**MongoDb Database connection with Spring Boot**

**Step 1:** Add mongodb dependency in pom file.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

**Step 2:** Add the below property in .yml file.

**spring:**

data:

mongodb:

host: localhost

port: 27017

database: stocksdb

**Step 3:** Start MongoDb.

To Start the mongo database: **C:\Program Files\MongoDB\Server\5.0\bin>mongod.**

NOTE: In C drive data directory should be available.

To Start the mongo client: **C:\Program Files\MongoDB\Server\5.0\bin>**mongo (Open in new window.)

**Some important commands of mongo**

**show dbs:** to show the database

**db.createCollection(‘stocks’) :** to create collections

**show collections:** to show all collections

NOTE: In mongodb primary key is automatic generated. If you want to override it the we have to provide the primary key.

**Step 4:** Create Document (Table) like.

@Document(collection = "tokens")

**public** **class** JwtBlackListedTokenEntity {

@Id

**private** **int** id;

**private** String jwtToken;

**public** JwtBlackListedTokenEntity() {}

**public** JwtBlackListedTokenEntity(**int** id, String jwtToken) {

**super**();

**this**.id = id;

**this**.jwtToken = jwtToken;

}

// Add getter setter and toString method.

}

**Step 5:** Create mongo repo

Entity relationship type:

1. One to one (Employee & Profile id)
2. One to many (Question & Answer)
3. Many to many (Student & Course)

**One to One**

**@OneToOne(cascade={CascadeType.ALL}, fetch=FetchType.LAZY)**

**Cascade:** means whichever operation you performed on employee table its also applicable for child. Ex: If parent will delete the child will also deleted. (Its use for create, delete, update DML queries).

**Fetch:** Fetch has two types LAZY & EAGER. EAGER will automatically call the child table if we select query for parent table so if we have more fields in profile table then it will create the issue for us. And we are only asking about employee(table). So to resolve this issue we will use LAZY. (Its use for select).

NOTE: Every JPA entity must have one primary key.

@Entity

@Table(name="EMPLOYEE")

public class EmployeeEntity {

@Id

@GeneratedValue

@Column(name="id")

private Long employeeId;

@Column(name="name")

private String name;

@Column(name="sal")

private double sal;

**@OneToOne(cascade={CascadeType.ALL},**

**fetch=FetchType.LAZY)**

**@JoinColumn(name="profile\_id")**

**private ProfileEntity profile;**

REST call details:

1) One to One relationship (Employee & Profile)

A) Create new employee:

http://localhost:8080/myapp/employee [POST]

{

"name": "Jerry",

"sal": 28000,

"profile": {

"company": "Infosys",

"experience": 2.5

}

}

B) Read all employees:

http://localhost:8080/myapp/employee [GET]

**One to Many**

|  |  |
| --- | --- |
|  |  |

In question table

**@OneToMany(cascade={CascadeType.PERSIST, CascadeType.REMOVE}, fetch=FetchType.LAZY, mappedBy = "question")**

**private QuestionEntity question;**

mappedBy: the attribute in answer entity that’s represent the Question table.

In Answer table

**@ManyToOne(cascade={CascadeType.ALL}, fetch=FetchType.EAGER)**

**@JoinColumn(name="question\_id")**

**private QuestionEntity question;**

**REST Call details**

A) Create new question:

http://localhost:8080/myapp/question [POST]

{

"qusDesc": "Which are european countries?",

"answers": [

{"answer": "Sweden"}, {"answer": "France"}, {"answer": "England"}

]

}

B) Read all questions:

http://localhost:8080/myapp/question [GET]

**Many to Many**

In many to many relationships always need to create 3rd table to store the foreign key. Because one student has multiple course or multiple courses for same student. So we can’t identify the course or student.

**In Student table:**

**@ManyToMany(cascade={CascadeType.ALL})**

**@JoinTable(**name = "**STUDENT\_COURSE**",joinColumns = { @JoinColumn(name = "STUDENT\_ID")},inverseJoinColumns = { @JoinColumn(name = "COURSE\_ID") }**)**

**Private Set<CourseEntity> cources=new HashSet<CourseEntity>();**

**Here, “STUDENT\_COURSE” is a new table with two column. First column** "STUDENT\_ID" is a primary key of student table and "COURSE\_ID" is primary key of course table.

**In Course table:**

**@ManyToMany(cascade={CascadeType.ALL})**

**@JoinTable(**name = "**STUDENT\_COURSE**",joinColumns = { @JoinColumn(name = " COURSE\_ID ")},inverseJoinColumns = { @JoinColumn(name = " STUDENT\_ID ") }**)**

**Private Set<StudentEntity> cources=new HashSet<StudentEntity>();**

REST call details

A) Create new student:

http://localhost:8080/myapp/student [POST]

{

"studentName": "Jerry",

"courses": ["Jenkins", "Spring", "Angular"]

}

B) Read all students:

http://localhost:8080/myapp/student [GET]

**NOTE: In ManyToMany releationship there is no parent child concept so we have to save it either with Student table or Course table manually.**