JpaRepository methods

<S extends T> List<S> findAll(Example<S> example, Sort sort)

Returns the reords as List<T> objects by taking non-null values of given Example obj . based Entity object data and sorts the selected records accorring to sort object info. uses only non-null values of given entity obj as the criteria values with and clause conditions

=>Example obj is spring data supplied object containing other obejct .. It is like Optional object of Java8 ..

Example obj
Doctor obj
...
... contains data
....

Example example=Example.of(doctor);

Example objet and Optional object both holds other object .. but they utilization is different.

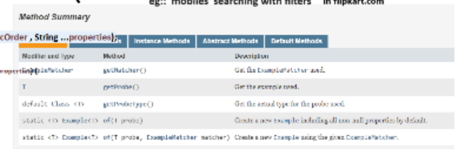
=> All methods of pre-defined Repositories can search /delete/update the records only by taking id value as the criteria value.. where as findAll(Example example) method can search and get the records by taking all Entity obj's non-null property values as the criteria values

Example class in spring data api is given by inspiring from hibernate api

findAll(Example example) is very much use in real project when ever there is a need of generating select query with dynamic conditions

usecases :: Searching products in e-commerce app by applying 0 or more filters

eg:: mobiles searching with filters in flipkart.com



Method Summary

Instance Methods | Abstract Methods | Default Methods

Modifier and Type	Method	Description
ExampleMatcher	getMatcher()	Get the ExampleMatcher used.
T	getProbe()	Get the example used.
default Class <T>	getProbeType()	Get the actual type for the probe used.
static <T> Example<T>	of(T probe)	Create a new Example including all non-null properties by default.
static <T> Example<T>	of(T probe, ExampleMatcher matcher)	Create a new Example using the given ExampleMatcher.

In service Interface

```java
public List<Doctor> showDoctorsByExampleData(Doctor exDoctor, boolean ascOrder , String ...properties);
```

In service Impl class

```java
@Override
	public List<Doctor> showDoctorsByExampleData(Doctor exDoctor, boolean ascOrder, String... properties) {
		//Prepare Sort object
		Sort sort=Sort.by(ascOrder?Direction.ASC:Direction.DESC,properties);
		// Example object
		Example example=Example.of(exDoctor);
		// use the repo
		List<Doctor> list=doctorRepo.findAll(example,sort);
		//return the collection
		return list;
```

In Runner class

```java
Doctor doctor=new Doctor();
		doctor.setSpecialization("cardio"); doctor.setIncome(90000.0);
		service.showDoctorsByExampleData(doctor, true, "income").forEach(System.out::println);
```

note:: The industry standard Repository in Spring Data jpa based projects is JpaRepository

output
========

```
2023-02-08T19:54:04.158+05:30  INFO 24696 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactor
2023-02-08T19:54:04.917+05:30  INFO 24696 --- [           main] roj3PagingAndSortingRepsitoryApplication : Started BootDataJpaProj3PagingAnc
Hibernate: select d1_0.doc_id,d1_0.doc_name,d1_0.income,d1_0.specialization from jpa_doctor_info d1_0 where d1_0.specialization=? and d1
Doctor [docId=5674, docName=suresh, specialization=cardio, income=90000.0]
Doctor [docId=4565, docName=raja, specialization=cardio, income=90000.0]
2023-02-08T19:54:05.194+05:30  INFO 24696 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerF
2023-02-08T19:54:05.199+05:30  INFO 24696 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiate
2023-02-08T19:54:05.216+05:30  INFO 24696 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown comple
```

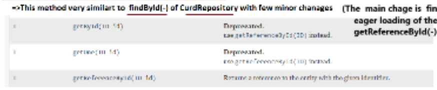
What is the difference among findAll() methods of CrudRepository , PagingAndSortingRepository and JpaRepository

method	Return type	Sorting ability	Pagination Ability	Taking Example obj	Ability to use in SQL andNoSQL Db s/ws
findAll() in CrudRepository	Iterable<T>	no	no	no	yes
findAll() in PASRepository	Iterable<T>	yes	yes	no	yes
findAll() in JpaRepository	List<T>	yes	no	yes	no

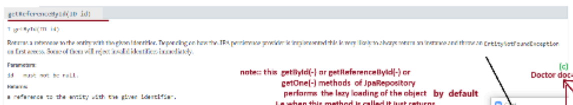
=>While designing Java Bean based Entity classes, DTO classes (Data Transfer Object) It is recommended to take Wrapper type properties becoz they null value representing no value.. if we simple data type properties they take 0 or 0.0 as default which are actually values to consider (These values really becomes problematic while dealing with Example<T> obj based serach actitives)

getReferenceById(-) [alternate to getById(-) and getOne(-) methods of the same JpaRepository)

=>This method very similart to findById(-) of CurdRepository with few minor chanages (The main chage is findById(-) performs eager loading of the object where as getReferenceById(-) performs lazy Loading the object)



T	getById(ID id)	Deprecated. use getReferenceById(ID) instead.
T	getOne(ID id)	Deprecated. use getReferenceById(ID) instead.
T	getReferenceById(ID id)	Returns a reference to the entity with the given identifier.



getReferenceById(ID id)

T getById(ID id)

Returns a reference to the entity with the given identifier. Depending on how the JPA persistence provider is implemented this is very likely to always return an instance and throw an EntityNotFoundException on first access. Some of them will reject invalid identifiers immediately.

Parameters:
id - must not be null.

Returns:
a reference to the entity with the given identifier.

note:: this getById(-) or getReferenceById(-) or getOne(-) methods of JpaRepository performs the lazy loading of the object by default i.e when this method is called it just returns Proxy object (subclass of the entity class obj) having id value (i.e no hit to db s/w) . when non-indentifier method is called on the top of proxy object the real hit to db s/w takes place and the gathered record will be stored in to another Entity class object (real object) that is linked with Proxy object

example
=======

In service Interface

```java
public Doctor findDoctorById(Integer id);
```

In service Impl class

```java
@Override
	public Doctor findDoctorById(Integer id) {
		//Doctor doctor=doctorRepo.getById(id);
		Doctor doctor=doctorRepo.getReferenceById(id);
		return doctor;
	}
```

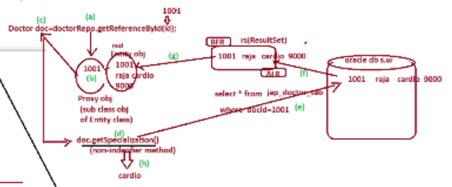
In client App

```java
System.out.println(service.findDoctorById(5674));
```

In application.properties

```
#To enable lazy loading of record in the underlying Hibernate f/w
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

For getOne(-) ,getById(-) and getReferenceById() methods need this property to make underlying Hibernate f/w to support lazy loading even though Transactional env.. is not taken.. In old versions of spring data jpa just @Transacationl in service is class sufficient.



=> The proxy object returned by the getReferenceById(-) method is the object InMemory proxy class that extends from Entity class. if the Entity class is taken as the final class then this InMemory proxy class generation fails. This makes getReferenceById(-) performing earger loading of the object

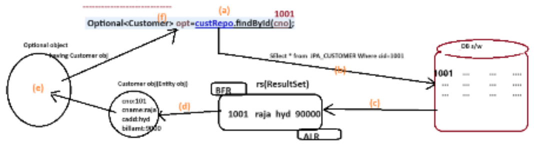
What is the difference b/w getById(-) of JpaRepository or getReferenceById(-) and findById(-) of CrudRepository?

getById(-) / getOne()/getReferenceById()	findById(-) (CrudRepository)
(a) retuns Entity obj ref representing record that is selected for given id (first gives proxy object then gives real entity object)	(a) returns Optional<T> obj having Entity obj for the same (Directly gives Optional object having the Entity obj)
(b) performs Lazy Loading of record/object i.e first returns proxy object and when that proxy is used then record will be retrieved from DB s/w to put into Entity class obj (real obj)	(b) perform eager loading i.e gets the record/object from DB table directly with out involving any proxy object irrespective of wheather that object is used or not
(c) If record is not found we can not throw custom exception .. It gives the fixed EntityNotFoundException	(c) with the support of Optional API we can throw Custom Exception.. or custom message
(d) Works only in SQL Db s/w	(d) Works in both SQL and NO SQL DB s/w
(e) Impl depends on on undelrying Hibernate framework	(e) Impl is given in spring dataJpa itself
(f) addtional property cfg in in applicaiton.properties is reuqired	(f) not required.
(g) if record not found we get exception (EntityNotFoundException)	(g) if record not found, we get Empty Optional object for which we can send custom message or we can throw exception
(h) This method getReferenceById(-) does not actually talks with Db s/w.. The non-indentifier getter methods generated in the proxy class talks with DB s/w by generating SQL Query	(h) This method directly talks with DB s/w by generating the SQL Query
(i) if Entity class is taken as final class then InMemory Proxy class will not be generated So , this method performs only eager loading	(i) This method always performs eager loading irrespective of wheather entity class is taken as final class or not ?
(j) In getReferenceById(-) method call 1 proxy obj and 1 real obj is involved generally	(j) In findById(-) method , only one real object is involved

flow diagram for findById(-) method (eager Loading)
==============================

In service Impl class

```java
Optional<Customer> opt=custRepo.findById(cno);
```



While working with spring Data JPA choose persistence methods in the following order

(a) CrurdRepository methods
==>If not sufficient then
(b) PagingAndSortingRepository methods
==>if not sufficient then
(c) JpaRepository methods
==> If not sufficient then
(d) Custom methods in our Repository Interface.

Common Reposorities

- (i) custom finder methods (only for select opertions) [Also called as custom findBy methods]
- (ii) @Query methods (for HQL/JPQL and Native SQL Queries based custom Select operations)
- (iii) @Query + @Modifying methods (for HQL/JPQL and Native SQL Queries based custom non-Select operations) (insert,update, delete operations)

Best

various options to place custom methods in the our Repository interfaces

HQL :: Hibernate Query Language
JPQL :: Jakarata Persistence API Query Language

What is the difference b/w Example object and Optional Object?

Ans) Example object is useful to hold another Entity object with example data/values using which the findAll(Example) method performs search operation to perform select operation and get records by given Entity data with and clause conditions

Optional object useful to hold another Entity object . This object holds Entity given by findById(-) method

JpaRepository methods

`<S extends T> List<S>`

the

`findAll(Example<5> example, Sort sort)`

Returns reords as List<T> objects by taking non-null values of given Example obj. based Entity object and sorts the selected records accorring

to sort object info. uses only non-null values of given entity obj as the criteria values with and clause conditions

=>Example obj_ is spring data supplied object containing other obejct .. It is like Optional object of Java8..

Example obj Doctor obi

Example example-Example.ofdoctor);

... contains data

=> All methods of pre-defined Repositories

Example objet and Optional object both holds other object .. but they utilization is different.

can search/delete/update the records only by taking

id value as the criteria value.. where as findAll(Example example) method can search and get the records by taking all Entity obj's non-null property values as the criteria values

Example class in spring data api is given

by inspiring from hibernate api

-

much d

findAll(Example example) is very use in real project when ever there is a need of generating select query with dynamic conditions

usecases :: Searching products in e-commerce app by applying 0 or more filters

eg:: mobiles searching with filters

in flipkart.com

In service Interface

Method Summary

public List<Doctor> showDoctorsByExampleData(Doctor exDoctor, boolean ascOrder, String ...properties);

In service Impl class @Override

Instance Methods

Modifier and Type

Method

Abstract Methods Default Methods Description

T

`getMatcher() getProbe()`

`default Class <T>`

getProbeType()

public List<Doctor> showDoctorsByExampleData(Doctor exDoctor, boolean ascOrder, String... properties(pleMatcher //Prepare Sort object

Sort sort=Sort.by(ascOrder?Direction.ASC: Direction.DESC, properties);

// Example object

Example example-Example.of(exDoctor);

// use the repo

List<Doctor> list=doctorRepo.findAll(example,sort);

//return the collection

return list;

In Runner class

Doctor doctor=new Doctor();

```
static <T> Example<T> of (T probe)
```

```
Get the ExampleMatcher used.
```

Get the example used.

Get the actual type for the probe used.

Create a new Example including all non-null properties by default.

```
static <T> Example<T> of (T probe, ExampleMatcher matcher) Create a new Example using
the given ExampleMatcher.
```

note:: The industry standard Repository in Spring Data jpa based projects is JpaRepository

doctor.setSpecialization("cardio"); doctor.setIncome (90000.0);

service.showDoctors ByExampleData(doctor, true, "income").forEach(System.out::println);

output

========

2023-02-08T19:54:04.158+05:30 INFO 24696 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactor 2023-02-08T19:54:04.917+05:30 INFO 24696 --- [main] roj3PagingAndSortingRepsitoryApplication: Started BootDataJpaProj3PagingAnc Hibernate: select d1_0.doc_id,d1_0.doc_name,d1_0.income,d1_0.specialization from jpa_doctor_info d1_0 where d1_0.specialization=? and d1 Doctor [docid=5674, docName=suresh, specialization-cardio, income=90000.0] Doctor [docid=4565, docName=raja, specialization-cardio, income=90000.0]

2023-02-08T19:54:05.194+05:30 INFO 24696 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean: Closing JPA EntityManagerFa 2023-02-08T19:54:05.199+05:30 INFO 24696 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiate 2023-02-08T19:54:05.216+05:30 INFO 24696 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource

What is the difference among findAll() methods of CrurdReposiotry, PagingAndSortingRepository and JpaRepository

: HikariPool-1 - Shutdown comple

method

findAll() in CrudRepository

Ability to

Return type

Sorting ability

Pagination Ability

Taking Example obj

use in SQL and NoSQL DB s/w s

Iterable<T>

no

no

no

yes

findAll() in PASRepository findAll() in JpaRepository

Iterable<T>

yes

yes

no

yes

List<T>

yes

no

yes

no

getReferenceById(-) [alternate to getById(-) and getOne(-) methods of the same JpaRepository)

==========

=>This method very similart to findById(-) of CurdRepository with few minor chanages (The main chage is findById(-) performs

```
T

getById(ID id)

T

getOne (ID id)

getReferenceById(ID id)

getReferenceById(ID id)

T getById(ID id)
```

T

Deprecated.

`use getReferenceById(ID) instead.`

Deprecated.

`use getReferenceById (ID) instead.`

eager loading of the object where as

getReferenceById(-) performs lazy Loading the object)

Returns a reference to the entity with the given identifier.

=>While designing Java Bean based Entity classes, DTO classes (Data Transfer Object) It is recommended to take Wrapper type properties becoz they null value representing no value.. if we simple data type properties they take 0 or 0.0 as default which are actually values to consider (These values really becomes problematic while dealing with Example<T> obj based serach actitives)

1001

(c)

(a)

↓

Doctor doc=doctorRepo.getReferenceById(id);

BFR

rs(ResultSet)

real

Entity obj

(g)

1001 (b)

1001 raja cardio

1001 raja cardio 9000 (f)

oracle db s.w

LAIR

1001

raja cardio 9000

9000

Returns a reference to the entity with the given identifier. Depending on how the JPA persistence provider is implemented this is very likely to always return an instance and throw an EntityNotFoundException on first access. Some of them will reject invalid identifiers immediately.

Parameters:

`id must not be null.`

Returns:

`a reference to the entity with the given identifier.`

example

In service Interface

public Doctor findDoctorById(Integer id);

In service Impl class

@Override

public Doctor findDoctorById(Integer id) {

}

In client App

note:: this getById(-) or getReferenceById(-) or

//Doctor doctor-doctorRepo.getById(id); Doctor doctor=doctorRepo.getReferenceById(id); return doctor;

System.out.println(service.findDoctorById(5674));

getOne(-) methods of JpaRepository performs the lazy loading of the object by default i.e when this method is called it just returns Proxy object (subclass of the entity class obj) having id value (i.e no hit to db s/w). when non-indentifier method is called on the top of proxy object the real hit to db s/w takes place and the gathered record will be stored in to Entity class object (real object) that is linked with Proxy object

In application.properties

another

#To enable lazy loading of record in the underlying Hibernate f/w
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true

and getReferenceById() need

For getOne(-),getById(-) methods this property to make underlying Hibernate f/w to support lazy loading even though Transactional env.. is not taken.. In old versions spring data jpa just @Transacationl in service is class sufficient.

=> The proxy object returned by the getReferenceById(-) method is the object InMemory proxy class that extends from Entity class. if the Entity class is taken as the final class then this InMemory proxy class generation fails. This makes getReferenceById(-) performing earger loading of the object

Proxy obj (sub class obj

of Entity class)

(d) doc.getSpecialization()

(non-indenfier method)

(h) cardio

select from jap_doctor tab where docid=1001 (e)

flow diagram for findById(-) method

(eager Loading)

====

In service Impl class

(a) Optional<Customer> opt=custRepo.findById(cno);

1001

What is the difference et

getById(-) of JpaRepository and findById(-) of CrudRepository? or getReferenceById(-)

getById(-) getOne()/getReferenceById()

(a) retuns Entity obj ref representing

record that is selected for given id

(first gives proxy object then gives real entity object)

(b) performs Lazy Loading of record/object

i.e first returns proxy object and when that

proxy is used then record will be retrieved

from DB s/w to put into Entity class obj (real obj)

is

(c) if record not found we can not throw

custom exception.. it gives the fixed EntityNotFoundException

(d) Works only in SQL DB s/w

(e) Impl depends on on undelrying Hibernate framework

findById(-) (CrudRepository)

(a) returns Optional<T> obj having Entity obj for the same (Directly gives Optional object having the Entity obj)

Optional object

aving Customer obj

(b) perform eager loading i.e gets the record/object from DB table directly with out involving any proxy object irrespective of wheather that object is used or not

(c) with the support of Optional API we can throw Custom Exception.. or custom message

(d) Works in both SQL and NO SQL DB s/w

(e) Impl is given in spring dataJpa itself

(f) addtional property cfg in in applicaiton.properties (f) not required.

is reuqired

(g) if record not found we get exception (EntityNotFoundException)

(h) This method getReferenceById(-) does not actually talks with Db s/w.. The non-indentifier getter methods generated proxy class talks with DB s/w by generating SQL Query

(i) if Entity class is taken as final class then InMemory Proxy class will not be generated

So,

this method performs only eager loading

(j) In getReference ById(-) method call 1 proxy

obj and 1 real obj is involved generally

(g) if record not found, we get Empty Optional object

for which we send custom message or we can throw exception

(h) This method directly talks with DB s/w by generating the SQL Query

(i) This method always performs eager loading irrespective of wheather entity class is taken as final class or not?

(j) In findById(-) method, only one real object is involved

While working with spring Data JPA choose persistence methods in the following order

(a) CrurdRepository methods

==>if not sufficient then

(b) PagingAndSortingRepository methods

Common Reposorities

==>if not sufficient then

(c) JpaRepository methods

==> if not sufficient then

(d) Custom methods in our Repository Interface.

Best

custom

custom

finder methods (only for select opertions) [Also called as findBy methods]

(ii) @Query methods (for HQL/JPQL and Native SQL Queries

custom

based Select operations)

(iii) @Query + @Modifying methods

(for HQL/JPQL and Native SQL Queries based

custom non-Select operations)

(insert,update, delete operations)

various options to place

custom methods in the our Repository interfaces

HQL:: Hibernate Query Language

JPQL:: Jakarata Persistence API Query Language

What is the difference b/w Example object and Optional Object?

hold

Ans) Example object is useful to another Entity object with example data/values using which the findAll(Example) method performs search operation to perform select operation and get records by given Entity data with and clause conditions

Optional object useful to hold another Entity object. This object holds Entity given by findById(-) method representing the record that is given by select SQL Query execution

Select from JPA_CUSTOMER Where cid=1001

Customer obj(Entity obj)

rs(ResultSet)

BFR

cno:101 cname:raja cadd:hyd billamt:9000

(d)

(c)

1001 raja hyd 90000

LAIR

DB s/w

1001

...

....

...

...

....