Date : 01-Jul-21
Spring Boot 9AM
Mr. RAGHU

Spring Data REST:
https://www.youtube.com/watch?v=uptKDrR6bLg

Spring Boot Reactive Programming

*) Spring Boot - Producer Application
    @RestController, @GetMapping, @RequestBody,...etc
     Return : Mono<T> or Flux<T>
     (ResponeEntity --Do not support reactive calls)

*) Spring Boot - Consumer Application
    (RestTemplate -- Do not support Reactive Calls)
    WebClient

======(Producer coding steps)===

a. Create one Spring Starter Project
Name : SpringBoot2ReactiveProducer
Dep : Spring Reactive Web, Lombok, DevTools, Reactive MongoDB

b. application.yml

```
server:
  port: 8081

spring:
  data:
    mongodb:
      database: nitdb
      host: localhost
      port: 27017
```

c. model class

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

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import lombok.Data;

@Data
@Document
public class Student {

        @Id
        private Integer sid;
        private String sname;
        private Double sfee;
}
```

d. Repository interface

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

import
org.springframework.data.mongodb.repository.ReactiveMongoRepository;
import in.nareshit.raghu.model.Student;

public interface StudentRepository
        extends ReactiveMongoRepository<Student, Integer>{

}
```

e. Service class

```
package in.nareshit.raghu.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import in.nareshit.raghu.model.Student;
import in.nareshit.raghu.repo.StudentRepository;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Service
public class StudentService {

        @Autowired
        private StudentRepository repo;

        public Mono<Student> saveStudent(Student s) {
                return repo.save(s);
        }

        public Flux<Student> getAllStudents() {
                //return repo.findAll();
                return repo.findAll().switchIfEmpty(Flux.empty());
        }

        public Mono<Student> getOneStudent(Integer id) {
                //return repo.findById(id);
                //return
repo.findById(id).switchIfEmpty(Mono.just(null));
                return repo.findById(id).switchIfEmpty(Mono.empty());
        }

        public Mono<Void> deleteOneStudent(Integer id) {
                return repo.deleteById(id);
        }

}
----------------------------------------------------------------------
```

f. RestController class

```
package in.nareshit.raghu.rest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
```

```java
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import in.nareshit.raghu.model.Student;
import in.nareshit.raghu.service.StudentService;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/student")
public class StudentRestController {

        @Autowired
        private StudentService service;

        /*
        @PostMapping("/create")
        public Mono<String> saveStudent(
                        @RequestBody Student student
                        )
        {
                Mono<String> mono = null;
                try {
                        service.saveStudent(student);
                        mono = Mono.just("Student Created!");
                } catch (Exception e) {
                        e.printStackTrace();
                        mono = Mono.just("Creation Failed!");
                }

                return mono;
        }*/

        @PostMapping
        public Mono<Student> saveStudent(
                        @RequestBody Student student
                        )
        {
                return service.saveStudent(student);
        }

        @GetMapping
        public Flux<Student> getAllStudents() {
                return service.getAllStudents();
        }

        @GetMapping("/{id}")
        public Mono<Student> getOneStudents(
                        @PathVariable Integer id)
        {
                return service.getOneStudent(id);
        }

        @DeleteMapping("/{id}")
        public Mono<Void> deleteStudent(
                        @PathVariable Integer id
                        )
        {
                return service.deleteOneStudent(id);
        }

}
---(MongoDB)----------------------------------------------------------
------
a. Start Monogdb Server
cmd> mongod

b. Client Commands
cmd> mongo

> show dbs
> use nitdb
> show collections
> db.student.drop()
> db.student.find().pretty()

======================================================================
===
```