Date : 07/04/2021
Spring Boot 9AM
Mr. RAGHU

---

Custom Query methods in MongoDB

@Query : This annotation is used to provide custom query
for mongoDB using abstract methods defined in
Repository interface.

Ex#1 When we are fetching one JSON doc data
it may return null also, then use a wrapper Optional
to avoid null pointer exception.

```
--Bad Practice(but works)----
@Query("{ id : ?0 }")
Book getBookById(Integer id);

---Good Practice------
@Query("{ id : ?0 }")
Optional<Book> getBookById(Integer id);
```

*) Note: Here { } inside @Query indicates where condition
and ?<index> indicates data comes at runtime.

*) Operators and symbols

| | |
|---|---|
| < | $lt |
| <= | $lte |
| > | $gt |
| >= | $gte |

*) Stream(java.util) is added in JDK 1.8 , used to execute
set of operations without loops in a better way with
less code like filter, sorting, maping, collecting,
printing,..etc

Ex#3

```
@Query("{ noOfPages : { $gte : ?0 }  }")
//List<Book> getBooksByNoOfPages(Integer noOfPages);
Stream<Book> getBooksByNoOfPages(Integer noOfPages);

repo.getBooksByNoOfPages(150)
 .filter(ob->ob.getCategory().equals("BackEnd"))
 .sorted((ob1,ob2)-> ob1.getId().compareTo(ob2.getId()))
 .map(ob->ob.getTitle()+" is written by :" +ob.getWriter())
 .forEach(System.out::println);
```

==========

writing and/or

```
$or : [ { condition#1 }, { condition#2 }, { condition#3 }, ... ]

$and : [ { condition#1 }, { condition#2 }, { condition#3 }, ... ]

SQL: where sid =? or sname = ?
$or : [ { sid:?0 } , { sname : ?1} ]
```

```
SQL: where sfee=? and (sid =? or sname = ?)
$and : [ { sfee=?0 },  $or : [ { sid:?1 } , { sname : ?2} ] ]

SQL: where sid>? and sfee<=?
$and : [ { sid : { $gt: ?0 } }, { sfee:  { $lte: ?1 } } ]


-> between using > and < symbols
db.book.find( { cost : { $gt: 15, $lt: 20 } } )

SQL: select * from book where id>? and (writer=? or category=?)
{
  $and : [
            { id : { $gt : ?0 } },

            {
                $or : [
                          { writer : ?1} ,

                          {category : ?2}
                      ]
            }
         ]
}
  ====================code============================
1. Model
package in.nareshit.raghu.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Book {
        @Id
        private Integer id;
        private String title;
        private Integer noOfPages;
        private String writer;
        private String category;

}

2. Repository
package in.nareshit.raghu.repo;

import java.util.List;
import java.util.Optional;
import java.util.stream.Stream;
```

```
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import in.nareshit.raghu.model.Book;

public interface BookRepository
        extends MongoRepository<Book, Integer> {

        //custom query methods
        //SQL: select * from book where id=?
        @Query("{ id : ?0 }")
        //Book getBookById(Integer id);
        Optional<Book> getBookById(Integer id);

        //SQL: select * from book where writer=? and category=?
        //@Query("{ writer : ?0 , category : ?1}")
        @Query("{ $and : [ { writer : ?0} , {category : ?1} ] }")
        List<Book> getBooksByWriterAndCategory(String writer, String
category);

        //-----------------------------------------
        //SQL: select * from book where noOfPages>=?
        //@Query("{ noOfPages : ?0 }")  //noOfPages=?
        @Query("{ noOfPages : { $gte : ?0 }  }")
        //List<Book> getBooksByNoOfPages(Integer noOfPages);
        Stream<Book> getBooksByNoOfPages(Integer noOfPages);

        @Query("{ writer : ?0 , noOfPages : { $lt : ?1 } }")
        List<Book> getBooksByWriterAndNoPages(String writer,Integer
noOfPages);

        //-------------------------------------------------
        //SQL: select * from book where writer=? or category=?
        @Query("{ $or : [ { writer : ?0} , {category : ?1} ] }")
        List<Book> getBooksByWriterOrCategory(String writer, String
category);

        //SQL: select * from book where id>? and (writer=? or
category=?)
        @Query("{ $and : [ { id : { $gt : ?0 } }, { $or : [ { writer :
?1} , {category : ?2} ] } ] }")
        List<Book> getBooksByDataA(Integer id,String writer, String
category);


}
```

3. Insert Runner

```
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;

import in.nareshit.raghu.model.Book;
import in.nareshit.raghu.repo.BookRepository;

//@Component
```

```
public class DataInsertRunner implements CommandLineRunner {

	@Autowired
	private BookRepository repo;

	public void run(String... args) throws Exception {
		repo.deleteAll();

		repo.save(new Book(10, "Core Java", 250, "SAM",
"BackEnd"));
		repo.save(new Book(11, "Adv Java", 260, "SYED",
"BackEnd"));
		repo.save(new Book(12, "Angular", 350, "SAM",
"FrontEnd"));
		repo.save(new Book(13, "HTML", 120, "SYED",
"FrontEnd"));
		repo.save(new Book(14, "Spring Boot", 850, "SYED",
"BackEnd"));
		repo.save(new Book(15, "Microservices", 350, "SAM",
"BackEnd"));
		repo.save(new Book(16, "ReactJS", 180, "RAM",
"FrontEnd"));


	}

}
```

4. Test Runner

```
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.repo.BookRepository;

@Component
public class TestQueryRunner implements CommandLineRunner {

	@Autowired
	private BookRepository repo;

	public void run(String... args) throws Exception {
		/*Optional<Book> opt = repo.getBookById(1505);
		if(opt.isPresent()) {
			System.out.println(opt.get());
		} else {
			System.out.println("No Data Found");
		}*/
		//----------

		//repo.getBooksByWriterAndCategory("SAM", "BackEnd")
		/*
		repo.getBooksByNoOfPages(150)
		.filter(ob->ob.getCategory().equals("BackEnd"))
		.sorted((ob1,ob2)->
```

```
ob1.getId().compareTo(ob2.getId()))
                .map(ob->ob.getTitle()+" is written by :"
+ob.getWriter())
                .forEach(System.out::println);
                */

                //repo.getBooksByWriterAndNoPages("SAM", 900)
                //repo.getBooksByWriterAndCategory("SAM", "BackEnd")
                //repo.getBooksByWriterOrCategory("SAM", "FrontEnd")
                repo.getBooksByDataA(5, "SAM", "FrontEnd")
                .forEach(System.out::println);

        }

}
```

5. application.properties

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nit
spring.data.mongodb.username=NIT
spring.data.mongodb.password=RAGHU
```