SpringBoot Data JPA

Jpa Methods:

1.CrudRepository<Synchrony, Integer> :---

1.<S **extends** T> S save(S entity);

To save only one object in DB

EX:

service.save(**new** Synchrony(1, "Hyderabad"));

service.save(**new** Synchrony(1, "Hyderabad"));

2. <S **extends** T> Iterable<S> saveAll(Iterable<S> entities);

To Avoid n/w calls for save bulk objets(inserts)

EX:

service.saveAll(Arrays.*asList*(

**new** Synchrony(100, "Chennai"),

**new** Synchrony(101, "Bangalore")));

3. Iterable<T> findAll();

To Fetch all records at one shot

Ex:

service.findAll().forEach(System.***out***::println);

4. Iterable<T> findAllById(Iterable<ID> ids); Note :Java 9

To fetch some record based on id ex: where id=100,103

Ex:

service.findAllById(List.of(100,103)).forEach(System.***out***::println);

5. Optional<T> findById(ID id);

To fetch only one record based on Id ex: where id=100

Optional<Synchrony>opt= service.findById(1000);

**if**(opt.isPresent())

{

System.***out***.println(opt.get());

}**els**{ System.***out***.println("NO DATA FOUND");}

6.**boolean** existsById(ID id);

Given Id is Exist or not

Ex:

**boolean** exist1 = service.existsById(101);

**boolean** exist2 = service.existsById(109);

System.***out***.println(exist1 +"--"+ exist2);

7. **long** count();

How many record saved in DB

Ex:

service.count();

8. **void** deleteById(ID id);

To delete one record based on PK where id=101

Ex:

service.deleteById(101);

9. **void** delete(T entity); [Sun-JPA Spec]

To delete only one object in DB

Ex:

Synchrony syf = **new** Synchrony();

syf.setSyfId(101);

service.delete(syf);

10. **void** deleteAll(Iterable<? **extends** T> entities);Note:Java 9[Sun-JPA Spec]

To delete multiple rows at a time

EX:

service.deleteAll(

List.of(

**new** Synchrony(105),

**new** Synchrony(101)

)

);

11. **void** deleteAll();

To delete all record at one shot

Ex:

service.deleteAll();

2.PagingAndSortingRepository<Synchrony, Integer> :

12. Iterable<T> findAll(Sort sort);

Here Sort is a class that is used to fetch data based on given property /column name in default ASC order.

Ex:

service.findAll(

Sort.by("empName")

).forEach(System.out::println);;

repo.findAll(

Sort.by(Direction.DESC,"empName")

).forEach(System.out::println);

13. Page<T> findAll(Pageable pageable);

Pagination: It is a process of fetching few rows from DB table using Pagination inputs.

//input for Pagination

//pageNum,pageSize

Pageable pageable = PageRequest.of(2, 3);

//execute and get output

Page<Employee> page = repo.findAll(pageable);

//print data

List<Employee> list = page.getContent(); //page data

list.forEach(System.out::println);

//---output details-------

System.out.println("First?"+page.isFirst()); //boolean

System.out.println("Page Has Data?"+page.hasContent()); //boolean

System.out.println("Last?"+page.isLast()); //boolean

System.out.println("Next?"+page.hasNext()); //boolean

System.out.println("Prev?"+page.hasPrevious()); //boolean

System.out.println("PageNumber?"+page.getNumber());

System.out.println("Size?"+page.getSize()) ; //page size

System.out.println("Total Pages?"+page.getTotalPages()) ; //Total Pages

System.out.println("Total rows?"+page.getTotalElements()) ; //Total Rows

System.out.println("Page Count?"+page.getContent().size()) ; //Total Rows

Q) What is the difference between findAll() and

findAll(PageRequest.of(pageNum, pageSize))?

A) findAll() - select all rows

findAll(PageRequest.of(pageNum, pageSize))

It will select few rows based on pagesize given.

3.JpaRepository<Synchrony, Integer>

14. List<T> findAll();

This method returns all rows from DB table as List Format.

15. <S **extends** T> List<S> findAll(Example<S> example);

This method is used to fetch rows by generating dynamic SQL

with no-null object input.

=> It will compare non-null values(given in object) with every row in DB table

Ex:

repo.findAll().forEach(System.out::println);

//1. create Employe with empDept

Employee emp = new Employee();

emp.setEmpDept("Dev");

//2. convert to Example object using static method of(prob)

Example<Employee> example = Example.of(emp);

//3. execute using findAll

List<Employee> list = repo.findAll(example);

//4. print data

list.forEach(System.out::println);

2. Custom Query Method:

a) findBy [abstract method converted SQL]

b) @Query (HQL/JPQL/SQL) [Manual Query]

A)findBy [abstract method converted SQL]

Basic Syntax:

<RT> findBy<VariablesAndConditions>(<Params>);

=> This concept supports only select operation.

=> Non-select operations like DELETE/INSERT/UPDATE not supported by findBy

=> findBy Variable name must match with model class variable name

else : PropertyReferenceException:

No property eSal found for type Employee! Did you mean 'empSal'?

=> If we did not specify any keyword for condition then default is = (Is,Equals)

Steps to fallow findBy\_\_()

Step 1:create interface extends JpaRep…()

**public** **interface** ISynchronyService **extends** JpaRepository<Synchrony, Integer> {

List<Synchrony> findBySyfId(Integer syfId);

Step 2: To call method in Controller class for interface method

List<Synchrony> i=service.findBySyfId(11);

System.***out***.println(i);

Ex:#1

Step 1:

@Service

**public** **interface** ISynchronyService **extends** CrudRepository<Synchrony, Integer> {

@Query(" FROM Synchrony s WHERE s.syfId=:syfId")

Synchrony getMyData(Integer syfId);

}

Step 2:

Synchrony syf=iSynchronyService.getMyData(20);

System.***out***.println(syf);

Ex:#2

Step 1:

@Service

**public** **interface** ISynchronyService **extends** CrudRepository<Synchrony, Integer> {

@Query("SELECT s.syfLoc FROM Synchrony s WHERE s.syfId=:syfId")

String getDetails(Integer syfId);

}

Step 2:

String syfone=iSynchronyService.getDetails(20);

System.***out***.println(" syfLoc:"+ syfone);

Ex:#3

Step 1:

@Service

**public** **interface** ISynchronyService **extends** CrudRepository<Synchrony, Integer> {

@Query("SELECT s.syfId,s.syfLoc FROM Synchrony s WHERE s.syfId=:syfId")

Object getMultiDetails(Integer syfId);

}

Step 2:

Object ob=iSynchronyService.getMultiDetails(20);

//internally call Object[]

Object[] obArr=(Object[])ob;

System.***out***.println(obArr[0]+"-"+obArr[1]);