cu

Dat

li

(

F

Different GUI Tools to work with MongoDB

a) Compass (Bit heavy s/w)

b) Robo3T (Light weight) (Renamed to Studio 3T)

Working with Robo3T (studio 3T)

=>donwload s/w from :: https://robomongo.org/download

=>install s/w like any other windows/w

Procedure to create Logical DB with

==========

collection in MongoDB using Robo3T

==========

step1) launch robo3t and connect to MongoDB by creating new connection studio3t

step2) create Logica DB

Right click connection ---> create Data base ---> name:: NTSPBMS715DB step3) create Collection and insert document

expand logical DB

NTSPBMS715DB ---> right Click on Collection ---> new collection :: Employee

double click on cusotmer collection

insert

view

delete

edit

Result Query Code Explain

→

50

Documents to

(2 added)

[mployee > ename

_id

eno

ename

eadd

salary

id 630d6d95fff45b5f73392a74

123 101.0

"_" raja

Lid 630d6dccfff15b5f73392a75

123 156.0

"_" rajesh

"" delhi "_"hyd

123 9000.0

step4) To update the document

option1: View the document ---> edit document and modify attribute vlaues (or)

right click on customer (document) --> update document --> with new values

step4) perform other operations on the documents and collection ..

=>remove document

=>rename cllection

=> drop collection

note:: Adding the username and password is optional, we can connect this Logical Db

Add username, password to Logical DB for Authentication using Robot3T /Studio 3T

=>

select

LogicaDB (ntsp715DB)

username :: testuser

==>

--->users --> add user --->

(in tool bar) (or) right click LogicalDB

password :: testuser

manage users --> adduser-> ....

select

readwrite grant

Disconnect from MongoDB ---> connect ----> delete the existing connection ---> create new connection name: con1---> Authentication tab ---> select legacy ---> type username: testuser, password: testuser --> Logical DB name: NTSPBMS715DB ---> TestConnection ---> save ---> connect

Edit User testuser

con1 (localhost:27017) > NTSPBMS715DB

Name: testuser

Password: .....

The password is set, but not shown. Use the field above to

Roles Custom Data

Role

readWrite

Database

NTSPBMS715DB

with out username andn password

Connection name: con1

Connection group: <root level>

Server Authentication SSL SSH Proxy IntelliShell Mong

Authentication Mode: Legacy (SCRAM-SHA-1)

You have chosen an authentication method which might make your Mong Please refer to the MongoDB Security Checklist to help secure your databa

User name:

Password:

testuser

.....

Authentication DB:

NTSPBMS715DB

The database where the user is defined

Developing Document class whose objects represent documents of a collection in MongoDB

Employee |--> id

|--->eno Doc1

|-->ename

|-->eadd

Employee

Here the attributes

are not fixed,So

|--> id

|--->eno

we must take @Document class

if want to represent the documents of a collection using

the objects of java class.. then that class must be

designed having highest possible properties representing the attributes of the Document.

The java class whose objects represent documents of Collection is called Document class (@Document class)

|-->ename

|-->eadd

|-->salary

Doc2

having max attrs

version1::

(Good to use)

==========

=>Like Hibernate /JPA we do not

have readymade generators cfg

for Id property in MongoDB document class.

In MongoDB officially there is no

PK, FK constraits.. but we can

bring that effect indirectly..

PK constriant using _id attribute

FK constraint using documents nesting

or chanining.

@Document

@Data

public class Employee{ @Id

private String id; private String ename; private String eadd; pivate Double salary;

}

// if u want to use MongoDB generated id value

as the id value of Document object then take

String property having @Id as the id property as shown here. // if u r not intrested to use MongoDB generated id value as the

id value of the Document then we can take support one or another generator support to generate the id value

(like UUIDGenerator class)

Version2: (Not recomanded to use)

==========

@Document

@Data

public class Employee{ @Id

private Integer eno:

private String ename; private String eadd; private Double salary;

(Third party supplied generator)

// if u feel that u can store unique value in "cno" property

then make "cno" property as the id property.. otherwise follow "version1".

}

conclusion::

The java Document object representing the MongoDB document of a collection can have

dynamically generated string value as the id value

or custom String value as the id value (given by IdGenerator)

or our choice property value as the id value (version?)

(version1) (Best)

Procedure to develop First Springdata MongoDB Application using MongoRepository (1)

step1) create Spring Boot starter project

selecting spring data MongoDB, Lombok starters..

step2) add the following properties in application.properties file.

application.properties

#mongoDB connection properties

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database= NTSPBMS715DB spring.data.mongodb.username=testuser spring.data.mongodb.password=testuser

step3) Develop the Document class

=====

These are useful

we can perform CURD Operations on MonogoDB collection using spring data mongoDB in two ways (a) using MongoRepository() (best) (b) Using MongoTemplate (c)

for establishing connection with MongoDB software

note: if ur working with MongoRepository then we need to take custom Repository for every @Document class where as by using single MongoTemplate we can perform CURD operations on multiple @Document classes

@Document

if mongoDB s/w Logical DB is created with out having username and password then adding them in application for connectivity is also completely optional

@Data

public class Employee {

@Id

private String id;

private Integer eno; private String ename;

private String eadd; private Double salary; private Boolean isVaccinated;

Using Version1 approach to generate the id values

}

=>The properties count in the @Document class will keep on changing as the no.of attributes increasing the documents of the collections

=> While designing and developing Document class it is recomandded for not adding @NonNull constraint becoz the document of the collections do not have fixed attributes i.e they will have dynamically growing attributes

step4) develop Reposistory Interface extending from MongoRepository(1)

```
package com.nt.repository;
import org.springframework.data.mongodb.repository.MongoRepository; import com.nt.document. Employee;
public interface IEmployee Repo extends MongoRepository<Employee, String> {
}
```

MongoRepository<T, ID>

A saveAll(Iterable<S>) <S extends T> : List<S>

A findAll(): List<T>

A findAll(Sort): List<T>

A insert(S) <S extends T> : S

Ainsert(Iterable<S>) <S extends T> : List<S>

A findAll(Example<S>) <S extends T> : List<S>

A findAll(Example<S>, Sort) <S extends T> : List<S>

upto Spring boot 2.x Repository(1)

extends

(0 methods)

CrudRepository(1) (12 methods)

Common Repository

given by Spring data commons

From Spring boot 3.x

=======

=========

Repository(1)

extends PagingAndSortingRepository (1) (2)

CrudRepository(1)

extends

MongoRepository(1) (7)

This Repository is specific to MongoDB

ListCrudRepository(1)

step5) Develop the service interface and service Impl class

//Service Interface

//IEmployeeMgmtService.java

```
package com.nt.service;
import com.nt.document.Employee;
public interface IEmployeeMgmtService {
public String saveEmployee(Employee e);
}
```

PagingAndSortingRepository(1)

Common Repository Interfaces

ListPagingAndSortingRepository(1)

MongoRepository(1) (Specific to MongoDB)

MongoRepository(1) direct methods

Service Impl class

//EmployeeMgmtServiceImpl.java

@Service("empService")

public class EmployeeMgmtServiceImpl implements IEmployee MgmtService {

@Autowired

private IEmployee Repo empRepo;

=======

MongoRepository<T, ID>

========

021218

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

•^insert(Iterable<S>) <S extends T> : List<S> (alternate to saveAll(-) method)

*insert(S) <S extends T> : S (alternate to save(-))

@Override

public String saveEmployee (Employee e) {

return "MongoDB Doc is saved with id value :"+empRepo.insert(e).getId();

}

step6) Devleop the runner class invoking the service class b.method

public class MongoRepositoryTestRunner implements CommandLineRunner {

@Autowired

private IEmployee MgmtService service;

@Override

public void run(String... args) throws Exception {

Employee e=new Employee();

e.setEno(104); e.setEname("suresh"); e.setEadd("delhi"); e.setSalary(90000.0); e.setIsVaccinated(true);
System.out.println(service.saveEmployee(e));

}

}

}//run(-)

Employee > eno

_id

eno

ename

eadd

salary

isVaccinated

_class

[id] 630d6d95fff45b5f73392a74

123 101.0

"_" raja

"_" delhi

id 630d6dccfff45b5f73392a75

123 456.0

"_" rajesh

"_"hyd

123 10000.0

id 630d7a6d7d3722502834da33

32 104

"_" suresh

"_" delhi

123 90000.0

T/F true

====

To fetch All document of a Collection

==========

In service Interface

public List<Employee> showAllEmployees();

In service Impl class

@Override

public List<Employee> showAllEmployees() {

return empRepo.findAll();

}

In Runner class

"_"com.nt.document.Employe

If we insert docs to mongodb collection from any spring data mongoDB application (in fact any api based app) then

we get one additional key in the mongoDB document that is "_class" having @Document class name

{

"_id" : ObjectId("66194180b80e5e729f27f1ba"), "name": "raja",

"category": "hero",

}

The java class which

hold this document

"remuneration" : 456788.0,

"dob" : ISODate("1989-10-20T07:15:10.000+0000"), "single": false,

"_class": "com.nt.document.ArtistInfo"

note:: Both id and class are the dynamically generated keys in json doc that is

inserted in mongoDB using spring data mongiDB app

=> MongoDB docs of the collection does not show the keys

that are having "null" values or no values

=> Oracle DB table cols show empty cols in the recors representing the null values or no values

=> Java Object holds null values or default values or initial value when no data is bound explicitly

Spring data provides unified

env.. to work with both

SQL and NoSQL DB s/ws..

//========== findAll Documents method====

===== list all documents ==========

service.showAllEmployees().forEach(System.out::println)

Batch insertion of documents using saveAll(-) method (try as assignment)

==========

In service Interface

findout different b/w saveAll(-) and insert(Iterable<S>) methods?

public String registerMultipleProducts(List<Product> prods);

In service Impl class

@Override

public String register Multiple Products (List<Product> prods) {

List<Product> saved Prods=prodRepo.saveAll(prods);

List<String> listids=saved Prods.stream().map(Product::getId).collect(Collectors.toList());

return "multiple products are saved with id values "+listids;

}

In runner class

====saveAll(-) method===

try {

Product prod=new Product();

prod.setPname("table"); prod.setPrice(9000.0);

prod.setQty(10.0);

CURD Operations in MongoDB using MongoRepository(1) methods

=====

using findById(-) method (select a single documment based on given id)

In service Interface

```
public String searchEmployeeById(String idVal);
```

In service Impl class

```
@Override
public String searchEmployee Byld(String idVal) {
Optional<Employee> opt=empRepo.findById(idVal);
}
if(opt.isEmpty())
return " Document not found";
else
return opt.get().toString();
```

In runner class

```
System.out.println("Doc info:"+service.searchEmployeeById("630d6dccfff45b5f73392a75"));
```

Updating the Document of MongoDB

=>save(-) method performs both save object and update object operations becoz it internally calls persist(-) for save document operation and merge(-) for update document operation note: The insert(-) of MongoRepository(1) can perform only save document operation where as save(-) of MongoRepository(1) can perform either save document or update document operation service Interface

```
public String modifyEmployee Byld(String idVal,Double newSal);
```

Difference b/w save(-) of CrudRepository and insert(-) of MongoRepository

service Impl class

```
@Override
public String modify EmployeeById(String idVal, Double newSal) {
Optional<Employee> opt=empRepo.findById(idVal);
if(opt.isEmpty())
return " Document not found";
else {
Employee emp=opt.get();
emp.setSalary(newSal);
}
}//method
empRepo.save(emp);
return "Document found and updated";
```

in Runner class

```
//=======Updating the Document ======
System.out.println(service.modifyEmployee Byld("630d6dccfff45b5f73392a75", 27000.0));
```

Removing the document

=========

In service Interface

```
public String removeEmployee ById(String idVal);
```

service Impl class

Another Example on Removing Docs

==========

```
@Override
@Override
public String removeEmployee ById(String idVal) {
Optional<Employee> opt=empRepo.findById(idVal);
if(opt.isEmpty())
return " Document not found";
public String remove Products(String name, String status) {
//prepare Docuemnt class obj having given data
Product prod=new Product(); prod.setStatus("active"); prod.setName("table");
//prepare Example object
Example<Product> example-Example.of(prod);
else {
//get the documents
empRepo.deleteById(idVal);
List<Product> list-prodRepo.findAll(example);
return "Document found and deleted";
//delete the docs
}
prodRepo.deleteAll(list);
}
return list.size()+" no.of docs are deleted";
}
```

In runner class

```
//====== Deleting the document=======
System.out.println(service.removeEmployee ById("630d6dccfff45b5f73392a75"));
```

Getting documents by applying Sorting

In service interface

```
public List<Employee> showAllEmployees (boolean asc,String ...properties);
```

In service Impl class

```
@Override
```

```
public List<Employee> showAllEmployees (boolean asc, String... properties) {
// create the Sort object
Sort sort=Sort.by(asc? Direction.ASC:Direction.DESC,properties);
//Get the docs by Sorting
List<Employee> list=empRepo.findAll(sort);
return list;
}
```

In Runner class

```
// find all the docs
service.showAllEmployees(true, "ename").forEach(System.out::println);
```

Generating Id value of MongoDB Document explicilty using UUID,GUID Generators as hexadecimal values

===

UUID:: Universal Unique Id

These two thrid party generators generate 32 digit hexa decimal number as

GUID :: Global Unique Id

the unique id value for the docs by using the following details

a)

System date and time

b) System IP address

c) Current Process Id

and etc..

==========

Document class

```
@Document(collection = "Employee")
@Data
public class Employee {
@Id
private String id;
private Integer eno;
private String ename;
private
String eadd;
private Double salary;
private Boolean isVaccinated;
```

code in service Interface

```
public String saveEmployee(Employee e);
```

Code in serivce Impl class

```
@Override
public String saveEmployee(Employee e) {
return "MongoDB Doc is saved with id value :"+empRepo.insert(e).getId();
}
```

Code in Runner class

```
Employee e=new Employee();
e.setId(UUID.randomUUID().toString());
e.setEno(156); e.setEname("mahesh"); e.setEadd("mumbai"); e.setSalary(90000.0); e.setIsVaccinated(true);
System.out.println(service.saveEmployee(e));
```

note:: Spring Data jpa's entity class can use Hibernate/JPA Generators (12 (HB generators)+ 4 (jpa generators)) to

generate id value for Entity object MongoDB api or spring data mongo db api does not give any built-in generators to generate the id value

but we can use third party generators doing the same.

Iment class

(GUID and UUID)

Taking other than "String id " property as the @Id property

note:: In order to store the mongoDB generrated hexadecimal String value as the id value we need to take the property name as "id" of the String in the document class. To avoid

that problem.. we can make any other property as the @Id property

```
cument(collection = "Employee_Info")
ta
c class Employee {
@Id
private Integer eno;
```

Repository Interface

```
public interface IEmployeeRepo extends MongoRepository<Employee, Integer> {
}
```

```
private String ename;
private
String eadd;
private Double salary;
private Boolean isVaccinated;
```

Service Inteface

```
//IEmployeeMgmtService.java
package com.nt.service;
import java.util.Optional;
import com.nt.document.Employee;
```

```
public interface IEmployeeMgmtService {
public String registerEmployee(Employee e);
public Optional<Employee> showEmployee ById(int id);
}
service Impl class
/EmployeeMgmtServiceImpl.java
@Service("empService")
>ublic class Employee MgmtServiceImpl implements IEmployeeMgmtService {
@Autowired
private IEmployee Repo empRepo;
@Override
public String registerEmployee (Employee e) {
Runner class
package com.nt.runners;
import java.util.Optional;
import java.util.Random;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
return "MongoDB Doc is saved with id value :"+empRepo.insert(e).getEno();
import org.springframework.stereotype.Component;
}
import com.nt.document.Employee;
@Override
public Optional<Employee> showEmployee ById(int id) {
return empRepo.findById(id);
}
}//class
This attribute
of collection will
be mapped with @Id
property of @Document class
import com.nt.service.IEmployeeMgmtService;
@Component
public class Mongo RepositoryTestRunner implements CommandLineRunner {
@Autowired
private IEmployee MgmtService service;
@Override
```

```
public void run(String... args) throws Exception {
/* //create Document object
Employee e=new Employee();
e.setEno(new Random().nextInt(10000));
e.setEname("lokesh"); e.setEadd("delhi");
e.setIsVaccinated(true); e.setSalary(90000.0);
System.out.println(service.registerEmployee(e));*/
Optional<Employee> opt-service.showEmployee Byld(100);
if(opt.isPresent())
System.out.println("employee details ::"+opt.get());
else
}
}
System.out.println("Employee not found");
Employee Info > _id
id
ename
eadd
salary
isVaccinated
class
32 100
"_"lokesh
delhi
123 90000.0
TF true
32 124
"_"lokesh
"_" delhi
123 90000.0
T/F true
""com.nt.document.Employee
"_"com.nt.document.Employee
```