Delete Object Operations in CrudRepository
==

void	delete(T entity)	Deletes a given entity.
void	deleteAll()	Deletes all entities managed by the repository.
void	deleteAll(Iterable<? extends T> entities)	Deletes the given entities.
void	deleteAllById(Iterable<? extends ID> ids)	Deletes all instances of the type T with the given IDs.
void	deleteById(ID id)	Deletes the entity with the given id.

---

void	deleteAll()	Deletes all entities managed by the repository.

=> Deletes all entities/records of the db table by generating single select SQL query and multiple DELETE SQL Queries

=> First generates the single SELECT SQL Query and gets all the ID values (PK column values) and generates the multiple DELETE SQL Queries having submitting each ID value to each Delete SQL Query

=> It does not generate single DELETE SQL Query for deleting all the records .. In fact it generates multiple DELETE SQL Queries to delete multiple records.

> void deleteAll()
>
> Deletes all entities managed by the repository.

Example App

In service Interface

```
public  String  removeAllDoctors();
```

In service Impl class

```
@Override
	public String removeAllDoctors() {
		long count=doctorRepo.count();
		if(count>0) {
			doctorRepo.deleteAll();
			return count+" no.of records are deleted";
		}
		return "no records found delete";
	}
```

In Client App

```
		try {
			System.out.println(service.removeAllDoctors());
		}
		catch(Exception e) {
			e.printStackTrace();
		}
```

```
[illegible]
Hibernate: select count(*) from jpa_doctor_info di1_0
Hibernate: select di1_0.doc_id,di1_0.doc_name,di1_0.income,di1_0.specialization from jpa_doctor_info di1_0
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
Hibernate: delete from jpa_doctor_info where doc_id=?
14 no.of records are deleted
```

To get all the records and their ids

Here ids are submitted to delete queries to delete the records.

all the methods of JPA repositories execute in DB s/w by enabling auto commit mode on DB s/w... As developer we can not rollback the data that has been committed.
(If developers wants to get this control then he needs to enable Transactions/Tx Operations)

---

void	deleteAllById(Iterable<? extends ID> ids)	Deletes all instances of the type T with the given IDs.

=> Deletes the multiple records/entities of db table based on the given id values

> void deleteAllById(Iterable<? extends ID> ids)
>
> Deletes all instances of the type T with the given IDs.
>
> Entities that aren't found in the persistence store are silently ignored.
>
> Parameters:
>
> ids - must not be null. Must not contain null elements. (elements can be null)
>
> Throws:
>
> IllegalArgumentException - in case the given ids or one of its elements is null.
>
> Since:
>
> 2.5

Example app
===========

In service Interface

```
public String removeDoctorsByIds(Iterable<Integer> ids);
```

In service Impl class

```
@Override
	public String removeArtistsByIds(Iterable<Integer> ids) {
		//Load the objs by ids
		Iterable<Artist> list=artistRepo.findAllById(ids);
		long count= StreamSupport.stream(list.spliterator(), false).count();

		//delete objs by ids
		artistRepo.deleteAllById(ids);
		return count+" no.of Artists are deleted ";
	}
```

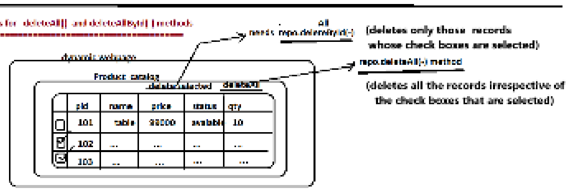
In client App

```
	try {
		System.out.println("artists count:"+artistService.removeArtistsByIds(Arrays.asList(50,88,99)));
	}
	catch(Exception e) {
		e.printStackTrace();
	}
```

```
Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid in (?,?,?)
Artist: 0-arg constructor
Artist: 0-arg constructor
Artist: 0-arg constructor
Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid=?
Artist: 0-arg constructor
Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid=?
Artist: 0-arg constructor
Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid=?
Artist: 0-arg constructor
Hibernate: delete from artist_info where aid=?
Hibernate: delete from artist_info where aid=?
Hibernate: delete from artist_info where aid=?
artists count:3 no.of Artists are deleted
```

---

usecases for deleteAll() and deleteAllById() methods
===



=> In gmail apps, in Photo Album apps we definetly need this deleteAll() , deleteAllById() methods

---

Delete Object Operations in CrudRepository

==========

`void`

`void`

`delete(T entity)`

`void`

`deleteAll()`

`void`

`void`

`void`

`deleteById(ID id)`

Deletes a given entity.

Deletes all entities managed by the repository.

deleteAll(Iterable<? extends T> entities) Deletes the given entities.

`deleteAllById(Iterable <? extends ID> ids) Deletes all instances of the type T with the given IDs.`

`deleteAll()`

Deletes the entity with the given id.

Deletes all entities managed by the repository.

=> Deletes all entities/records of the db table by generating single select SQL query and multiple DELETE SQL Queries

=> First generates the single SELECT SQL Query and gets all the ID values (PK column values) and generates the multiple DELETE SQL Queries having submiting each ID value to each Delete SQL Query

=> It does not generate single DELELTE SQL Query for deleting all the records .. In fact it generates multiple DELETE SQL Queries to delete multiple records.

`void deleteAll()`

Deletes all entities managed by the repository.

Example App

In service Interface

public String removeAllDoctors();

In service Impl class

```
@Override
public String removeAllDoctors() {
long count=doctorRepo.count();
if(count>0) {
doctorRepo.deleteAll();
return count+" no.of records are deleted";
```

}

return "no records found delete";

}

In Client App

try {

System.out.println(service.removeAllDoctors());

}

catch(Exception e) {

}

e.printStackTrace();

<terminated> BootDataJpaProj1Crud RepositoryApplication [Java Application] E:\Software\Eclipse\eclipse-jee-2022-03-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.I Hibernate: select count(*) from jpa_doctor_info d1_0

Hibernate: select d1_0.doc_id,d1_0.doc_name,d1_0.income,d1_0.specialization from jpa_doctor_info d1_0 Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=?

Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? Hibernate: delete from jpa_doctor_info where doc_id=? 14 no.of records are deleted

To get all the records and their ids

Here ids are submitted to delete queries to delete the records.

all the methods of JPA repositories execute

in DB s/w by enabling auto commit mode

on Db s/w... As developer can not rollback

the data that has been committed.

(if developers wants to get this control then he needs to enable TransactionMgmt Operations)

```
void
deleteAllById(Iterable <? extends ID> ids) Deletes all instances of the type T with the given IDs.
```

=> Deletes the multiple records/entities of DB table based on the given id values

```
void deleteAllById(Iterable <? extends ID> ids)
```

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

Entities that aren't found in the persistence store are silently ignored.

Parameters:

ids - must not be null. Must not contain null elements. (elements can be null)

Throws:

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

Since:

2.5

Example app

==========

In service Interface

public String removeDoctorsByIds(Iterable<Integer> ids);

In service Impl class

@Override

public String removeArtistsByIds(Iterable<Integer> ids) {

}

In client App

//Load the objs by ids

Iterable<Artist> list=artistRepo.findAllById(ids);

long count= StreamSupport.stream(list.spliterator(), false).count();

//delete objs by ids

artistRepo.deleteAllById(ids);

return count+" no.of Artits are deleted ";

try {

}

System.out.println("artits count::" +artistService.removeArtistsByIds (Arrays.asList(50,88,99)));

catch (Exception e) {

}

e.printStackTrace();

Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid in (?,?,?)

Artist:: 0-arg cosntructor

Artist:: 0-arg cosntructor

Artist:: 0-arg cosntructor

Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0.aid=? Artist:: 0-arg cosntructor

Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0_aid=? Artist:: 0-arg cosntructor

Hibernate: select a1_0.aid,a1_0.aname,a1_0.category,a1_0.fee from artist_info a1_0 where a1_0_aid=? Artist:: 0-arg cosntructor

Hibernate: delete from artist_info where aid=?

Hibernate: delete from artist_info where aid=?

Hibernate: delete from artist_info where aid=? artits count::3 no.of Artits are deleted

usecases for deleteAll() and deleteAllById(-) methods

=====

=====

dynamic webpage

Chat

from Abhishek (privately): 12:

3

All

needs repo.deleteById(-) (deletes only those records

whose check boxes are selected) repo.deleteAll(-) method

(deletes all the records irrespective of

the check boxes that are selected)

Product catalog

delete selected deleteAll

pid

name price status qty

101

table 99000 avaiable 10

102

103

=> In gmail Apps, in Photo Album Apps we definetly need this deleteAll(), deleteAllById(-) methods.

Assigment :: pratice this method

`void`

deleteAll(Iterable <? extends T> entities) Deletes the given entities.

(Though we are passing List of Entity objects to delete ..it actually internally use

the id values of given Entity objs for deleting the records)

deleteById(Integer id)

`void deleteById(ID id)`

Deletes the entity with the given id.

If the entity is not found in the persistence store it is silently ignored.

Parameters:

id - must not be null.

Throws:

IllegalArgumentException - in case the given id is null

In service Interface

delete

note:: All methods in Crud Repository is

having void as the return type So we need to write additional Code to check weather records are found or not

and Records are deleted or not

public String remove PatientById(int id);

In service Impl class

```
@Override
public String remove PatientById(int id) {
Optional Patient> opt-patientRepo.findById(id);
if(opt.isPresent()) {
patientRepo.deleteById(id);
return id+" patient found and deleted";
}
else {
return id+"patient bund for deletion";
}
}
```

In runner class

```
===============
System.out.println(patientService.remove PatientById(1007));
```

Assignments

`void`

`delete(T entity)`

Deletes a given entity.

`void`

deleteAll(Iterable <? extends T> entities) Deletes the given entities.

Can we delete the record content partially?

using

Ans) possible, but not delete(-) methods, use save(-) keeping null values for few

properties of the object... this is nothing but partial deletion of the record (nothing but partial update/delete operation) or we can take @Transient for few properties

What is meaning of Common Repositories and Specific Repositories?

Ans)

=>CrudRepository and PagingAndSortingRepository are called Common Repositories becoz they are common for both SQL and NOSQL DB s/ws.

=> JpaRepository methods are specific SQL DB s/ws So it is called as specific Repository

=> MongoRepository methods are specific Mongo DB s/ws (NoSQL),So it is called as specific Repository

=> Neo4jRepository methods are specific Neo4J DB s/ws (NoSQL),So it is called as specific Repository