Spring Webflux Reactive Programming with MongoDB

Lets create an Employee Management System. To make it fully non-blocking, lets use **mongodb** as back-end database.

1. File -> New -> Spring Starter Project

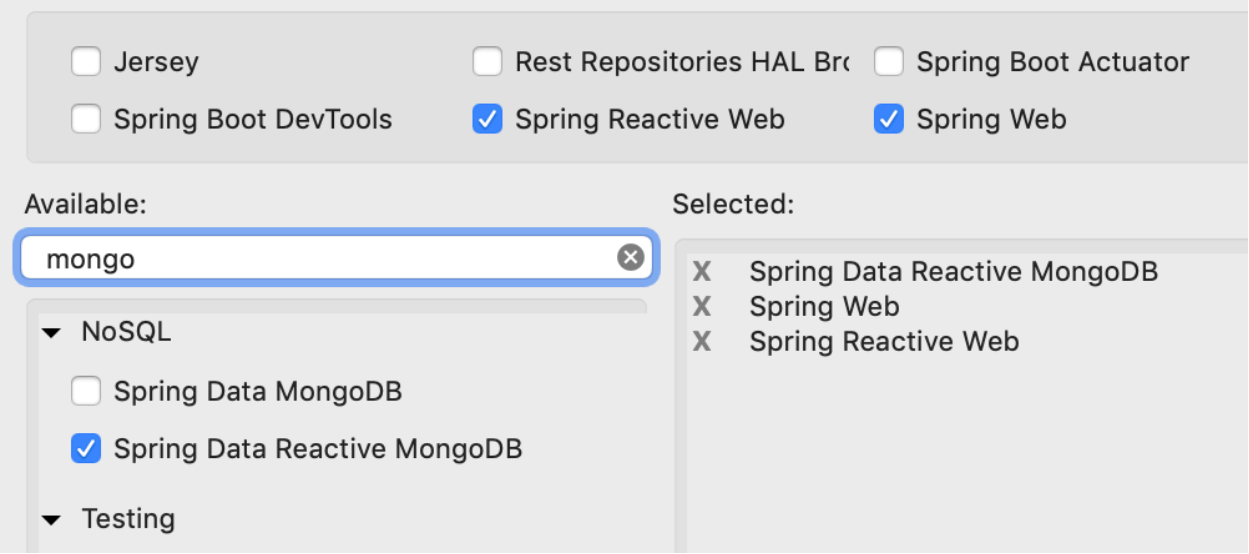
2. Name = mongodb.reactive

3. Group = mongodb.reactive

4. Package = com.demo

5. Click on Next

6. Select these modules as shown below:



7. Configurations

7.1 Webflux Configuration

WebFluxConfig.java

```
package com.demo.config;

import
org.springframework.context.annotation.Configuration;
import
org.springframework.web.reactive.config.EnableWebFlux;
import
org.springframework.web.reactive.config.WebFluxConfigurer;

@Configuration
@EnableWebFlux
public class WebFluxConfig implements WebFluxConfigurer
{
}
```

7.2 MongoDB Configuration

MongoConfig.java

```
package com.demo.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.mongodb.config.AbstractReactiveMongoCo
nfiguration;
import
org.springframework.data.mongodb.core.ReactiveMongoTemplate;
import
org.springframework.data.mongodb.repository.config.EnableReacti
veMongoRepositories;

import com.mongodb.reactivestreams.client.MongoClient;
import com.mongodb.reactivestreams.client.MongoClients;

@Configuration
@EnableReactiveMongoRepositories(basePackages = "com.demo.dao")
public class MongoConfig extends
AbstractReactiveMongoConfiguration
{
    @Value("${port}")
    private String port;

    @Value("${dbname}")
    private String dbName;

    @Override
    public MongoClient reactiveMongoClient() {
        return MongoClients.create();
    }

    @Override
    protected String getDatabaseName() {
        return dbName;
    }

    @Bean
    public ReactiveMongoTemplate reactiveMongoTemplate() {
        return new ReactiveMongoTemplate(reactiveMongoClient(),
getDatabaseName());
    }
}
```

7.3 Application Configuration

```
package com.demo.config;

import
org.springframework.beans.factory.config.PropertyPlacehold
erConfigurer;
import org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;

@Configuration
public class AppConfig
{
    @Bean
    public static PropertyPlaceholderConfigurer
getPropertyPlaceholderConfigurer()
    {
        PropertyPlaceholderConfigurer ppc = new
PropertyPlaceholderConfigurer();
        ppc.setLocation(new
ClassPathResource("application.properties"));
        ppc.setIgnoreUnresolvablePlaceholders(true);
        return ppc;
    }
}
```

8. Properties File for MongoDB config

src/main/resources/application.properties

```
port=27017
dbname=testdb
```

9. Logging Configuration

logback.xml

```
<configuration>

    <appender name="STDOUT"
        class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level
%logger{5} - %msg%n
            </pattern>
        </encoder>
    </appender>

    <logger name="org.springframework" level="DEBUG"
        additivity="false">
        <appender-ref ref="STDOUT" />
    </logger>

    <root level="ERROR">
        <appender-ref ref="STDOUT" />
    </root>

</configuration>
```

10. REST Controller

```
package com.demo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.demo.model.Employee;
import com.demo.service.EmployeeService;

import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@RestController
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;

    @RequestMapping(value = { "/create", "/" }, method =
RequestMethod.POST)
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public void create(@RequestBody Employee e) {
        employeeService.create(e);
    }

    @RequestMapping(value = "/{id}", method =
RequestMethod.GET)
    @ResponseBody
    public ResponseEntity<Mono<Employee>>
findById(@PathVariable("id") Integer id) {
        Mono<Employee> e = employeeService.findById(id);
        HttpStatus status = e != null ? HttpStatus.OK :
HttpStatus.NOT_FOUND;
        return new ResponseEntity<Mono<Employee>>(e, status);
    }
```

. . . CONTINUED IN THE NEXT PAGE . . .

```
@RequestMapping(value = "/name/{name}", method =
RequestMethod.GET)
    @ResponseBody
    public Flux<Employee> findByName(@PathVariable("name")
String name) {
        return employeeService.findByName(name);
    }

    @RequestMapping(method = RequestMethod.GET, produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
    @ResponseBody
    public Flux<Employee> findAll() {
        Flux<Employee> emps = employeeService.findAll();
        return emps;
    }

    @RequestMapping(value = "/update", method =
RequestMethod.PUT)
    @ResponseStatus(HttpStatus.OK)
    public Mono<Employee> update(@RequestBody Employee e) {
        return employeeService.update(e);
    }

    @RequestMapping(value = "/delete/{id}", method =
RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.OK)
    public void delete(@PathVariable("id") Integer id) {
        employeeService.delete(id).subscribe();
    }

}
```

11. Service Class

11.1 IEmployeeService.java

```
package com.demo.service;

import com.demo.model.Employee;

import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

public interface IEmployeeService
{
    void create(Employee e);

    Mono<Employee> findById(Integer id);

    Flux<Employee> findByName(String name);

    Flux<Employee> findAll();

    Mono<Employee> update(Employee e);

    Mono<Void> delete(Integer id);
}
```

11.2 EmployeeService.java

```java
package com.demo.service;

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

import com.demo.dao.EmployeeRepository;
import com.demo.model.Employee;

import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Service
public class EmployeeService implements IEmployeeService {

    @Autowired
    EmployeeRepository employeeRepo;

    public void create(Employee e) {
        employeeRepo.save(e).subscribe();
    }

    public Mono<Employee> findById(Integer id) {
        return employeeRepo.findById(id);
    }

    public Flux<Employee> findByName(String name) {
        return employeeRepo.findByName(name);
    }

    public Flux<Employee> findAll() {
        return employeeRepo.findAll();
    }

    public Mono<Employee> update(Employee e) {
        return employeeRepo.save(e);
    }

    public Mono<Void> delete(Integer id) {
        return employeeRepo.deleteById(id);
    }

}
```

12. DAO Repository

EmployeeRepository.java

```
package com.demo.dao;

import org.springframework.data.mongodb.repository.Query;
import
org.springframework.data.mongodb.repository.ReactiveMongoReposi
tory;

import com.demo.model.Employee;

import reactor.core.publisher.Flux;

public interface EmployeeRepository extends
ReactiveMongoRepository<Employee, Integer> {

   Flux<Employee> findByName(final String name);
}
```

13. Model Object

Employee.java

```
package com.demo.model;

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

@Scope(scopeName = "request", proxyMode = ScopedProxyMode.TARGET_CLASS)
@Document
public class Employee {

	@Id
	int id;
	String name;
	long salary;

	public Employee() {

	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public long getSalary() {
		return salary;
	}

	public void setSalary(long salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", salary=" +
salary + "]";
	}
}
```

14. Spring Application

WebfluxFunctionalApp.java

```
package com.demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class WebfluxFunctionalApp {

    public static void main(String[] args) {
        SpringApplication.run(WebfluxFunctionalApp.class,
args);
    }
}
```

15. Start MongoDB Server, if not started already

mongodb/bin# mongod

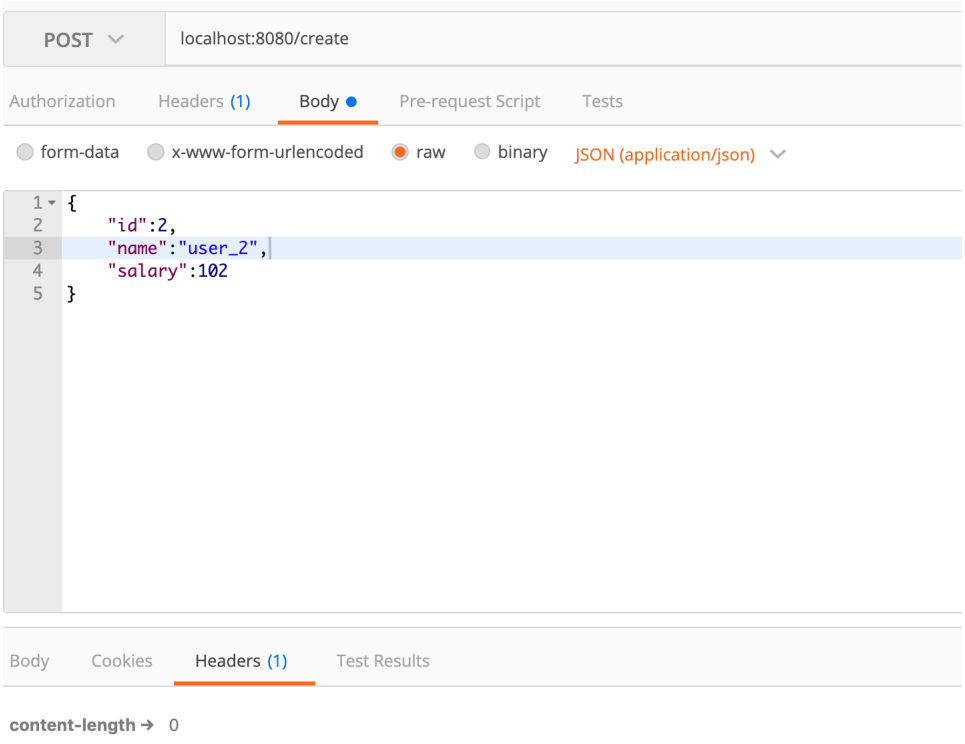
16. RUN the APP

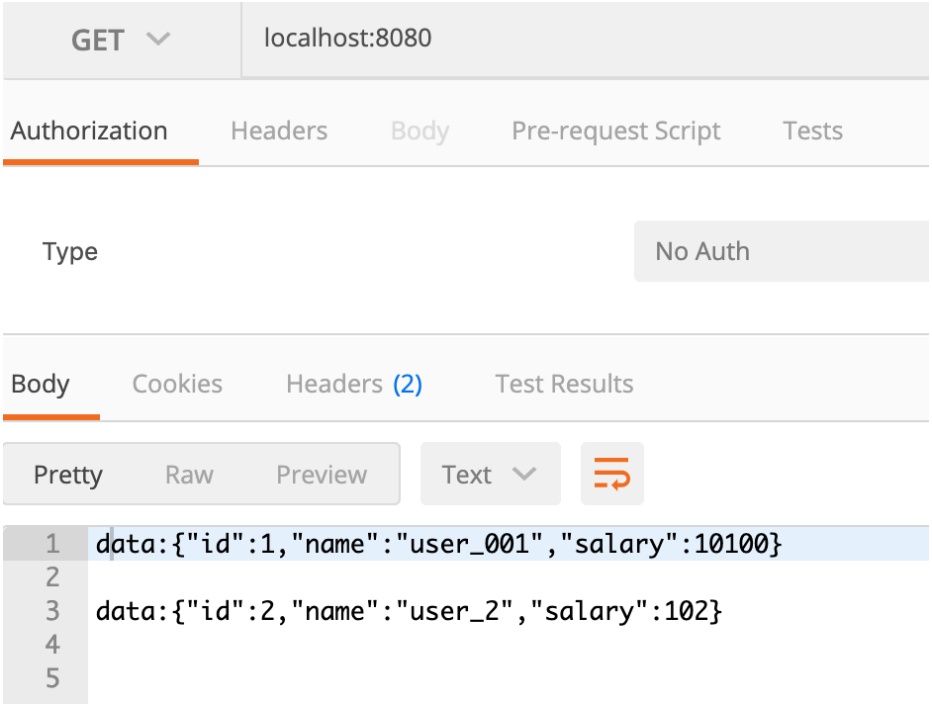
```
WebfluxFunctionalApp.java
     -> Right Click
          -> Run As
               -> Spring Boot App
```

```
Problems  @ Javadoc  Declaration  Console
<terminated> spring-webflux-demo - WebfluxFunctionalApp [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java (Oct 23, 2019, 11:59:32 PM)
2019-10-23 23:59:33.217  INFO 15195 --- [           main] com.demo.WebfluxFunctionalApp            : Starting WebfluxFunctionalApp on Nandakumars-MBP.home with PID 151
2019-10-23 23:59:33.219  INFO 15195 --- [           main] com.demo.WebfluxFunctionalApp            : No active profile set, falling back to default profiles: default
2019-10-23 23:59:33.474  INFO 15195 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2019-10-23 23:59:33.506  INFO 15195 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 30ms. Found 1 reposito
2019-10-23 23:59:33.796  INFO 15195 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017], mode=SINGL
2019-10-23 23:59:33.846  INFO 15195 --- [localhost:27017] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:1, serverValue:49}] to
2019-10-23 23:59:33.848  INFO 15195 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with description S
2019-10-23 23:59:33.965  INFO 15195 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017], mode=SINGL
2019-10-23 23:59:33.968  INFO 15195 --- [localhost:27017] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:2, serverValue:50}] to
2019-10-23 23:59:33.969  INFO 15195 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with description S
2019-10-23 23:59:34.229  INFO 15195 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017], mode=SINGL
2019-10-23 23:59:34.284  INFO 15195 --- [localhost:27017] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:3, serverValue:51}] to
2019-10-23 23:59:34.285  INFO 15195 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with description S
2019-10-23 23:59:34.334  INFO 15195 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017], mode=SINGL
2019-10-23 23:59:34.392  INFO 15195 --- [localhost:27017] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:4, serverValue:52}] to
2019-10-23 23:59:34.393  INFO 15195 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with description S
2019-10-23 23:59:34.421  INFO 15195 --- [           main] o.s.b.web.embedded.netty.NettyWebServer  : Netty started on port(s): 8080
```

17. OUTPUT on POSTMAN

Test all the APIs





PUT localhost:8080/update

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "id":1,
    "name":"user_001",
    "salary":10100
}
```

Body Cookies Headers (2) Test Results

Pretty Raw Preview JSON

```
{
    "id": 1,
    "name": "user_001",
    "salary": 10100
}
```

GET localhost:8080/2

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (2) Test Results

Pretty Raw Preview JSON

```
{
    "id": 2,
    "name": "user_2",
    "salary": 102
}
```

GET localhost:8080/name/user_001

Authorization Headers Body Pre-request Script

Type No Au

Body Cookies Headers (2) Test Results

Pretty Raw Preview JSON


```
[
    {
        "id": 1,
        "name": "user_001",
        "salary": 10100
    }
]
```