



# A functional endpoint in Spring WebFlux

- 1. Overview of Functional endpoint
- Home / Spring Framev 2. Functional reactive endpoints
  - 3. Test the endpoints using

WebTestClient

Conclusion

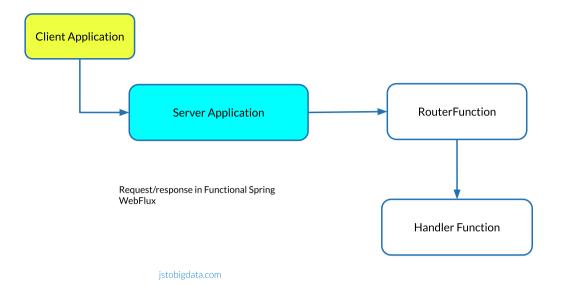
In this article, you will learn to create a Functional endpoint in Spring WebFlux. In the previous article, I have introduced you to create an endpoint using the Annotations, now we will look into the pure functional programming way to achieve this.

# 1. Overview of Functional endpoint

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- > Spring WebFlux Annotation based Controller

# Explore Abu Dhabi



An Http request initiated by a client app arrives at the Server (Netty/Undertow etc.). This request is forwarded to the **RouterFunction** to assign an appropriate **HandlerFunction** to serve the Http request. The RouterFunction is similar to @RequestMapping and the HandlerFunction is similar to the body of @RequestMapping method in the annotation-driven programming model.

# 2. Functional reactive endpoints

We will use the standard start.spring.io to generate the starter Spring-Boot-WebFlux project to Yimplement functional endpoints. The most important dependency that you need is spring-boot-

### Explore Abu Dhabi

Discover our amazing hotels, latest offers and flexible cancellation options.

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

#### WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller



### Step-1: Generate the project from start.spring.io

We will generate our Spring WebFlux project from **start.spring.io**, with the dependency spring-boot-starter-webflux. Other dependencies are for testing and ease of development. Use the link to generate the project, unzip it, and import it to your IDE.

```
<parent>
1.
       <groupId>org.springframework.boot
2.
       <artifactId>spring-boot-starter-parent</artifactId>
3.
       <version>2.3.0.RELEASE/version>
4.
       <relativePath/> <!-- lookup parent from repository -->
5.
     </parent>
6.
7. . . . . .
     <dependencies>
8.
       <dependency>
9.
         <groupId>org.springframework.boot
10.
         <artifactId>spring-boot-starter-webflux</artifactId>
11.
       </dependency>
12.
     <dependencies>
13.
14. ....
```

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

	Portronics My Buddy K Laptop Stand (POR 421, Grey)	Croma Wireless Mouse (XM5106, Black)	Portronics RuffPad POR- 628 21.59 cm (8.5 Inch) Portable E-Writer, Black	Fastrack 25 Litres I Backpack for 16' (Back Padding, / Gree	Table of Contents
	<b>₹1,140</b> -42%	₹ <b>599</b> -40%	<b>₹420</b> -53%	₹779	<ol> <li>Overview of Functional endpoint</li> <li>Functional reactive endpoints</li> <li>Test the endpoints using</li> <li>WebTestClient</li> <li>Conclusion</li> </ol>
<b>◯</b> Step-2: 0	Create the handler function	ons in HelloHandler			Spring WebFlux Tutorial

**Portronics My Buddy** K Laptop Stand (POR

₹1.999

421, Grey)

Portronics RuffPad POR-628 21.59 cm (8.5 Inch) Portable E-Writer, Black

**₹899** -53%

Croma Wireless Mouse (XM5106, Black)

₹1.000

**Spring WebFlux Tutorial** 

Fastrack 25 Litres F

₹1,425

Backpack for 16

(Back Padding, A

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- > Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

as HelloHandler with two methods. One will serve us the TEXT\_PLAIN and other one to serve APPLICATION STREAM JSON content.

### HelloHandler.java FunctionalAppConfig.java

```
    package c.jbd.webflux;

   2.
   3. import org.springframework.http.MediaType;
   4. import org.springframework.stereotype.Component;
     import org.springframework.web.reactive.function.server.ServerRequest;
   6. import org.springframework.web.reactive.function.server.ServerResponse;
   7. import reactor.core.publisher.Flux;
   8. import reactor.core.publisher.Mono;
   9.
      import java.time.Duration;
   11.
   12. @Component
  13. public class HelloHandler {
   14.
        /**
         * Serves a plain_text
   15.
         */
   16.
        public Mono<ServerResponse> monoMessage(ServerRequest request) {
   17.
          return ServerResponse.ok()
   18.
            .contentType(MediaType.TEXT_PLAIN)
   19.
            .bodv(
   20.
              Mono.just("Welcome to JstoBigdata.com"), String.class
   21.
            );
   22.
23.
```

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

#### WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

```
public Mono<ServerResponse> fluxMessage(ServerRequest request) {
28.
       return ServerResponse.ok()
29.
         .contentType(MediaType.APPLICATION STREAM JSON)
30.
         .body(
31.
           Flux.just("Welcome ", "to ", "JstoBigdata.com")
32.
              .delayElements(Duration.ofSeconds(1)).log(), String.class
33.
         );
34.
35.
36. }
```

		WeblestClient			
	Portronics My Buddy K Laptop Stand (POR 421, Grey)	Wonderchef Large Tadka Pan for Stoves & Cooktops (Environment-Friendly, 50001900, Black)	Zebronics ZEB-NS2000 Laptop Stand (7 Adjustable Levels, A31-NS2000, Dark Grey)	Portronics RuffPa 628 21.59 cn Portable E-W	Conclusion
	<b>₹1,140</b> -42%	₹299	₹899 -55%	₹420	Spring WebFlux Tutorial
					> Project Reactor - Introduction
					> Project Reactor – Mono
					> Project Reactor - Flux
					<ul><li>&gt; Project Reactor – Transform and combine</li></ul>
<b>▼</b> Step-	-3: Create the Router functio	> Project Reactor – Backpressure			



We have the logic in the HelloHandler to serve the response. Now, we will write our router function in HelloRouter class to bind the specific endpoints. Basically, every endpoint will be

Yinked to the enecific router function

## Explore Abu Dhabi

Discover our amazing hotels, latest offers and flexible cancellation options.



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

### WehTestClient

> Spring WebFlux - Annotation based Controller

```
    package c.jbd.webflux;

2.
3. import org.springframework.context.annotation.Bean;
4. import org.springframework.context.annotation.Configuration;
5. import org.springframework.web.reactive.function.server.RequestPredicates;
6. import org.springframework.web.reactive.function.server.RouterFunction;
7. import org.springframework.web.reactive.function.server.RouterFunctions;
8. import org.springframework.web.reactive.function.server.ServerResponse;
9.
10. @Configuration
11. public class HelloRouter {
     @Bean
12.
     public RouterFunction<ServerResponse> functionalRoutes(HelloHandler
   helloHandler) {
       return RouterFunctions
14.
         .route(RequestPredicates.GET("/functional/mono")
15.
            , helloHandler::monoMessage)
16.
         .andRoute(RequestPredicates.GET("functional/flux")
17.
           , helloHandler::fluxMessage);
18.
19.
20. }
```

It is recommended to declare the **Router Function class** using **@Configurations** annotations, and the router functions as beans.



### Step-4: Start the App and Test the endpoints

### Explore Abu Dhabi

Discover our amazing hotels, latest offers and flexible cancellation options.

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

#### WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

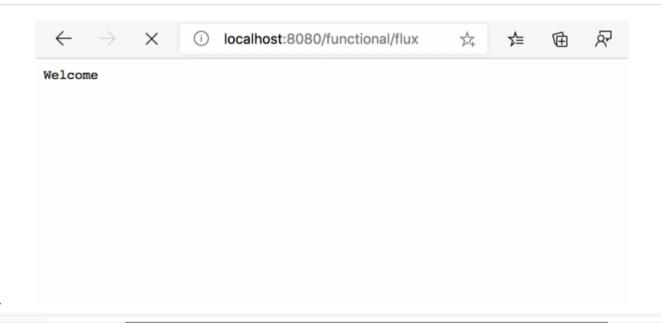
To test the Mono endpoint, open the below url in the browser.

http://localhost:8080/functional/mono

You will be able to get the message, Welcome to JstoBigdata.com on the browser. There is nothing special here.

Similarly, to test the **Flux endpoint**, open the below url in browser and observe the output.

http://localhost:8080/functional/flux



#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

# 3. Test the endpoints using WebTestClient

The spring-test module provides mock implementations of ServerHttpRequest,

ServerHttpResponse, and ServerWebExchange. The **WebTestClient** is built on these mock request and response objects to provide support for testing WebFlux applications without an HTTP server. You can use the WebTestClient for end-to-end integration tests, too.

```
1. package c.jbd.webflux;
2.
3. import org.junit.jupiter.api.Test;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import
   org.springframework.boot.test.autoconfigure.web.reactive.AutoConfigureWebTe
   stClient:
6. import org.springframework.boot.test.context.SpringBootTest;
7. import org.springframework.http.MediaType;
8. import org.springframework.test.web.reactive.server.WebTestClient;
9. import reactor.core.publisher.Flux;
10. import reactor.test.StepVerifier;
11.
12. @SpringBootTest
13. @AutoConfigureWebTestClient //Important
14. public class FunctionalAppTest {
     @Autowired
     private WebTestClient webTestClient;
```

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using WebTestClient
  Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

```
public void testMonoEndpoint() {
21.
       Flux<String> msg$ = webTestClient.get()
22.
          .uri("/functional/mono")
23.
          .accept(MediaType.TEXT_PLAIN)
24.
25.
          .exchange()
          .expectStatus().is0k()
26.
          .returnResult(String.class).getResponseBody()
27.
          .log();
28.
29.
       StepVerifier.create(msg$)
30.
          .expectNext(TEST MESSAGE)
31.
          .verifyComplete();
32.
33.
34.
     @Test
35.
     public void testFluxEndpoint() {
36.
       Flux<String> msg$ = webTestClient.get()
37.
          .uri("/functional/flux")
38.
          .accept(MediaType.APPLICATION_STREAM_JSON)
39.
          .exchange()
40.
          .expectStatus().is0k()
41.
          .returnResult(String.class).getResponseBody()
42.
          .log();
43.
44.
       StepVerifier.create(msg$)
45.
          .expectNext(TEST_MESSAGE)
46.
          .verifyComplete();
47.
48.
49. }
```

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

### WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

### **Conclusion**

I have given you a brief guide on creating functional endpoint in Spring WebFlux. We will explore much more in the upcoming articles. Reactive programming provides better performance with limited hardware as compared to the traditional Spring MVC with too many threads. I would suggest trying this framework to see if this fulfils your project needs. You can download the complete source code from Github.

□ CODE EXAMPLE

By Bikram Kundu | July 10th, 2020 | Categories: Spring Framework | Tags: Spring 5.x, Spring WebFlux

Share This Page, Choose Your Platform!















#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

# Explore Abu Dh

Discover our amazing hotels, offers and flexible cancellation



### **Leave A Comment**

Name (required)	Email (required)	Website	
			//
Comment			
C			

#### **Table of Contents**



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using

WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi

Table of Contanta

This site uses Akismet to reduce spam. Learn how your comment data is processed.

ABOUT JSTOBIGDATA

> The API Gateway Pattern in Microservices
> Introduction to Microservices Architecture

> Getting Started with JUnit 5

> Spring @Import and @ImportResource annotations

> Handle Resources in Spring

ABOUT JSTOBIGDATA

> About us

> Contact us

> Privacy Policy

> Terms & Conditions

> Sitemap

## Explore Abu Dhabi

Discover our amazing hotels, latest offers and flexible cancellation options.

> JUnit 5 - Software Testing Framework

and Spring Data

> Spring WebFlux REST Api with MongoDB

> A functional endpoint in Spring WebFlux

> Use of @Order annotation in Spring

© Copyright 2022 | All Rights Reserved | Created by Bikram Kundu



- 1. Overview of Functional endpoint
- 2. Functional reactive endpoints
- 3. Test the endpoints using WebTestClient

Conclusion

### **Spring WebFlux Tutorial**

- > Project Reactor Introduction
- > Project Reactor Mono
- > Project Reactor Flux
- > Project Reactor Transform and combine
- > Project Reactor Backpressure
- Spring WebFlux Annotation based Controller

# Explore Abu Dhabi