saveAll(-) method in CrudRepository

==================================

=>This method is given for bulk records insertion /Bulk objs insertion =>Geneates multiple insert SQL Queries to to insert the multiple records

saveAll

note:: Iterable<T> is super interface for all collections

<S extends T> Iterable <S> saveAll(Iterable <S> entities) Saves all given entities.

Parameters:

entities - must not be null nor must it contain null.

Returns:

=>This method is useful in bulk insertion activites like

group ticket reservation, group rooms booking in hotel and etc..

the saved entities; will never be null. The returned Iterable will have the same size as the Iterable passed as an argument.

Throws:

IllegalArgumentException - in case the given entities or one of its entities is null.

In service Interface

public String registerArtistBatch(List<Artist> list);

In service Impl class

@Override

public String registerArtist Batch(List<Artist> list) {

}

//save the objs

Iterable<Artist> savedList=artistRepo.saveAll(list);

List<Integer> ids-StreamSupport.stream(saved List.spliterator(),false).map(Artist::getAid).collect(Collectors.toList()); return ids.size()+" no.of artits are registered having the idValues "+ids;

@Override

(or)

hote:: Collection(), on based collections like List, Set, Map we can apply stream directly.. But Iterable<T> does not allow stream() directly So we need to use StreamSupport.stream(Iterable<T> obj) for the same

public String registerArtistBatch(List<Artist> list) { //save the objs

List<Artist> savedList={List<Artist>)artistRepo.saveAll(list);

List<Integer> ids=saved List.stream().map(art->art.getAid()).collect(Collectors.toList());

return ids.size()+" no.of Artists are saved with the id values ::"+ids;

}

(or)

@Override

Select operations using CrudRepository of spring data JPA Optional<T> findById(ID id); --> to get single record

Iterable<T> findAll(); ->to get all records

Iterable<T> findAllById(Iterable<ID> ids); -->to get multiple records based on given ids

boolean existsById(ID id); -> To check record is avaiable or not long count(); --> to get count of records.

How to disable block commenting in java coding of Eclipse IDE?

Ans) window menu --->preferences ---> search for formatter --->

X

Formatter

Optional API is java 8 feature, Optional object

holds one another object or remain empty and provides various methods to check wheather another object came or not

=>java.util.Iterable<E> is the top most interface in the inheritance hierarchy of collection apis.. Using this we can refer any List collection or Set collection object directly or indirectly

formatter

✓ Ant

✓ Editor

Formatter

✓ Java

Active profile: Eclipse [built-in]

✓ Code Style

New...

name of formatter :: f1

Formatter

✓ Debug

Detail Formatters

java 9 imp features

Preview:

**/

▸ Indentation

▸ **Brace positions**

▸ Parentheses positions

▸ Whitespace

▸ **Blank Lines**

▸ **New Lines**

► **Line Wrapping**

▾ **Comments**

Maximum width for comments:

Count width from comment's starting position

Enable Javadoc comment formatting

Enable block comment formatting

Enable line comment formatting

(De select this checkbox)

=>List.of(-,-,-)

=>Set.of(-,-,-) =>Map.of(-,-,-)

[Gives the Collections as the immutable Collections]

important java 8 features

============================

=>java 8 interface default methods

=> java 8 interface static methods

=> functional interface

=> lamdas

=> streaming api

=> Optional api

=> static method reference ::

=> constructor reference

=> non-static method reference =>forEach() method

and etc..

=> Date and time api (JODA api)

count

`long count()`

Returns the number of entities available.

Returns:

the number of entities.

example

In Service Interface

public long fetchDoctorsCount();

In service Impl class

@Override

public long fetch DoctorsCount() {

return doctorRepo.count();

=>Spring data jpa api, hibernate api exceptions are unchecked exceptions

=> Jdbc Exceptions are checked exceptions.. the hibernate

apis internally translates the jdbc checked exceptions into unchecked exceptions uusing Exception propagation concept

=> Spring data jpa api internally translates hibernate api exceptions to spring data exceptions using exception propagation concept

}

In Client App

try {

System.out.println("count of records ::"+service.fetchDoctorsCount());

}

catch(Exception e) {

e.printStackTrace();

}

DB s/w

JPA_DOCTOR_NFO

internal process

================

SELECT COUNT(*) FROM JPA_DOCTOR_INFO

Every RS(ResultSet) contains

two positions by default

a) BFR :: Before First Record

b) ALR :: After Last Record

existsById

`boolean existsById(ID id)`

Returns whether an entity with the given id exists.

Parameters:

id must not be null.

Returns:

true if an entity with the given id exists, false otherwise.

Throws:

`IllegalArgumentException -if id is null.`

BFR

rs(ResultSet)

10

(1)

ALR

rs getInt(1)

example

Code in service Interface

public boolean checkDoctorAvailbility(Integer id);

Code in service Impl class

@Override

public boolean check DoctorAvailbility(Integer id) {

return doctorRepo.existsById(id);

}

code in Client App

try {

}

10

System.out.println("201 Id doctor exists ?::"+service.checkDoctorAvailbility(201));

catch(Exception e) {

}

e.printStackTrace();

DB s/w

JPA_DOCTOR_NFO

SELECT COUNT(*) FROM JPA_DOCTOR_INFO WHERE DOC_ID=?

↓

201

BER

RS

1/0

(1)

findAll

`Iterable <T> findAll()`

Returns all instances of the type. (gives all the records of table in form of

Returns:

all entities

Iterable/List of entity class objs)

of

=>java.lang.Iterable is top most interface in the inheritence hierachty collections api

example

Code in service Interface

public Iterable<Doctor> showAllDoctors();

Code in service Impl class

@Override

public Iterable<Doctor> showAllDoctors() {

return doctorRepo.findAll();

}

Code in Client App

try {

Iterable<Doctor> it-service.showAllDoctors(); it.forEach(doc->{

int count=rs.getInt(1) if(count==0)

return false;

else

return true:

public String registerArtist Batch(List<Artist> list) { //save the objs

Iterable<Artist> saved List=artistRepo.saveAll(list); List<Integer> ids List=new ArrayList();

savedList.forEach(artist->{

});

}

idsList.add(artist.getAid());

return idsList.size()+" no.of artits are registered having the idValues "+idsList.toString(); Client app (or) runner

Artist artist1=new Artist("mahesh1","HERO", 4000000.0);

Artist artist2=new Artist("SRK1","HERO", 5000000.0); Artist artist3=new Artist("Prabhas1","HERO", 3000000.0);

List<Artist> list=List.of(artist1, artist2,artist3); try {

String msg=artistService.registerArtistBatch(list);

System.out.println(msg);

}

catch(Exception e) {

e.printStackTrace();

}

=> From java 8 onwords, we can use forEach(-) method to read and display the element values.. if want to perform aggregate operations or sub grouping operations or conversion operations then prefer using streaming api

try {

Iterable<Doctor> list=docService.findAllDoctors(); list.forEach(doc->{

Java 8 forEach(-) method

System.out.println(doc);

});

System.out.println("_

_");

System.out.println(".

_");

it.forEach(doc->System.out.println(doc)); | Improved forEach(-) method of java8

it.forEach(System.out::println); | java 8 forEach(-) + static method reference

System.out.println(".

_");

```
for(Doctor doc:it) {
System.out.println(doc);
Java 5 enhanced for loop
}
}
((List<Doctor>)it).stream().forEach(System.out::println);
catch(Exception e) {
(Working java 8 forEach() method and streaming api)
e.printStackTrace();
}
DB s/w
JPA_DOCTOR_NFO
SELECT
*
FROM JPA_DOCTOR_INFO
BFR
rs(ResultSet)
System.out.println(doc);
});
System.out.println("-
-");
list.forEach(doc->System.out.println(doc));
System.out.println("-
-");
list.forEach(System.out::println);
System.out.println("-
-");
StreamSupport.stream(list.spliterator(), false).forEach(System.out::println);
System.out.println("-
-");
long count=StreamSupport.stream(list.spliterator(), false).count();
System.out.println("No.of records ::" +count);
System.out.println("
-");
StreamSupport.stream(list.spliterator(),
Iterable<String> specialatiesList=
false).map(Doctor::getSpecialization).collect(Collectors.toList());
```

```
System.out.println(specialatiesList);
System.out.println("-
-");
for(Doctor doc:list) {
System.out.println(doc);
}
System.out.println("-
-");
List<Doctor> list1=StreamSupport.stream(list.spliterator(), false).toList(); for(int i=0;i<count; ++i) {
System.out.println(list1.get(i));
Doctor obj
}
List collection(Iterable obj)
with the objs of Entity class
}
catch(Exception e) {
e.printStackTrace();
Doctor obj
}
BFR: Before First Record
ALR
ALR: After Last Record
Doctor obj
Internal jdbc code
PreparedStatement ps=
con.prepareStatement("SELECT * FROM JPA_DOCTOR_INFO");
ResultSet rs=ps.executeQuery();
Iterable<Doctor> it=new ArrayList();
while(rs.next()){
Doctor doc=new Doctor();
doc.setDocId(rs.getInt(1));
Copying each record of
}
doc.setDocName(rs.getString(2));
doc.setIncomde(rs.getFloat(4));
it.add(doc);
RS object to Entity class obj
```

Adding each Entity obj to List/Iterable Collection

Doctor obj

Why are we not using @Reposotory annotation in spring data jpa/ spring boot data jpa applications? Ans) In spring /spring boot data jpa applicaton we just develop custom Repsitory Interface having inherited methods and optional custom methods.. For these methods the implementation logics will be given by spring/spring boot data jpa in the dynamically generated InMemory Proxy class that is implementing custom Repository interface and that class internally becomes spring bean.. Since we are not developing DAOImpl class manually.. So there is no need of placing @Repository annotation

The In Memory Proxy class that is generated implementing our CustomRepository interface will get @Repository annotation automatically

findAllById

`Iterable <T> findAllById(Iterable <ID> ids)`

Returns all instances of the type T with the given IDs.

If some or all ids are not found, no entities are returned for these IDs.

Note that the order of elements in the result is not guaranteed.

Parameters:

ids must not be null n

Returns:

ids collection

(elements can be null)

guaranteed to be not null. The size can be equal or less than the number of given ids.

Throws:

`IllegalArgumentException - in case the given ids is null.`

example

In service Interface

public Iterable<Doctor> showAllDoctorsByIds(Iterable<Integer> ids);

In service Impl class

@Override

public Iterable<Doctor> showAllDoctorsByIds(Iterable<Integer> ids) {

// TODO Auto-generated method stub

return doctorRepo.findAllById(ids);

=>List<Integer> list=new ArrayList(); gives mutable list Collection i.e we can add, remove, modify the element values .Arrays.asList(.,.,.) also gives mutable Collection

=> Set.of(-,-,-), List.of(-,-,-),Map.of(-,-) and etc.. methods are introduced from java 9... and these method return immutable collection objs i.e once elements are added to the collection, they can not be modified. more over these collections do not allow "null" values as the elements..

List<Integer> list1=List.of(10,20,30); list1.add(40); //gives error

List<Integer> list2=List.of(10,20,30,null); // throws NullPointerException List<Integer> list3=Arrays.asList(10,20,40,null); //success

}

In Client app

try {

service.showAllDoctorsByIds(List.of(1,2,100,200,101)).forEach(System.out::println);

}

(or)

catch(Exception e) {

e.printStackTrace();

service.showAllDoctorsByIds(null); //throws leagalArgumentException

(or)

}

service.showAllDoctorsByIds(List.of(101,null,234); //throws NullPointerException (or)
service.showAllDoctorsByIds (Arrays.asList(1,2,101,null); ///valid

Q) What is optional API in Java8? what is the benifit of it?

Ans) It is java8 feature which gives Optional object containing another object.. optional api that can be invoked

on the optional object is very useful to check weather other object is stored or not inside the Optional object.. note:: An optional object can store only one another at maximum and at a time

Optional<Student> opt=Optional.of(new Student());~

Optional<Student> opt1=Optional.empty();

Optional obj(opt)

Student

obj

static method

Optional obj(opt1)

Obj

if(opt.isPresent()){ sysout("Optional is having another obj");

O

Student st=opt.get();

}

if(opt1.empty()){

else{

Sysout("op1 obj is an empty obj");

else

Sysout("Optional obj is not having another obj");

{

Sysout("opt1 obj is having another obj");

}

}

// get. the object from the Optional obj

Student st=opt.get();

Optional API, in java 8 is given to avoid NullPointerException in the execution of the

java App, especially while dealing with the methods whose return type class name

This Optonal is very useful to avoid NullPointerException in the execution of the Application

Method Summary

All Methods

Modifier and Type

Static Methods Instance Methods Concrete Methods

static <T> Optional<T>

`boolean`

`Optional<T>`

`<U> Optional<U>`

`T`

`int`

`void`

Method and Description

`empty()`

Returns an empty Optional instance.

`equals(Object obj)`

Indicates whether some other object is "equal to" this Optional.

`filter(Predicate<? super T> predicate)`

If a value is present, and the value matches the given predicate, return an Optional describing the value, otherwise return an empty Optional. flatMap(Function<? super T,Optional<U>> mapper)

If a value is present, apply the provided Optional-bearing mapping function to it, return that result, otherwise return an empty Optional.

get()

If a value is present in this Optional, returns the value, otherwise throws NoSuchElementException.

hashCode()

Returns the hash code value of the present value, if any, or 0 (zero) if no value is present.

`ifPresent (Consumer<? super T> consumer)`

If a value is present, invoke the specified consumer with the value, otherwise do nothing.

`boolean`

`<U> Optional<U>`

`isPresent()`

`Return true if there is a value present, otherwise false.`

`map (Function<? super T,? extends U> mapper)` `static <T> Optional<T>`

of (T value)

`static <T> Optional<T>`

`ofNullable(T`

`value)`

If a value is present, apply the provided mapping function to it, and if the result is non-null, return an Optional describing the result.

Returns an Optional with the specified present non-null value.

Returns an Optional describing the specified value, if non-mull, otherwise returns an empty Optional.

`T`

`T`

`<X extends Throwable>`

`T`

`String`

Methods inherited from class java.lang.Object

`clone, finalize, getClass, notify, notifyAll, wait, wait, wait`

`orElse(T other)`

Return the value if present, otherwise return other.

`orElseGet (Supplier<? extends T> other)`

`Return the value if present, otherwise invoke other and return the result of that invocation.`

`orElseThrow(Supplier<? extends X> exceptionSupplier)`

`Return the contained value, if present, otherwise throw an exception to be created by the provided supplier.`

toString()

`Returns a non-empty string representation of this Optional suitable for debugging.`

yess ir

To: Ev

Enter