Date : 02-Jul-21
Spring Boot 9AM
Mr. RAGHU

---

https://github.com/javabyraghu/SpringBoot2ReactiveProducer
https://github.com/javabyraghu/SpringBoot2ReactiveConsumer

Spring Boot - Reactive (Consumer App)

*) RestTemplate(C) : To make HTTP calls but supports only normal web process
ie Blocking based.

*) WebClient(I) [org.springframework.web.reactive.function.client]
> It is a new API to make HTTP calls that supports reactive process.
> It follows builder API
> It support all HTTP methods and Header with Body.

> Spring Web Flux Dependency must be added (Spring Boot Reactive)
> Must specify model/entity at Consumer App
> Output Format is either Mono<T> or Flux<T>
(it is not ResponseEntity<T>)

==========(Consumer Application)==========
1. Define Producer URL
2. Create WebClient Object
3. Provide Request Information (method,body,header..etc)
4. Execute and covert response to Mono/Flux
5. print result

Name : SpringBoot2ReactiveConsumer
Dep : Spring Reactive Web, Lombok

*) Runner class

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

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;

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

@Component
public class ConsumerGetAllTestRunner implements CommandLineRunner {

	public void run(String... args) throws Exception {
		//1. Define Producer URL
		String url = "http://localhost:8081/student";

		//2. Create WebClient Object
		WebClient client = WebClient.create(url);

		//3. Provide Request Information (method, path , body,header..etc)
```

```
                //4. Execute and covert response to Mono/Flux
                Flux<Student> flux = client
                                .get()
                                .uri("/all")
                                .retrieve()
                                .bodyToFlux(Student.class);


                //5. print result
                flux.doOnNext(System.out::println);
        }

}

*) application.properties
server.port=8086

---------Execution Order----------------
1. Start MongoDB
cmd> mongod

2. Run Producer App
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;

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

@Component
public class ConsumerGetAllTestRunner implements CommandLineRunner {

        public void run(String... args) throws Exception {
                //1. Define Producer URL
                String url = "http://localhost:8081";

                //2. Create WebClient Object
                WebClient client = WebClient.create(url);

                //3. Provide Request Information (method, path ,
body,header..etc)
                //4. Execute and covert response to Mono/Flux
                Flux<Student> flux = client
                                .get()
                                .uri("/student/all")
                                .retrieve()
                                .bodyToFlux(Student.class);


                //5. print result
                flux.doOnNext(System.out::println).blockLast();

        }

}
```

```
====(For Delete)================================================
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;

import reactor.core.publisher.Mono;

@Component
public class ConsumerDeleteTestRunner implements CommandLineRunner {

	public void run(String... args) throws Exception {
		//1. Define Producer URL
		String url = "http://localhost:8081";

		//2. Create WebClient Object
		WebClient client = WebClient.create(url);

		//3. Provide Request Information (method, path ,
body,header..etc)
		//4. Execute and covert response to Mono/Flux
		Mono<Void> mono = client
					.delete()
					.uri("/student/remove/{id}",1102)
					.retrieve()
					.bodyToMono(Void.class);


		//5. print result
		mono.subscribe(System.out::println);
		System.out.println("OBJECT DELETED!!");
	}

}

============(For create)=========================================
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.web.reactive.function.client.WebClient;

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

//@Component
public class ConsumerPostTestRunner implements CommandLineRunner {

	public void run(String... args) throws Exception {
		//1. Define Producer URL
		String url = "http://localhost:8081";

		//2. Create WebClient Object
		WebClient client = WebClient.create(url);

		//3. Provide Request Information (method, path ,
body,header..etc)
```

```
                //4. Execute and covert response to Mono/Flux
                Mono<Student> mono = client
                                .post()
                                .uri("/student/create")
                                .body(Mono.just(new Student(1105,
"RRR", 300.0)), Student.class)
                                .retrieve()
                                .bodyToMono(Student.class);


                //5. print result
                mono.subscribe(System.out::println);
                System.out.println("OBJECT CREATED!!");
        }

}
=======(for get one)================================
package in.nareshit.raghu.runner;

import org.springframework.boot.CommandLineRunner;
import org.springframework.web.reactive.function.client.WebClient;

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

//@Component
public class ConsumerGetOneTestRunner implements CommandLineRunner {

        public void run(String... args) throws Exception {
                //1. Define Producer URL
                String url = "http://localhost:8081";

                //2. Create WebClient Object
                WebClient client = WebClient.create(url);

                //3. Provide Request Information (method, path ,
body,header..etc)
                //4. Execute and covert response to Mono/Flux
                Mono<Student> mono = client
                                .get()
                                .uri("/student/one/{id}",1102)
                                .retrieve()
                                .bodyToMono(Student.class);


                //5. print result
                mono.subscribe(System.out::println);

        }

}
============================================================
```