Fizz Buzz API documentation

Kalmár Gábor

1. Introduction

This document describes how I created a FizzBuzz web service in Eclipse using JAX-RS and WildFly.

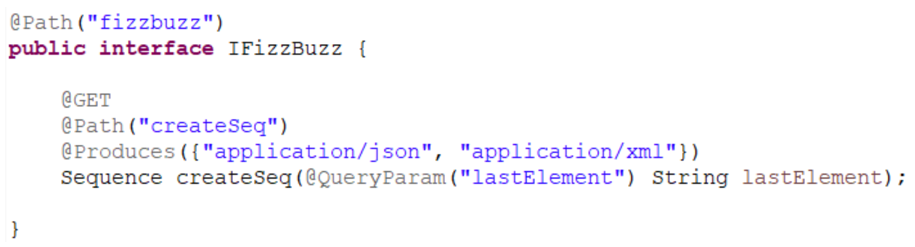
2. Requirements

- Eclipse IDE: 2021-12 R
- JDK, version: Temurin 11 (LTS)
- Maven, version: 3.8.4
- WildFly Jakarta EE Full & Web Distribution: 26.0.1.Final
- JBoss Tools is required for Eclipse so that it can use the WildFly server
- TestNG, version: 7.4.0

3. RESTful Service

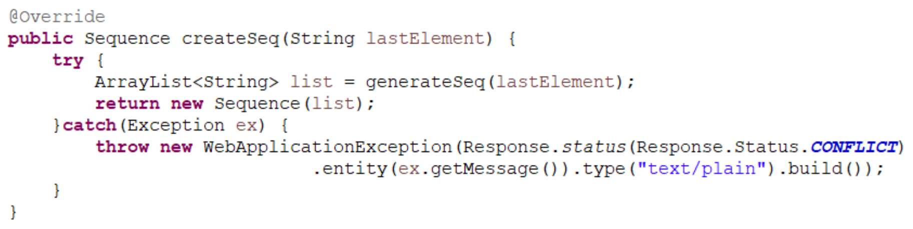
I have created a new Maven Project in Eclipse called FizzBuzz using the maven-archetype-webapp archetype. The JAX-RS annotated classes will be handled by the javax.ws.rs.core.Application servlet.

I have created a new interface called IFizzBuzz and a class called FizzBuzz in the src/main/java folder under the package FizzBuzz. I annotated the only endpoint (createSeq) with the required informations. The function can produce JSON and XML responses, based on the content-type in the request header.



```
@Path("fizzbuzz")
public interface IFizzBuzz {

    @GET
    @Path("createSeq")
    @Produces({"application/json", "application/xml"})
    Sequence createSeq(@QueryParam("lastElement") String lastElement);

}
```

The function accepts a string parameter as input, therefore it can provide valuable information about the input validity The function returns a Sequence wrapper object.



```
@Override
public Sequence createSeq(String lastElement) {
    try {
        ArrayList<String> list = generateSeq(lastElement);
        return new Sequence(list);
    }catch(Exception ex) {
        throw new WebApplicationException(Response.status(Response.Status.CONFLICT)
                        .entity(ex.getMessage()).type("text/plain").build());
    }
}
```

generateSeq function parses the input to integer. If the input cannot be parsed to integer or it is a negative number, the function throws an exception with the specific error message. Otherwise it generates the Fizz Buzz sequence and returns the ArrayList of strings.

```java
public ArrayList<String> generateSeq(String lastElement) throws Exception {

    int lastInt = 0;

    try {
        lastInt = Integer.parseInt(lastElement);
    }catch(NumberFormatException ex) {
        throw new Exception("Unable to parse invalid parameter");
    }

    if(lastInt < 0)
        throw new Exception("Given number is less then 0");

    ArrayList<String> list = new ArrayList<>();
    for(int i = 1; i <= lastInt; i++) {
        if((i%3 == 0) && (i%5 == 0)) {
            list.add("Fizz Buzz");
        }else if(i%5 == 0){
            list.add("Buzz");
        }else if(i%3 == 0){
            list.add("Fizz");
        }else {
            list.add(Integer.toString(i));
        }
    }
    return list;
}
```

The Sequence is a simple wrapper class that is necessary for JSON and XML parsing.

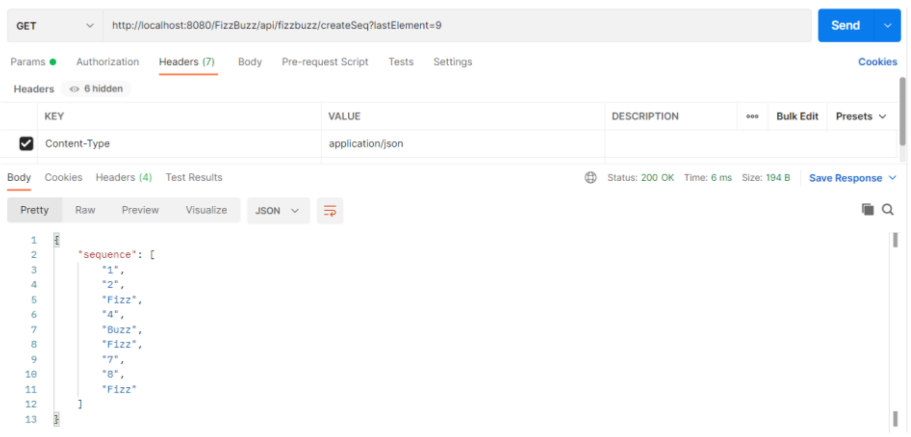
```java
@XmlRootElement(name = "sequence")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType
public class Sequence {

    @XmlElement(name = "sequence", type = String.class)
    private List<String> sequence;

    public Sequence() {
        sequence = new ArrayList<>();
    }

    public Sequence(List<String> s) {
        this.sequence = new ArrayList<>(s);
    }

    public List<String> getSequence() {
        return sequence;
    }

    public void setSequence(List<String> s) {
        this.sequence = s;
    }
}
```

4. Run

To configured WildFly under Eclipse, first open the Servers tab at the bottom of the Eclipse window. Click on the link to create a new server, then select WildFly 24+. Click on the Browse... button at the Home Directory textbox and locate the WildFly installation directory, switch to the Alternate JRE and select the installed JDK. Right click on the server and start it. The server should start without errors, the logs can be seen in the Console window. To start the API, first right click on the project, click on the Run As button, select Run on Server, finally select the started WildFly server.

GET	http://localhost:8080/FizzBuzz/api/fizzbuzz/createSeq?lastElement=9
Request HTTP body:	

Response HTTP body:	{
	    "sequence": [
	        "1",
	        "2",
	        "Fizz",
	        "4",
	        "Buzz",
	        "Fizz",
	        "7",
	        "8",
	        "Fizz"
	    ]
	}



http://localhost:8080/FizzBuzz/api/fizzbuzz/createSeq?lastElement=9

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	

Status: 200 OK Time: 6 ms Size: 194 B

```
{
    "sequence": [
        "1",
        "2",
        "Fizz",
        "4",
        "Buzz",
        "Fizz",
        "7",
        "8",
        "Fizz"
    ]
}
```

5. Testing

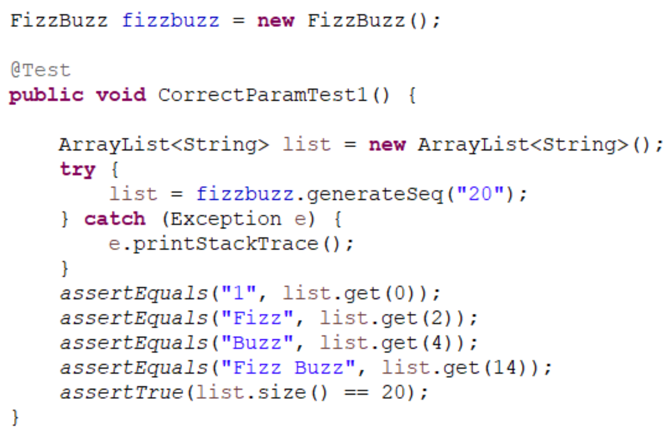
I both implemented UNIT and INTEGRATION test cases. I have used JUnit and TestNG for the implementation.

5.1. Unit Test

In order to use JUnit I added the required dependencies to the pom.xml file.

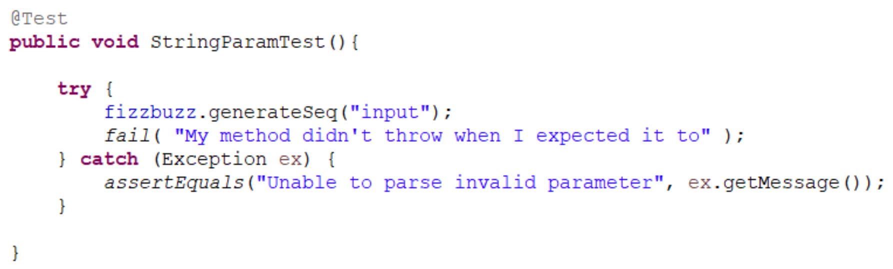
```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.glassfish.jersey.core</groupId>
        <artifactId>jersey-common</artifactId>
        <version>2.22.2</version>
        <scope>test</scope>
</dependency>
```

I have created a class called FizzBuzzTest under the scr/test/java folder. I implemented the following three test cases. I have tested the generateSeq function with a positive number (expected input). I have compared some expected values with the result values of the tested function.



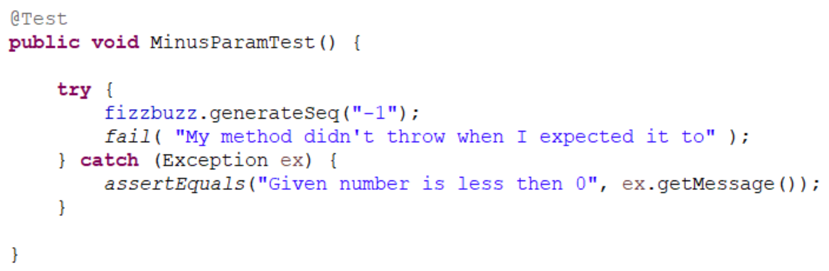
```
FizzBuzz fizzbuzz = new FizzBuzz();

@Test
public void CorrectParamTest1() {

    ArrayList<String> list = new ArrayList<String>();
    try {
        list = fizzbuzz.generateSeq("20");
    } catch (Exception e) {
        e.printStackTrace();
    }
    assertEquals("1", list.get(0));
    assertEquals("Fizz", list.get(2));
    assertEquals("Buzz", list.get(4));
    assertEquals("Fizz Buzz", list.get(14));
    assertTrue(list.size() == 20);
}
```

I have tested the generateSeq function with a random (non number) string. I expect to throw an exception and I compared the error message with the expected error message.



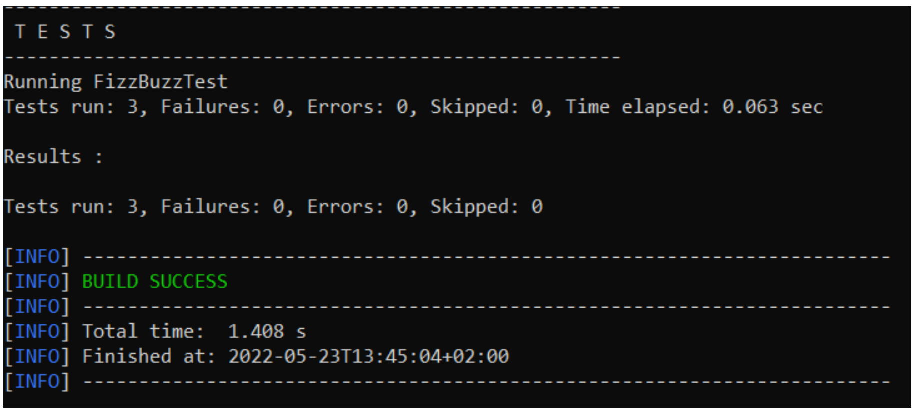
```
@Test
public void StringParamTest(){

    try {
        fizzbuzz.generateSeq("input");
        fail( "My method didn't throw when I expected it to" );
    } catch (Exception ex) {
        assertEquals("Unable to parse invalid parameter", ex.getMessage());
    }

}
```

I have tested the generateSeq function with a negative number. I expect to throw an exception and I compared the error message with the expected error message.



```
@Test
public void MinusParamTest() {

    try {
        fizzbuzz.generateSeq("-1");
        fail( "My method didn't throw when I expected it to" );
    } catch (Exception ex) {
        assertEquals("Given number is less then 0", ex.getMessage());
    }

}
```

To execute the test cases I had to run „mvn test" command in the project folder.



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running FizzBuzzTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.063 sec

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.408 s
[INFO] Finished at: 2022-05-23T13:45:04+02:00
[INFO] ------------------------------------------------------------------------
```

5.2. Integration Test

I have used Rest-assured and TestNG for integration testing. I created a new maven project called FizzBuzzApiTest. I added the required dependencies to the pom.xml file.

```
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.0.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.6.0</version>
    <scope>test</scope>
</dependency>
```

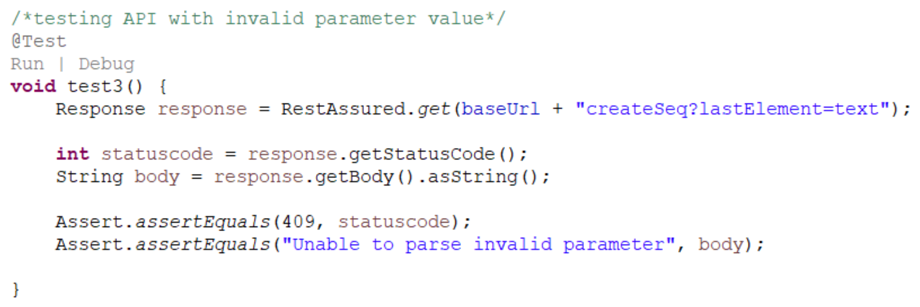
I created a class called TestGET under the scr/test/java folder. I have tested the API communication with the following three test cases. I annotated the test cases with the @Test annotation for TestNG. I have tested the API with a positive number as parameter. I have compared the response status code and some expected values with the response values.

```
String baseUrl = "http://localhost:8080/FizzBuzz/api/fizzbuzz/";

@Test
Run | Debug
void test1() {
    Response response = RestAssured.get(baseUrl + "createSeq?lastElement=16");

    System.out.println(response.getStatusCode());
    System.out.println(response.getBody().asString());

    int statuscode = response.getStatusCode();

    Assert.assertEquals(200, statuscode);

    RestAssured.get(baseUrl + "createSeq?lastElement=16").then()
        .statusCode(200)
        .body("sequence[0]", equalTo("1"))
        .body("sequence[2]", equalTo("Fizz"))
        .body("sequence[4]", equalTo("Buzz"))
        .body("sequence[14]", equalTo("Fizz Buzz"));
}
```

I have tested the API with a negative number as parameter. I expected to get a 409 error code. I compared the error message with the expected error message.

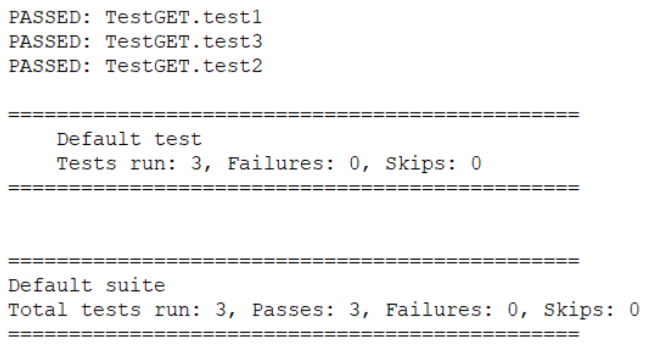
```
/*testing API with negative number*/
@Test
Run | Debug
void test2() {
    Response response = RestAssured.get(baseUrl + "createSeq?lastElement=-2");

    int statuscode = response.getStatusCode();
    String body = response.getBody().asString();

    Assert.assertEquals(409, statuscode);
    Assert.assertEquals("Given number is less then 0", body);

}
```

I have tested the API with a random (non number) string as parameter. I expected to get a 409 error code. I compared the error message with the expected error message.



```
/*testing API with invalid parameter value*/
@Test
Run | Debug
void test3() {
    Response response = RestAssured.get(baseUrl + "createSeq?lastElement=text");

    int statuscode = response.getStatusCode();
    String body = response.getBody().asString();

    Assert.assertEquals(409, statuscode);
    Assert.assertEquals("Unable to parse invalid parameter", body);

}
```

To run the tests, I had to right click on the project, click on the Run As button, then select TestNG Test.



```
PASSED: TestGET.test1
PASSED: TestGET.test3
PASSED: TestGET.test2

===============================================
    Default test
    Tests run: 3, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
===============================================
```