Date : 18/04/2021
Spring Boot 9AM
Mr. RAGHU

MongoDB : Template Example

Ref: https://docs.mongodb.com/manual/aggregation/

MonogoTemplate(C):-
 *) MonogoTemplate is auto-configured class given by Spring
    Data MongoDB, used to perform all operations over
    MongoDB Collection.

*) When we use MongoTemplate(C), no need of writing
   MongoRepository interface.

*) Directly use MongoTemplate(C) [Autowired] inside your app.

*) Query based Operations:-
   We can define Query with Criteria which is also called
   as WHERE condition to execute FETCH/UPDATE/DELETE operations

  => Criteria provides methods for all operations
     lt, ge, and, or, not, is , ...etc

 find(q,class) : used to fetch matching data
 findAndModify(q,update,class): Used to modify given Updates
 findAndRemove(q,class) : used to delete selected data

--Example---------
```
 q.addCriteria(
   Criteria.where("stdName").is("C")
        .and("stdFee").gt(50.0)
        .and("stdName").ne(null)
        );
```

==========Full Code==========

1. SpringBoot App
name: SpringBoot2MongoDbTemplate
Dep : MongoDB, Lombok

2. 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 Student {
```

```
	@Id
	private Integer stdId;
	private String stdName;
	private Double stdFee;
}
```

3. properties

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nitone
```

4. Runner class

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.Student;

@Component
public class TestTemplateRunner implements CommandLineRunner {
	@Autowired
	private MongoTemplate mt;

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

		//1. save (insert/update)
		/*mt.save(new Student(10, "A", 3.3));
		mt.save(new Student(11, "B", 2.3));
		mt.save(new Student(12, "C", 4.3));
		*/
		//2. fetch data and print
		/*List<Student> list = mt.findAll(Student.class);
		list.forEach(System.out::println);
		*/

		//3. conditional based (Query) update/fetch/delete

		// ..where stdName='C'
		Query q = new Query();
		//q.addCriteria(Criteria.where("stdName").is("C"));
		q.addCriteria(
				Criteria.where("stdName").is("C"));

		//a. fetch data
		/*mt.find(q, Student.class)
		.forEach(System.out::println);
		*/

		//b. update data
		// update student set stdFee=9.9
		/*Update u = new Update();
		u.set("stdFee", 9.9);
```

```
                //u.set("stdName", "CAM");

                mt.findAndModify(q, u, Student.class);
                */

                //c. remove data
                mt.findAndRemove(q, Student.class);
                System.out.println("DONE");

        }

}
```

==========

*) MongoDB Aggregation concepts:-
  Executing large set of operations over collections data

AggregationOperation : condition, sort, group, project ..etc

*) Aggregation(C) supports
  -> creating new Aggregation object
  -> providing contiditional test
  -> supports data sort
  -> execute field projections
  -> converts source format to result class type

--Additional code-----------

1. model

```
package in.nareshit.raghu.model;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
public class MyResult {

        private String stdName;
}
```

2. Runner

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.aggregation.Aggregation;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.MyResult;
import in.nareshit.raghu.model.Student;

@Component
public class TestAggrRunner implements CommandLineRunner {
```

```
	@Autowired
	private MongoTemplate mt;

	public void run(String... args) throws Exception {
		//creation
		Aggregation ag = Aggregation.newAggregation(

Aggregation.match(Criteria.where("stdFee").gt(1.1)),

Aggregation.sort(Direction.DESC,"stdName"),
				Aggregation.project("stdName")
				);
		//execute (aggregate, source, ResultClasstype)
		mt.aggregate(ag, Student.class,MyResult.class)
		.forEach(System.out::println);
	}

}
```

==========With Lookup process using DBRef==========

1. Models

```
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 Address {
	@Id
	private Integer id;
	private String hno;
	private String loc;
	private Long pinCode;
}
--
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 Department {
	@Id
	private Integer deptId;
```

```
        private String deptCode;
        private String deptName;

}
--
package in.nareshit.raghu.model;

import java.util.List;
import java.util.Map;
import java.util.Set;

//ctrl+shift+O
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.DBRef;
import org.springframework.data.mongodb.core.mapping.Document;

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

@Data
@Document
@NoArgsConstructor
@AllArgsConstructor
public class Employee {

        @Id
        private Integer empId;
        private String empName;
        private Double empSal;

        private Set<String> empPrjs;
        private List<String> empPrjVer;
        private String[] empGrades;

        private Map<String,String> empClient;
        @DBRef
        private Address addr;//HAS-A
        @DBRef
        private List<Department> dobs; //HAS-A


}

2.proeprties
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nitone

3. Repository
package in.nareshit.raghu.repo;
import org.springframework.data.mongodb.repository.MongoRepository;
import in.nareshit.raghu.model.Employee;
public interface EmployeeRepository
        extends MongoRepository<Employee, Integer> {

}
```

```
--
package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Address;

public interface AddressRepository
        extends MongoRepository<Address, String> {

}
--
package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Department;

public interface DeptRepository
        extends MongoRepository<Department, String> {

}
```

4. Data Insert Runner

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

import java.util.List;
import java.util.Map;
import java.util.Set;

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

import in.nareshit.raghu.model.Address;
import in.nareshit.raghu.model.Department;
import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.AddressRepository;
import in.nareshit.raghu.repo.DeptRepository;
import in.nareshit.raghu.repo.EmployeeRepository;

@Component
@Order(1)
public class DataInsertRunner implements CommandLineRunner {

        @Autowired
        private EmployeeRepository repo;
        @Autowired
        private AddressRepository arepo;
        @Autowired
        private DeptRepository derepo;

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

```
                Address addr = new Address(109,"8-9/A", "HYD",
500032L);
                arepo.save(addr);

                List<Department> dobs = List.of(
                                new Department(11,"D1", "DEV-AB"),
                                new Department(12,"D2", "QA-RB"),
                                new Department(13,"D3", "SUPRT-MN")
                        );
                derepo.saveAll(dobs);

                repo.save(
                                new Employee(
                                                10, "SAM", 200.0,

Set.of("HTC","NIT","ORCL"),
                                                List.of("3.2GA","6.5
RELEASE","0.1 ALPHA"),
                                                new String[]
{"A+","GR-T","UI-NEW"},
                                                Map.of("C1","TEC-
N","C2","US-ARMY","C3","JANSON & JANSON"),
                                                addr,
                                                dobs

                                                )
                                );

        }

}
```

5. Data Fetch Runner

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

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.core.annotation.Order;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.aggregation.Aggregation;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.Employee;

@Component
@Order(2)
public class DataFetchRunner implements CommandLineRunner {
        @Autowired
        private MongoTemplate mt;

        public void run(String... args) throws Exception {
                Aggregation aggr =
```

```
                                Aggregation.newAggregation(

Aggregation.match(Criteria.where("_id").is(10)),

Aggregation.sort(Direction.DESC,"empName"),

Aggregation.project("empName","addr","dobs")
                                                    );

                List<Employee>  list = mt.aggregate(aggr, "employee",
Employee.class)
                                .getMappedResults();
                list.forEach(System.out::println);
        }

}
```