Date: 14/04/2021
Spring Boot 9AM
Mr. RAGHU

MongoDB Complex Types using Embeddded and DBRef

Embeddded : Using single collection to store both
parent and child models class obejcts as
JSON (inner JSON/Embedded JSON) Format.

```
{
  //parent
  hasAVariable : {
     //child JSON
  }

}
```

DBRef: Using 2 collection for 2 Models and creating
a link b/w JSON Documents, for re-using of same
data it helps.

```
child JSON
{
  _id:---
}

parent json
{
   var : ("child",_id);    //reference
}
```

*) it is like PK-FK link.
*) There is no such annotations like
@OneToMany, @ManyToOne ..etc
Use Embeeded and DBRef only

Embeeded	DBRef
1...1	*...1
1...*	*...*

=> When Parent is 1(one) the use Embedded
When Parent is *(mny) use DBRef.

=> Every Default HAS-A (Association Mapping) is Embedded
only. To use DBRef concept use annotation @DBRef

==========

Embedded Example:

```
        1...1
  Student---<>Address
        1...*
  Student---<>Subject
```

DBRef Example:

```
          *...1
  Employee ---<> Department
          *...*
  Employee ---<> Project

=> Define Every model as a collection(table) in MongoDB
   using annotations @Document

=> Define PK variable in Child Model class even, @Id
=> Provide Repository Interfaces for all models
=> Save child data before parent data.

======Ex#1 Embedded Code===========
Name : SpringBoot2MongoDBEmbeddedEx
Dep  : Spring Data MongoDB, Lombok

1. Model
package in.nareshit.raghu.model;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Subject {

        private String code;
        private Integer mrks;
}
--
package in.nareshit.raghu.model;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Address {

        private String hno;
        private String loc;
}
---
package in.nareshit.raghu.model;

import java.util.List;


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 sid;
	private String sname;
	private Double sfee;

	private Address addr;
	private List<Subject> sobs;

}
```

2. Repository

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

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

import in.nareshit.raghu.model.Student;

public interface StudentRepository
	extends MongoRepository<Student, Integer> {

}
```

---

3. properties

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

4. Runner

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

import java.util.Arrays;

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

import in.nareshit.raghu.model.Address;
import in.nareshit.raghu.model.Student;
import in.nareshit.raghu.model.Subject;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class StudentDataInsertRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo;

	public void run(String... args) throws Exception {
		repo.deleteAll();
		repo.save(
				new Student(101, "SAM", 300.0,
```

```
                                                        new Address("8-96/A",
"CHN"),
                                                        Arrays.asList(
                                                                        new
Subject("ENG", 98),
                                                                        new
Subject("MAT", 99),
                                                                        new
Subject("SCI", 100)
                                                                        )
                                                        )
                                );

        }

}

--5. output-----
{
        "_id" : 101,
        "sname" : "SAM",
        "sfee" : 300,
        "addr" : {
                "hno" : "8-96/A",
                "loc" : "CHN"
        },
        "sobs" : [
                {
                        "code" : "ENG",
                        "mrks" : 98
                },
                {
                        "code" : "MAT",
                        "mrks" : 99
                },
                {
                        "code" : "SCI",
                        "mrks" : 100
                }
        ],
        "_class" : "in.nareshit.raghu.model.Student"
}

=======Ex#2=====================================
Name : SpringBoot2MongoDBRefEx
Dep  : Spring Data MongoDB, Lombok

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 Department {
        @Id
        private Integer did;
        private String dname;
        private String admin;
}
--
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 Project {

        @Id
        private Integer pid;
        private String pname;
        private String cname;


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

import java.util.List;

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
@NoArgsConstructor
@AllArgsConstructor
@Document
public class Employee {
        @Id
        private Integer eid;
        private String ename;
        private Double esal;

        //*...1
```

```
        @DBRef
        private Department dob;
        //*...*
        @DBRef
        private List<Project> pobs;

}
--------
2. Repository interface
package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Department;

public interface DepartmentRepository
        extends MongoRepository<Department, Integer> {

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

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

import in.nareshit.raghu.model.Project;

public interface ProjectRepository
        extends MongoRepository<Project, Integer> {

}
--
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> {

}

3. properties
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nittest

4. Runner class
package in.nareshit.raghu.runner;

import java.util.Arrays;

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

import in.nareshit.raghu.model.Department;
```

```
import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.model.Project;
import in.nareshit.raghu.repo.DepartmentRepository;
import in.nareshit.raghu.repo.EmployeeRepository;
import in.nareshit.raghu.repo.ProjectRepository;

@Component
public class MyDataInsertRunner implements CommandLineRunner {
        @Autowired
        private DepartmentRepository drepo;

        @Autowired
        private ProjectRepository prepo;

        @Autowired
        private EmployeeRepository erepo;

        public void run(String... args) throws Exception {
                Department d1 = new Department(101, "DEV", "SAM");
                Department d2 = new Department(102, "QA", "SYED");

                drepo.save(d1);
                drepo.save(d2);

                Project p1 = new Project(501, "P1", "HTC");
                Project p2 = new Project(502, "P2", "NIT");
                Project p3 = new Project(503, "P3", "XYZ");
                Project p4 = new Project(504, "P4", "ORCL");

                prepo.save(p1);
                prepo.save(p2);
                prepo.save(p3);
                prepo.save(p4);


                Employee e1 = new Employee(2021, "A", 500.0, d1,
Arrays.asList(p1,p2));
                Employee e2 = new Employee(2022, "B", 600.0, d1,
Arrays.asList(p2,p3));
                Employee e3 = new Employee(2023, "C", 700.0, d2,
Arrays.asList(p3,p4));
                Employee e4 = new Employee(2024, "D", 800.0, d2,
Arrays.asList(p3,p1));

                erepo.save(e1);
                erepo.save(e2);
                erepo.save(e3);
                erepo.save(e4);

        }

}
```

5. output

```
> db.department.find().pretty();
> db.project.find().pretty();
> db.employee.find().pretty();
```

```
--Syntax:
{

  hasAvariable: DBRef("CollectioName", _idValue);
}
```