Spring Data JPA Interview Questions and Answers

Table of Contents

* [**What is Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_is_Spring_Data_JPA)
* [**How to create a custom repository in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_create_a_custom_repository_in_Spring_Data_JPA)
* [**How you will write custom method in the repository in Spring Data JPA? What are rules to define Query methods(query creation from method names)?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_you_will_write_custom_method_in_the_repository_in_Spring_Data_JPA_What_are_rules_to_define_Query_methods(query_creation_from_method_names))
* [**What are the important predefined repository interfaces and classes in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_are_the_important_predefined_repository_interfaces_and_classes_in_Spring_Data_JPA)
* [**What is the hierarchy of repository interfaces/classes in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_is_the_hierarchy_of_repository_interfaces/classes_in_Spring_Data_JPA)
* [**What are the important methods of Crudrepository to perform CRUD operations in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_are_the_important_methods_of_Crudrepository_to_perform_CRUD_operations_in_Spring_Data_JPA)
* [**What are the features/benefits of Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_are_the_features/benefits_of_Spring_Data_JPA)
* [**How to enable Spring Data JPA features.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_enable_Spring_Data_JPA_features)
* [**Define a simple repository method which returns all records/entities for a given name?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Define_a_simple_repository_method_which_returns_all_records/entities_for_a_given_name)
* [**How to define Query Methods for the nested property.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_define_Query_Methods_for_the_nested_property)
* [**Write JPQL using @Query annotation in Spring Data JPA.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Write_JPQL_using_@Query_annotation_in_Spring_Data_JPA)
* [**@NamedQuery vs @NamedNativeQuery in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#@NamedQuery_vs_@NamedNativeQuery_in_Spring_Data_JPA)
* [**Difference between CrudRepository and JpaRepository in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Difference_between_CrudRepository_and_JpaRepository_in_Spring_Data_JPA)
* [**Difference between Repository and CrudRepository in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Difference_between_Repository_and_CrudRepository_in_Spring_Data_JPA)
* [**How CrudRepository save() methods internally works in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_CrudRepository_save()_methods_internally_works_in_Spring_Data_JPA)
* [**Tell something about the CrudRepository saveAll() method.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Tell_something_about_the_CrudRepository_saveAll()_method)
* [**How to write a query method for sorting using Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_write_a_query_method_for_sorting_using_Spring_Data_JPA)
* [**How to implement projection using Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_implement_projection_using_Spring_Data_JPA)
* [**How to write query using @NamedQueries and @NamedNativeQueries in Spring Data JPA.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_write_query_using_@NamedQueries_and_@NamedNativeQueries_in_Spring_Data_JPA)
* [**Difference between findById() and getOne() in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Difference_between_findById()_and_getOne()_in_Spring_Data_JPA)
* [**Explain the basic flow of Basic Spring Data JPA Flow.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Explain_the_basic_flow_of_Basic_Spring_Data_JPA_Flow)
* [**Difference between delete() vs deleteInBatch() Methods in Spring Data JPA.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Difference_between_delete()_vs_deleteInBatch()_Methods_in_Spring_Data_JPA)
* [**Difference between deleteAll() Vs deleteAllInBatch() in Spring Data JPA.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#Difference_between_deleteAll()_Vs_deleteAllInBatch()_in_Spring_Data_JPA)
* [**How to write named parameters in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_write_named_parameters_in_Spring_Data_JPA)
* [**What will happen when we define wrong Query Methods in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#What_will_happen_when_we_define_wrong_Query_Methods_in_Spring_Data_JPA)
* [**How to define case insensitive search Query Methods in Spring Data JPA?**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#How_to_define_case_insensitive_search_Query_Methods_in_Spring_Data_JPA)
* [**List of important keywords and corresponding Query Methods.**](https://www.netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/#List_of_important_keywords_and_corresponding_Query_Methods)

What is Spring Data JPA?

Spring Data JPA is one of Spring Data module which provides predefined repository methods to perform CRUD operation. Using Spring Data JPA we define the repository interface and query methods(query creation from method names) to access the data from the database. It makes easier to build Spring applications that use data access technologies.

How to create a custom repository in Spring Data JPA?

We can create custom repository extending any of these interfaces according to need.

Repository  
CrudRepository  
PagingAndSortingRepository  
JpaRepository  
QueryByExampleExecutor

StudentRepository.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | @Repository  public interface StudentRepository extends JpaRepository<Student, Serializable> {    public List<Student> findByNameAndRollNumber(String name, String rollNumber);    public List<Student> findByNameOrRollNumber(String name, String rollNumber);    public List<Student> findByNameAndRollNumberOrUniversity(String name, String rollNumber, String university);    } |

See more details [**here**](https://netsurfingzone.com/jpa/how-to-create-a-custom-repository-in-spring-data-jpa/).

How you will write custom method in the repository in Spring Data JPA? What are rules to define Query methods(query creation from method names)?

Consider we have an entity called Student.java and we have some records in the database as below.

Student.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package com.netsurfingzone.entity;    @Entity  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;      } |

To write query methods first we need to define repository interface.

|  |  |
| --- | --- |
| 1  2  3  4 | @Repository  public interface StudentRepository extends JpaRepository<Student, Serializable> {    } |

Rules to define Query methods.

Rule 1 – The name of the query method must start with findBy or getBy  or queryBy or countBy or readBy prefix. The findBy is mostly used by the developer.

For example findByName(String name), getByName(String name), queryByName(String name), countByName(String name), readByName(String name),

Note – All the above query methods will return all students whose name is “what ever name we provide as parameter”.

Rule 2 – The first character of field name should capital letter. Although if we write the first character of the field in small then it will work but we should use camelcase for the method name.

Both are valid query method defined below but we should follow the first way.

public List<Student> findByName(String name);

public List<Student> findByname(String name);

Rule 3 – While using findBy or getBy or queryBy or countBy or readBy the character B must be in capital letter, else we will get an exception while deployment.

Caused by: org.springframework.data.mapping.PropertyReferenceException: No property querybyName found for type Student!

Invalid query method.

public List<Student> findbyName(String name);

Rule 4 – We can write the query method using multiple fields using predefined keywords(eg. And, Or etc) but these keywords are case sensitive. We must use “And” instead of “and”.

**Write query method using @Query.**

Writing JPQL using Spring Data Jpa @Query.

@Query("select s from Student s where s.name = ?1")  
List<Student> getStudents(String name);

See a complete example of the JPQL using Spring Boot and Oracle [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-jpql-and-native-query-example/).

**Writing the Named Parameter @Query.**

@Query("select s from Student s where s.name = :name")  
List<Student> findByName(@Param("name") String name);

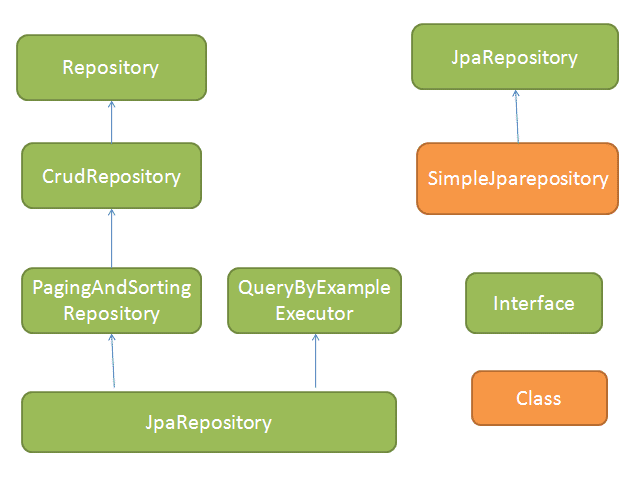
See a complete example of the Named Parameter using Spring Boot and Oracle [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-named-parameters/).

What are the important predefined repository interfaces and classes in Spring Data JPA?

* Repository – Top-level interface defined in Spring Data Hierarchy. This is a marker interface i.e doesn’t contain any method. See more details about the Repository interface [**here**](https://netsurfingzone.com/jpa/difference-between-repository-and-crudrepository/).
* CrudRepository – The CrudRepository interface extends Repository interface, provides methods to perform CRUD operation. See more details about the CrudRepository interface [**here**](https://netsurfingzone.com/jpa/crudrepository-methods-example/).
* PagingAndSortingRepository – The PagingAndSortingRepository interface extends CrudRepository interface and provides additional methods to retrieve entities using the pagination and sorting.
* QueryByExampleExecutor – The QueryByExampleExecutor interface used to execute Query by Example.
* JpaRepository – The JpaRepository interface extends PagingAndSortingRepository and QueryByExampleExecutor interface, provides some additional batch methods. See more details [**here**](https://netsurfingzone.com/jpa/difference-between-crudrepository-and-jparepository-in-spring-data-jpa/).
* SimpleJpaRepository – The SimpleJpaRepository is the implementation class of the CrudRepository interface.
* QueryDslJpaRepository – This is a class.

What is the hierarchy of repository interfaces/classes in Spring Data JPA?

The Repository (marker interface) is the top-level interface defined in Spring Data Hierarchy.

**[](https://netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/attachment/jpaapi1/)**

What are the important methods of Crudrepository to perform CRUD operations in Spring Data JPA?

The Crudrepository contains below methods.

* save(S entity) – Used to save a single entity at a time. See an example of the save() method [**here**](https://netsurfingzone.com/hibernate/spring-data-crudrepository-save-method/).
* saveAll(Iterable<S> – we can save multiple entities at a time. See a complete example of the CrudRepository saveAll() method from scratch [**here**](https://netsurfingzone.com/hibernate/spring-data-crudrepository-saveall-and-findall/).
* findById(ID id) – Use to get entity basis of id. See an example of the findById() method [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-finbyid/).
* existsById(ID id) – Used to check whether an entity is already exited in DB for given Id. See an example of the existsById() method [**here**](https://netsurfingzone.com/jpa/spring-data-crudrepository-existsbyid/).
* findAll() – Find all entity of particular type. See an example of the findAll() [**here**](https://netsurfingzone.com/hibernate/spring-data-crudrepository-saveall-and-findall/).
* findAllById(Iterable<ID> ids)  – Return all entity of given ids. See an example [**here**](https://netsurfingzone.com/jpa/crudrepository-findallbyid-example-using-spring-boot/).
* count() – Returns the number of entities. See an example of the count() method [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-count-example/).
* deleteById(ID id) – Delete the entity on basis of id. See a complete example deleteById() method [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-deletebyid-example/).
* delete(T entity) – delete the entity which one we are passing. See a complete example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-delete-and-deleteall/).
* delete(Iterable<? extends T> entities) – delete multiple entities which we are passing.  See a complete example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-delete-and-deleteall/).
* deleteAll() – delete all entities. See a complete example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-crudrepository-delete-and-deleteall/).

What are the features/benefits of Spring Data JPA?

Spring Data JPA provides features to Query creation from method names. For example, consider we have a method defined in Studentrepository  public List<Student> findByName(String name);  and we have an entity Student.java as below.

StudentRepository.java



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @Repository  public interface StudentRepository extends CrudRepository<Student, Serializable> {    public List<Student> findByName(String name);    } |

Student.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Entity  public class Student {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;  @Column(name = "name")  private String name;  @Column(name = "roll\_number")  private String rollNumber;  @Column(name = "university")  String university;  } |

The following query will get created.

Hibernate:  
select  
student0\_.id as id1\_0\_,  
student0\_.name as name2\_0\_,  
student0\_.roll\_number as roll\_number3\_0\_,  
student0\_.university as university4\_0\_  
from  
student student0\_  
where  
student0\_.name=?

Easy to define query methods for data access. For example, public List<Student> findByName(String name); will return all students for the given name. Spring Data JPA provides predefined keywords using that we can write different query methods.

Let’s see some important keywords and corresponding Query Methods.

| **Keyword** | **Query Method** | **Generated Query Sample** |
| --- | --- | --- |
| And | List findByNameAndRollNumber(String name, String rollNumber) | here student0\_.name=? and student0\_.roll\_number=? |
| Or | List findByNameOrRollNumber(String name, String rollNumber) | where student0\_.name=? or student0\_.roll\_number=? |
| Beetween | List findByRollNumberBetween(String start, String end) | where student0\_.roll\_number between ? and ? |

See more example of each keyword separately [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-query-methods/).

Since CrudRepository provides predefined methods and we can define query methods(for accessing data) which reduce code size and Using Spring Data Jpa we can remove a lot of code by creating an abstract repository class that provides CRUD operations for our entities.

We can complex query using @Query annotation in Spring Data JPA. See an example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-query-annotation/).

We can also write JPQL and Native Query with entity using @NamedQuery and @NamedNativeQuery annotation. See an Example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-namedquery/).

Spring Data Jpa provides PagingAndSortingRepository interface for sorting and pagination support. See an Example [**here**](https://netsurfingzone.com/jpa/sorting-and-pagination-in-spring-data-jpa/).

Spring Data Jpa provides Named Parameters interface. See an Example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-named-parameters/).

We are looking at Spring Data JPA Interview Questions and Answers.

How to enable Spring Data JPA features.

First, we need to define some configuration class let’s say JpaConfig.java and then we need to use @EnableJpaRepositories annotation with that class. Also, provide basePackages value where (we have our repositories). Let’s see below example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package com.netsurfingzone.config;    import org.springframework.context.annotation.Configuration;  import org.springframework.data.jpa.repository.config.EnableJpaRepositories;    @Configuration  @EnableJpaRepositories(basePackages = "com.netsurfingzone.repository")  public class JpaConfig {    } |

tails.

Define a simple repository method which returns all records/entities for a given name?

We can define a repository or query method as below. All three methods would do the same task

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | import java.io.Serializable;  import java.util.List;    import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;    import com.netsurfingzone.entity.Student;    @Repository  public interface StudentRepository extends JpaRepository<Student, Serializable> {    public List<Student> findByName(String name);    public List<Student> findByNameIs(String name);    public List<Student> findByNameEquals(String name);    } |

Spring Data JPA Interview Questions and Answers

How to define Query Methods for the nested property.

Consider we have two entities Student.java and Address.java. Student and Address entities are in one to one relationship and we want to fetch all students from the database who belongs to city pune.

Student.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package com.netsurfingzone.entity;    @Entity  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;    @OneToOne(cascade = CascadeType.ALL)  @JoinColumn(name = "address\_id")  Address address;    } |

Address.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package com.netsurfingzone.entity;    @Entity  public class Address {  @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "house\_number")  private String houseNumber;    @Column(name = "city")  private String city;      } |

Now we want all Students records which belong to city pune.

Query method to fetch all students who belong to city pune using Spring Data JPA.

List<Student> findByAddressCity(String city);

See an example [**here**](https://netsurfingzone.com/jpa/spring-data-jpa-nested-property-query-method/).

Write JPQL using @Query annotation in Spring Data JPA.

Writing JPQL using Spring Data Jpa @Query.

@Query("select s from Student s where s.name = ?1")  
List<Student> getStudents(String name);

@NamedQuery vs @NamedNativeQuery in Spring Data JPA?

@NamedQuery and @NamedNativeQuery annotations used with entity class.

@NamedQuery example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package com.netsurfingzone.entity;    @Entity  @NamedQuery(name = "Student.findByName", query = "select s from Student s where s.name = ?1")  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;  } |

@NamedNativeQuery example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package com.netsurfingzone.entity;  @Entity  @NamedNativeQuery(name = "Student.findByName", query = "select \* from Student where name = ?1", resultClass = Student.class)  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;  } |

Difference between CrudRepository and JpaRepository in Spring Data JPA?

| **CrudRepository** | **JpaRepository** |
| --- | --- |
| 1. CrudRepository extends Repository interface. | 1. JpaRepository extends PagingAndSortingRepository and QueryByExampleExecutor interface. |
| 2. CrudRepository provides methods to perform CRUD operations. | 2. JpaRepository provides additional methods like flush(), saveAndFlush(), deleteInBatch() etc. |
| 3. The saveAll(Iterable entities)  method of CrudRepository returns Iterable. | 3. The saveAll(Iterable entities)  method of JpaRepository returns List. |
| 4. If we have to perform mainly CRUD operation, define our repository using CrudRepository. | 4. If we have to perform CRUD as well as Batch operation define our repository extending JpaRepository. |

See more details [**here**](https://netsurfingzone.com/jpa/difference-between-crudrepository-and-jparepository-in-spring-data-jpa/).

Difference between Repository and CrudRepository in Spring Data JPA?

Repository interface.

The Repository is a top-level interface in hierarchy.

The Repository is a marker interface. It doesn’t have any method.

Repository interface has been defined as below.



|  |  |
| --- | --- |
| 1  2  3  4 | @Indexed  public interface Repository<T, ID> {    } |

CrudRepository interface.

The CrudRepository extends Repository interface. It has below methods to perform CRUD operation.

save(S entity)  
saveAll(Iterable<S> entities)  
findById(ID id)  
existsById(ID id)  
findAll()  
findAllById(Iterable<ID> ids)  
count()  
deleteById(ID id)  
delete(T entity)  
delete(Iterable<? extends T> entities)  
deleteAll()

How CrudRepository save() methods internally works in Spring Data JPA?

The CrudRepository’s save() method is used to perform save as well as update operation both. The implementation has been given in SimpleJpaRepository.java, where persist() and merge() is getting called. If we try to save entity first time then persist() method will get invoked and if we try to update the same entity merge() will get invoked.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public S save(S entity) {    if (entityInformation.isNew(entity)) {  em.persist(entity);  return entity;  } else {  return em.merge(entity);  }  } |

See an example [**here**](https://netsurfingzone.com/hibernate/spring-data-crudrepository-save-method/).

Tell something about the CrudRepository saveAll() method.

The CrudRepository saveAll() method used to save multiple entities and internally annotated with @Transactional annotation. It internally uses save() method only as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | @Transactional  public <S extends T> List<S> saveAll(Iterable<S> entities) {    List<S> result = new ArrayList<S>();    for (S entity : entities) {  result.add(save(entity));  }    return result;  } |

See an example [**here**](https://netsurfingzone.com/hibernate/spring-data-crudrepository-saveall-and-findall/).

How to write a query method for sorting using Spring Data JPA?

Consider we have the following entity.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | @Entity  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;    } |

The first scenario **–** Retrieve the data on the basis of one field(i.e university) and sort on the basis of another field(name).

public List<Student> findByUniversity(String university) {  
List<Student> response = studentRepository.findByUniversityOrderByNameAsc(university);  
return response;  
}

The second scenario**–** Retrieve all record(rows)  and sort on the basis of some field(name).

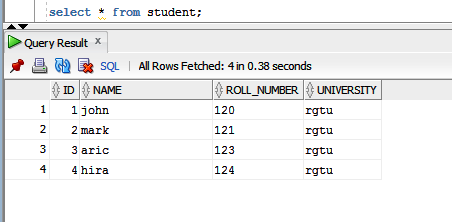
public List<Student> findAll() {  
List<Student> response = (List<Student>) studentRepository.findAllByOrderByNameAsc();  
return response;  
}

How to implement projection using Spring Data JPA?

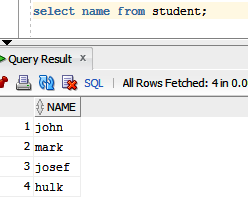
Consider we have an entity called Student.java as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package com.netsurfingzone.entity;    @Entity  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;      } |

and we have some record in the database as below.

**[](https://netsurfingzone.com/jpa/spring-data-jpa-and-or-example-using-spring-boot/attachment/nameandordb/)**

We want to fetch any single column. For example, we want all student name as below.

**[](https://netsurfingzone.com/jpa/spring-data-jpa-projection-example/attachment/projection/)**

Let’s see how to define Projection for the above requirement.

Define Interface with some method.

|  |  |
| --- | --- |
| 1  2  3  4  5 | package com.netsurfingzone.repository;    public interface StudentName {  String getName();  } |

Define Repository as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @Repository  public interface StudentRepository extends JpaRepository<Student, Serializable> {      List<StudentName> findAllBy();    } |

Generated query.

Hibernate:  
select  
student0\_.name as col\_0\_0\_  
from  
student student0\_

Some more Spring Data JPA Interview Questions and Answers.

How to write query using @NamedQueries and @NamedNativeQueries in Spring Data JPA.

Both @NamedQueries and @NamedNativeQueries annotations used with the entity.

Using @NamedQueries.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | package com.netsurfingzone.entity;    @Entity  @NamedQueries({ @NamedQuery(name = "Student.findByName1", query = "select s from Student s where s.name = ?1"),  @NamedQuery(name = "Student.findByNameAndRollNumber", query = "select s from Student s where s.name = ?1 and s.rollNumber = ?2"),  @NamedQuery(name = "Student.findByNameOrRollNumber", query = "select s from Student s where s.name = ?1 or s.rollNumber = ?2") })  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;    } |

Using @NamedNativeQueries.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package com.netsurfingzone.entity;    @Entity  @NamedNativeQueries({  @NamedNativeQuery(name = "Student.findByName", query = "select \* from Student where name = ?1", resultClass = Student.class),  @NamedNativeQuery(name = "Student.findByNameAndRollNumber", query = "select \* from Student where name = ?1 and roll\_number = ?2", resultClass = Student.class),  @NamedNativeQuery(name = "Student.findByNameOrRollNumber", query = "select \* from Student where name = ?1 or roll\_number = ?2", resultClass = Student.class) })  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;    } |

Corresponding repository methods.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | @Repository  public interface StudentRepository extends CrudRepository<Student, Serializable> {    List<Student> findByName(String name);    List<Student> findByNameAndRollNumber(String name, String rollNumber);    List<Student> findByNameOrRollNumber(String name, String rollNumber);    } |

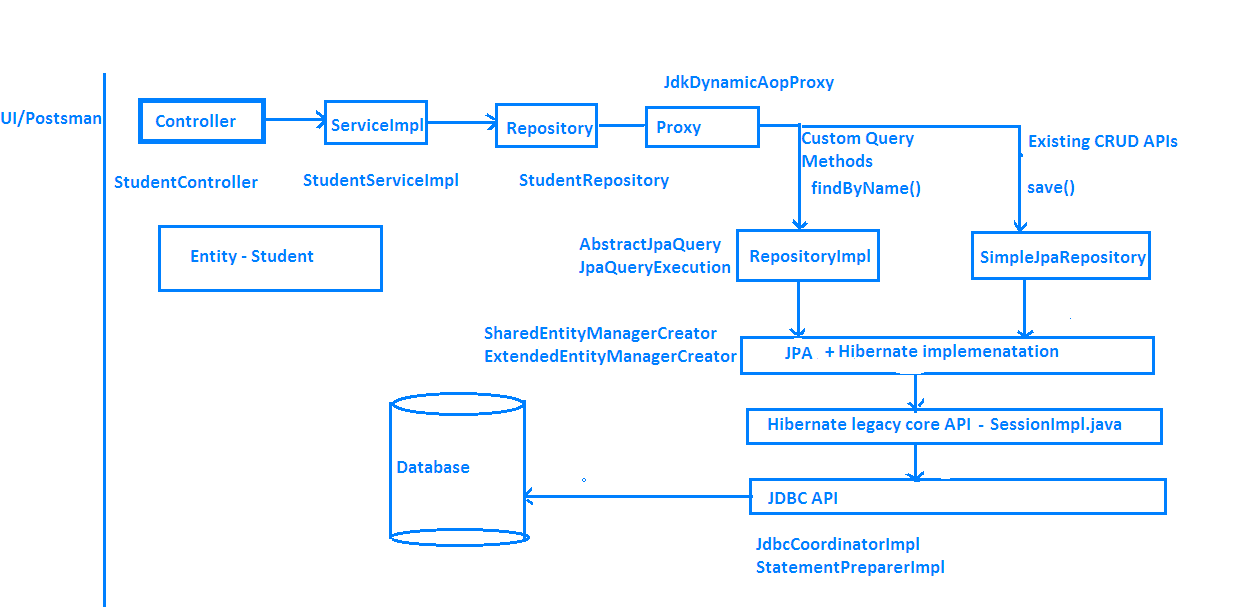
Spring Data JPA Interview Questions and Answers

Difference between findById() and getOne() in Spring Data JPA?

Let’s see some difference between findById() and getOne() methods.

|  |  |
| --- | --- |
| findById() | getOne() |
| 1. The findById() method is available in CrudRepository interface. | 1. The getOne() method is available in JpaRepositpry interface. |
| 2. The findById() method will return null if the record doesn’t exist in the database. | 2. The getOne() method throw EntityNotFoundException if the record doesn’t exist in the database. |
| 3. Internally findById() method use EntityManger find() method. | 3. Internally getOne() method use EntityManger getReference() method. |
| 4. Calling findById() returns a eager fetched entity. | 4. Calling getOne() returns a lazily fetched entity. |

Explain the basic flow of Basic Spring Data JPA Flow.

**[](https://netsurfingzone.com/jpa/spring-data-jpa-interview-questions-and-answers/attachment/springdataflow1-2/)**

Difference between delete() vs deleteInBatch() Methods in Spring Data JPA.

The delete() method has been defined in the CrudRepository interface with the following signature.

void delete(T entity);

The deleteInBatch() has been defined in the JpaRepository interface with the following signature.

void deleteInBatch(Iterable<T> entities);

The delete() method internally uses EntityManager’s remove() method as below.

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Transactional  public void delete(T entity) {    em.remove(em.contains(entity) ? entity : em.merge(entity));  } |

The deleteInBatch() internally implemented as below.

|  |  |
| --- | --- |
| 1  2  3  4  5 | public void deleteInBatch(Iterable<T> entities) {    applyAndBind(getQueryString(DELETE\_ALL\_QUERY\_STRING, entityInformation.getEntityName()), entities, em)  .executeUpdate();  } |

Observe both methods implementation. The first one i.e delete() internally uses remove() method. Before call remove() method it calls contains()/merge() method. Then flow will go inside SessionImpl.java class(fireDelete() method) and a lot of stuff happens.

On the other hand, deleteInBatch() prepares the query and collect some other information and directly calls the executeUpdate() method.

With the help of delete() method, we can delete a single record at a time whereas using deleteInBatch() we can delete multiple records.

The delete() method is a little slower as compare deleteInBatch() as delete() does some extra stuff than deleteInBatch().

Almost done with Spring Data JPA Interview Questions and Answers.

Difference between deleteAll() Vs deleteAllInBatch() in Spring Data JPA.

The deleteAll() method is defined in CrudRepository has two overloaded versions.

First one – void deleteAll(Iterable<? extends T> entities)

Using the above method we can delete all entities which we pass as request data. The deleteAll() internally use delete() method only.

|  |  |
| --- | --- |
| 1  2  3  4  5 | public void deleteAll(Iterable<? extends T> entities) {  for (T entity : entities) {  delete(entity);  }  } |

Spring Data JPA deleteAll() Vs deleteAllInBatch()

Second one – void deleteAll()

The above one will delete all records that belong to that repository. The deleteAll() internally uses findAll() and delete() method as below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public void deleteAll() {    for (T element : findAll()) {  delete(element);  }  } |

The query generated in case of deleteAll(). Suppose we are deleting twelve entities.

Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?  
Hibernate: delete from student where id=?

The deleteAllInBatch() method is defined in the JpaRepository interface as below.

void deleteAllInBatch()

Using deleteAllInBatch() method we can delete all entities from database. No need to pass entities as parameters.For example, let’s say we have an entity Student.java and also we have a corresponding repository class called StudentRepository.java something like below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | package com.netsurfingzone.repository;    @Repository  public interface StudentRepository extends JpaRepository<Student, Serializable> {    } |

If we want to delete all student records we need to do studentRepository.deleteAllInBatch().  See the below sample code.

|  |  |
| --- | --- |
| 1  2  3  4 | @Transactional  public void deleteAllInBatch() {  studentRepository.deleteAllInBatch();  } |

Note – Internally deleteAllInBatch()   uses createQuery() method as below.

|  |  |
| --- | --- |
| 1  2  3 | public void deleteAllInBatch() {  em.createQuery(getDeleteAllQueryString()).executeUpdate();  } |

In case of deleteAllInBatch() the query will generate as below.

Hibernate: delete from student

Both deleteAll() and deleteAllInBatch() use to delete all entities. The deleteAll() internally uses CrudRepository delete() method which further uses EntityManger remove() method which perform some other stuff. But in case of deleteAllInBatch(), EntityManger createQuery() method will preapare the query and it directly calls executeUpdate() method.

How to write named parameters in Spring Data JPA?

@Param and @Query annotations used to define Named Parameters.

Consider we have entity called Student.java.`

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package com.netsurfingzone.entity;    @Entity  public class Student {    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private int id;    @Column(name = "name")  private String name;    @Column(name = "roll\_number")  private String rollNumber;    @Column(name = "university")  String university;      } |

Query methods using named parameter in Spring Data JPA.

@Query("select s from Student s where s.name = :name")  
List<Student> findByName(@Param("name") String name);

@Query("select s from Student s where s.name = :name and s.rollNumber = :rollNumber")  
List<Student> findByNameAndRollNumber(@Param("name") String name, @Param("rollNumber") String rollNumber);

What will happen when we define wrong Query Methods in Spring Data JPA?

Suppose we have an entity called Student.java and Student class has fields called id and name. We want all student records for the given name.

Our query method would be List<Student> findByName(String name);

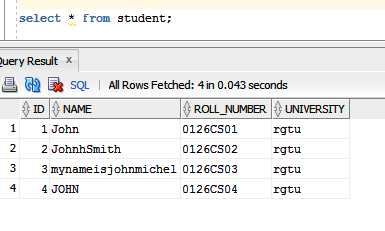
Just think by mistake developer writes findName(). instead of findByName(). In this case, while deployment we will get an error.

org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name ‘studentController’: Unsatisfied dependency expressed through field ‘studentService’; nested exception is org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name ‘studentServiceImpl’: Unsatisfied dependency expressed through field ‘studentRepository’; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name ‘studentRepository’: Invocation of init method failed; nested exception is java.lang.IllegalArgumentException: Failed to create query for method public abstract java.util.List com.netsurfingzone.repository.StudentRepository.findName(java.lang.String)! No property findName found for type Student!

Spring scans all repository methods while deployment itself and checks query methods have been written properly or not.

How to define case insensitive search Query Methods in Spring Data JPA?

Consider we have some records in the database as below.



We have to write a case insensitive search query that means for name “john”, our query method should return two records.

public List<Student> findByNameIgnoreCase(String name);

List of important keywords and corresponding Query Methods.

| **Keyword** | **Query methods** | **JPQL** |
| --- | --- | --- |
| And | findByLastnameAndFirstname | ...where x.lastname = ?1 and x.firstname = ?2 |
| Or | findByLastnameOrFirstname | ...where x.lastname = ?1 or x.firstname = ?2 |
| Is, Equals | findByFirstnameEquals | ...where x.firstname = ?1 |
| Between | findByStartDateBetween | ...where x.startDate between ?1 and ? |
| LessThan | findByAgeLessThan | ...where x.age < ?1 |
| LessThanEqual | findByAgeLessThanEqual | ...where x.age <= ?1< td> |
| GreaterThan | findByAgeGreaterThan | ...where x.age > ?1 |
| GreaterThanEqual | findByAgeGreaterThanEqual | ...where x.age >= ?1 |
| After | findByStartDateAfter | ...where x.startDate > ?1 |
| Before | findByStartDateBefore | ...where x.startDate < ?1 |
| IsNull | findByAgeIsNull | ...where x.age is null |
| IsNotNull, NotNull | findByAge(Is)NotNull | ...where x.age not null |
| Like | findByFirstnameLike | ...where x.firstname like ?1 |
| NotLike | findByFirstnameNotLike | ...where x.firstname not like ?1 |
| StartingWith | findByFirstnameStartingWith | ...where x.firstname like ?1 (parameter bound with appended %) |
| EndingWith | findByFirstnameEndingWith | ...where x.firstname like ?1 (parameter bound with prepended %) |
| Containing | findByFirstnameContaining | ...where x.firstname like ?1 (parameter bound wrapped in %) |
| OrderBy | findByAgeOrderByLastnameDesc | ...where x.age = ?1 order by x.lastname desc |
| Not | findByLastnameNot | ...where x.lastname <> ?1 |
| In | findByAgeIn(Collection ages) | ...where x.age in ?1 |
| NotIn | findByAgeNotIn(Collection ages) | ...where x.age not in ?1 |
| True | findByActiveTrue() | ...where x.active = true |
| False | findByActiveFalse() | ...where x.active = false |
| IgnoreCase | findByFirstnameIgnoreCase | ...where UPPER(x.firstame) = UPPER(?1) |