Spring Boot MongoDB Using MongoTemplate

==========

on

==========

=>MongoTemplate is given based TemplateMethod Design pattern that says i take care of common logics.. and u just perform specific logics development.

=>This is very much similar to working with JdbcTemplate, HibernateTemplate,JndiTempate classes. (which are given based on template method DP)

arę

=> if the persistence operations simple then prefer using MongoRepository style persistence logics..

=> if the persistence operations complex then prefer using MongoTemplate style persistence logics ..

note:: if needed u can place both styles of persistence logic in one application.

note:: Performing bulk non-select operations is bit complex using MongoRepository..

that is very much simplified in MongoTemplate.

note:: Working with Complex queries is bit difficult in MongoRepository.. that process is simplified in MongoTemplate.

note: MongoTemplate provides both direct methods and methods with Callback interfaces in order perform to persistence opreations

=>callback interfaces provide callback methods allowing us to write logics directly in native api like jdbc api, mongo api, hibernate api etc.. by using container supplied reday made objects.

if we add springboot MongoDB starter to the Project, the MongoTemplate class object will come automatically as springBean through auto configuration.. this object can be injected to service impl class in order to use for persistence operations..

with

note:: While working MongoTemplate .. there is no need of taking MongoRepository..

public class MongoTemplate

extends Object

implements MongoOperations, ApplicationContextAware, IndexOperationsProvider note:: MongoRepository is always tied to One document class.. where as MongoTemplate obj is not

spring data MongoDB can be used

in two ways

a) Using MongoRepository

b) Using MongoTemplate

=>The method that will be called automatically is called callback method =>The interface that contains the decl of callback method is called callback interface we

=> when implement callback method ..we get some container supplied objects as the arguments of callback methods and we can use those object to write customized logics to get customized results.

specific to any Document class i.e one MongoTemplate object can be used to perform persistence operations on multiple Document classes Procedure to develop SpringBootMongoDB Application using MongoTemplate

step1) create springBoot starter project adding SpringbootMongoDB, lombok api staters.

step2) Develop

@Document

Document class.

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

public class StockDetails { @Id

private Integer stockId;

private String stockName;

private double price;

private String exchangeName;

For MongoTemplate APIs docs refer this url:: https://docs.spring.io/spring-data/mongodb/docs/current/api/org/springframework/data/mongodb/core/MongoTemplate.html#method-summary

imp

}

step3) develop application.properties having MongoDB connection, auth properties

#MongoDB Connection,Auth Properties spring.data.mongodb.host=localhost

spring.data.mongodb.database=NTSPBMS615DB

spring.data.mongodb.port=27017

spring.data.mongodb.username=testuser

spring.data.mongodb.password=testuser

step 4) Develop service Inteface and service Impl class

//service interface

public interface IStockMgmtService {

public String registerStockDetails(StockDetails details);

}

//service Impl class

@Service("stockService")

public class StockMgmtServiceImpl implements IStockMgmtService {

@Autowired

private MongoTemplate template;

@Override

public String registerStockDetails(StockDetails details) {

//int idValue-template.save(details, "Stock").getStockId(); // given doc object data will be saved by creting collection called "Stock" ///int idValue=template.save(details).getStockId(); // given doc object data will be saved by creating collection with the Document class name

```
int idValue=template.insert(details).getStockId(); // given doc object data will be saved by creating collection with the Document class name return "Document is saved with id value ::"+idValue;
}
}
```

step5) Develop the runner class

```
//Runner class
@Component
public class MongoTemplateTestRunner implements CommandLineRunner {
@Autowired
private IStockMgmtService service;
@Override
public void run(String... args) throws Exception {
System.out.println("------- save document operation-----
");
StockDetails details=new StockDetails(new Random().nextInt(1000), "ICICI",99999, "BSE"); String result=service.registerStockDetails(details);
System.out.println(result);
}
}
```

What is the difference b/w insert(-) and save(-) method of MongoTemplate?

as

=>insert(-) method supports only insert document opeartion where save(-) supports both insert document, update document operation

=>insert(-) method support bulk/batch insertion by taking collection of Document class objs where as save(-) does not support the same

note:: upsert(-) and save(-) method funcationality is same

collection name

insertAll(Collection) or insert(Collection, String)

==========

=>Given to perform bulk insertin/batch insertion of documents.. This method takes collection of documents to save to MongoDB

In service Interface

```
public String registerStockDetailsBatch(List<StockDetails> list);
```

In service Impl class

```
@Override
public String registerStockDetailsBatch(List<StockDetails> list) {
int size=((List<StockDetails>) template.insertAll(list)).size(); return size+" no.of documents are saved";
}
```

In runner class

```
System.out.println("------- insertAll(-) to save multiple document
-");
```

StockDetails details1=new StockDetails(new Random().nextInt(1000), "ICICI", 99999, "BSE"); StockDetails details2=new StockDetails(new Random().nextInt(1000), "SBI",999, "BSE"); StockDetails details3=new StockDetails(new Random().nextInt(1000), "Bajaj",888, "BSE"); String msg=service.registerStockDetailsBatch(List.of(details1,details2,details3));

```
System.out.println(msg);
```

note:: once we specify our choice collection name in the save(-) method.. to continue all the persistence operations on that document we need Specify collection name every time becoz MongoTemplate object is not specific to one Collection. It is common for multiple collections

Every Repository interface of spring data module binds to one entity/document class.. where the Template class obj do not bind to any entity/ document class and the Template class obj is common for workign with all entity /document classes.

finding documents with Conditions/Cirterias

======

======

we can use find(Query query, ...) method for this operation. where Query object represents the Criteria condition (where clause condition through java statements)

In Service Interface

public List<StockDetails> fetchStocks DetailsByExchange(String exchange);

public List<StockDetails> fetchStocksDetailsByPriceRange(double startPrice,double endPrice);

In service Impl class

```
@Override
```

public List<StockDetails>fetchStocks DetailsByPriceRange(double startPrice, double endPrice)

```
{
```

Query query=new Query();

query.addCriteria(Criteria.where("price").gte(startPrice).lte(endPrice)); List<StockDetails> list=template.find(query, StockDetails.class); return list;

}

```
@Override
```

public List<StockDetails>fetchStocks DetailsByExchange(String exchange) {

Query query=new Query();

query.addCriteria(Criteria.where("exchangeName").is(exchange)); List<StockDetails> list=template.find(query, StockDetails.class); return list;

```
}
```

In runner class

//System.out.println("------- find(-,-0) for selecting the documents

service.fetchStocksDetailsByExchange("BSE").forEach(System.out::println);

System.out.println("-

-------");

-");

service.fetchStocks DetailsByPriceRange(500, 10000).forEach(System.out::println);

findById(id, doc class), findById(id,doc class, collection name) of MongoTempate

=========

=>This method is given to search and get single Document object based on given id value. code in service impl class

signature :: public <T> T findById(Object id, Class<T> entityClass)

findById(--) --> for single doc retrieving findAll(-) -->for all docs retrieving find(Query,-,-) --> for retrieving single or

multiple docs based on condition

@Override

public StockDetails fetchStockDetailsByStockId(int stockid) { return template.findById(stockId,StockDetails.class);

}

Code in runner class

System.out.println("770 stockId Stock details are ::"+service.fetchStockDetailsByStockId(770));

findAndModify(Query,Update,doc class)

=>performs single doc retrieving based on given Query obj condititon

and modifies the doc with the given Update object data.

signaure ::

public <T> T findAndModify(Query query, UpdateDefinition update, Class<T>

docClass)

code in serivce impl class

}

@Override

public String fetchAndUpdateStockDetailsByStockId(int stockId, double newPrice, String newExchangeName) {

//Query object for single doc retrieving

Query query=new Query();

query.addCriteria(Criteria.where("stockId").is(stockId));

//Update object for modification

Update update=new Update();

update.set("price",newPrice);

update.set("exchangeName", newExchangeName);

//call the method

This method updates only single doc .. not the multiple docs

StockDetails details=template.findAndModify(query, update, StockDetails.class); if the Query object codition

return details==null?"stock does not found":"stock found and updated";

code in runner class

returns the multiple docs

then it updates the first

doc..

System.out.println(service.fetchAndUpdateStockDetailsByStockId(770,56789.7,"NSE"));

updateMulti(Query, update, doc class)

the

=>This method is useful to perform bulk update operation for the given Query condition with given Update object

data.

signuare :: public UpdateResult updateMulti(Query query, UpdateDefinition update, Class<?> entityClass)

service impl class

@Override

}

public String modifyExchangeByStockPriceRange(double startPrice, double endPrice, String newExchangeName) { //Query object for single doc retrieving

Query query=new Query(); query.addCriteria(Criteria.where("price").gte(startPrice).and Operator(Criteria.where("price").lte(end Price)));

//update object

Update update=new Update().set("exchangeName", newExchangeName);

//call the method

UpdateResult result-template.updateMulti(query, update, StockDetails.class);

return result.getModifiedCount()+" no.of records are effected";

S

code in runner class

System.out.println(service.modifyExchangeByStockPriceRange(400, 60000, "NYKSE"));

upSert(Query, Update, doc class)

=> capable performing insert or update operation.. (single document)

=>

if doc is not found for the given Query object condition then it will try to insert new documennt with given Update object data

=> if doc is found for the given Query object condition then it will try to update the doc with given Update object data.

public UpdateResult upsert(Query query, UpdateDefinition update, Class<?> entityClass)

code in service Impl class

@Override

public String registerOrUpdateStockByStockName(String stockName, double newPrice, String newExchange) {

```
//Query object for single doc retrieving
Query query=new Query();
//update object
query.addCriteria(Criteria.where("stockName").is(stockName));
Update update=new Update();
update.set("exchangeName",newExchange);
update.setOnInsert("stockName", stockName),
update.setOnInsert("stockId",new Random().nextInt(10000));
update.set("price",newPrice);=
//invoke the method
will be used for both
insertion and updation
Will be used only for insertion
UpdateResult result-template.upsert(query, update, StockDetails.class);
if(result.getModifiedCount()==0)
return " new Documnet is inserted with id value :"+result.getUpsertedId();
else
}
return "Existing doc is updated";
code in runner class
System.out.println(service.registerOrUpdateStockByStockName("SBH",7890,"CHSE"));
```

=>upsert(-) can perform single doc updation or insertion .. if the Quey obj condition gives multiple docs then it picks first doc from that list to perfrom update operation. if .. Query object condition does not give any doc or

docs then it performs insert doc operation using the data given in update object.. =>upsert(-) method and save(-) is similar... The only difference is save(-) takes id value as the criteria value where as upsert(-) take given. Query object condition data asfiteria value.

upsert(-,-) = update +insert operation

findAndRemove(Query, doc class)

=> To perform single doc removing operation by finding it through Query object condition.

public <T> T findAndRemove(Query query, Class<T> entityClass)

=>if Query object condition finds multiuple docs then it will delete the first document from the list ...

In service impl class

```
@Override
public String fetchAndRemoveByStockName(String stockName) {
//Query object for single doc retrieving
Query query=new Query();
query.addCriteria(Criteria.where("stockName").is(stockName));
```

```
//call the method
StockDetails details= template.findAndRemove(query, StockDetails.class); return details==null?"stockNotFound":" Stock Found and deleted";
}
```

In client App

==========

```
System.out.println(service.fetchAndRemoveByStockName("SBH"));
```

findAllAndRemove(Query, doc class)

(Assignment)

=> To perform bulk delete operations of the doc based on given Query object condition.

```
public <T> List<T> findAllAndRemove(Query query, String collectionName)
@Override
public String removeStocksByPriceRange(double start, double end) {
Query query=new Query();
query.addCriteria(Criteria.where("price").gte(start).lte(end)); int count-template.findAllAndRemove(query, StockInfo.class).size(); return count+" no.of docs are removed";
}
```