comments

Question **14**

Correct

Mark 1.00 out of 1.00

Question text

You are tasked to develop a JPA application to manage the employees of an organization. You are working with Spring Data JPA in order to implement the employee application.Choose the correct code snippet to define the Entitles from below  
SQL Script  
CREATE TABLE employee  
(employee\_id int NOT NULL,employee\_name varchar(255) DEFAULT NULL, PRIMARY KEY (employee\_id)) ;  
CREATE TABLE department (department\_id int NOT NULL, department\_name varchar(255) DEFAULT NULL, PRIMARY KEY (department\_id));  
CREATE TABLE department\_employees (department\_department\_id int NOT NULL, employees\_employee\_id int NOT NULL, UNIQUE KEY UK\_employee\_id (employees\_employee\_id),KEY FK\_dept\_dept\_id (department\_department\_id),CONSTRAINT FK\_emp\_id FOREIGN KEY (employees\_employee\_id) REFERENCES employee (employee\_id),CONSTRAINT FK\_dept\_dept\_id FOREIGN KEY (department\_department\_id) REFERENCES department (department\_id)) ;

Select one:

@ToString  
@Data  
@Entity  
public class Employee {  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
@Column(name = "employee\_name")  
private String employeeName;  
}  
  
@Data  
@ToString  
@Entity  
public class Department {  
@Id  
@Column(name = "department\_id")  
private int departmentId;  
@Column(name = "department\_name")  
private String departmentName;  
@OneToMany  
private Collection<Employee> employees;}

@ToString  
@Data  
@Entity  
public class Employee {  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
@Column(name = "employee\_name")  
private String employeeName;  
}  
  
@Data  
@ToString  
@Entity  
public class Department {  
@Id  
@Column(name = "department\_id")  
private int departmentId;  
@Column(name = "department\_name")  
private String departmentName;  
@ManyToOne  
private Collection<Employee> employees;}

@ToString  
@Data  
@Entity  
public class Employee {  
@Column(name = "employee\_id")  
private int employeeId;  
@Column(name = "employee\_name")  
private String employeeName;  
}  
  
@Data  
@ToString  
@Entity  
public class Department {  
@Column(name = "department\_id")  
private int departmentId;  
@Column(name = "department\_name")  
private String departmentName;  
@OneToMany  
private Collection<Employee> employees;}

@ToString  
@Data  
public class Employee {  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
@Column(name = "employee\_name")  
private String employeeName;  
}  
  
@Data  
@ToString  
public class Department {  
@Id  
@Column(name = "department\_id")  
private int departmentId;  
@Column(name = "department\_name")  
private String departmentName;  
@OneToMany  
private Collection<Employee> employees;}

Feedback

The correct answer is: @ToString  
@Data  
@Entity  
public class Employee {  
@Id  
@Column(name = "employee\_id")  
private int employeeId;  
@Column(name = "employee\_name")  
private String employeeName;  
}  
  
@Data  
@ToString  
@Entity  
public class Department {  
@Id  
@Column(name = "department\_id")  
private int departmentId;  
@Column(name = "department\_name")  
private String departmentName;  
@OneToMany  
private Collection<Employee> employees;}

Comments

Question **15**

Correct

Mark 1.00 out of 1.00

Question text

Which of the following statements are true about spring.jpa.hibernate.ddl-auto=validate?

Select one:

None of the listed options

Hibernate validates whether the tables and columns exist otherwise it will create tables;

Hibernate validates whether the tables and columns exists otherwise it will throw an exception.

By default hibernate will validate the tables and columns if we are not setting additional property.

Feedback

The correct answer is: Hibernate validates whether the tables and columns exists otherwise it will throw an exception.

Comments