



2.3 – Pruebas de Servicios de Internet

# Tema 3 - Pruebas de API REST



Universidad  
Rey Juan Carlos

**Micael Gallego**

micael.gallego@urjc.es  
@micael\_gallego

**Michel Maes**

michel.maes@urjc.es





## Tema 3 - Pruebas de API REST

# Tema 3.2 – Pruebas con REST Assured (Spring y Quarkus)



Universidad  
Rey Juan Carlos

**Micael Gallego**

micael.gallego@urjc.es  
@micael\_gallego

**Michel Maes**

michel.maes@urjc.es



- Introducción a Rest Assured
- Pruebas en Spring con Rest Assured
- Pruebas en Quarkus con Rest Assured

- **Introducción a Rest Assured**
- Pruebas en Spring con Rest Assured
- Pruebas en Quarkus con Rest Assured

## REST-assured

- **REST Assured** es una librería que facilita la implementación de **pruebas automáticas** funcionales de sistema de **APIs REST**
- Estos tests se pueden implementar **usando clientes REST**, pero la librería facilita la tarea y los **tests son más consisos y fáciles de entender**

<http://rest-assured.io/>

- REST Assured se usa en combinación con JUnit

## API REST

`http://localhost:8080/lotto/{id}`

```
{
  "lotto": {
    "lottoId": 5,
    "winning-numbers": [2, 45, 34],
    "winners": [
      {
        "winnerId": 23,
        "numbers": [2, 45]
      },
      {
        "winnerId": 54,
        "numbers": [45, 34]
      }
    ]
  }
}
```

## Test

```
@Test
public void test() {

    when().
        get("/lotto/{id}", 5).
    then().
        statusCode(200).
        body(
            "lotto.lottoId", equalTo(5),
            "lotto.winners.winnerId", containsOnly(23, 54)
        );
}
```

- ## Dependencias Maven

```
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>${io-rest-assured.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>json-path</artifactId>
  <version>${io-rest-assured.version}</version>
</dependency>
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>xml-path</artifactId>
  <version>${io-rest-assured.version}</version>
</dependency>
<dependency>
  <groupId>org.hamcrest</groupId>
  <artifactId>hamcrest-all</artifactId>
  <version>1.3</version>
</dependency>
```

- Esquema básico de un test

```
import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

...

@Test
public void test(){

    given().
        params("firstName", "John", "lastName", "Doe").
    when().
        post("/greet").
    then().
        statusCode(200).
        body("greeting.firstName", equalTo("John")).
        body("greeting.lastName", equalTo("Doe"));
}
```



- Configuración de la petición en given()
- Parámetros

```
given().  
    params("firstName", "John", "lastName", "Doe").  
  
    param("param1", "value1").  
  
    //Parámetro sin valor  
    param("param3").  
  
    //Parámetro con múltiples valores  
    param("param2", "value1", "value2", "value3").
```

```
//Parámetro con múltiples valores con List<String>  
  
List<String> values = ...  
given().param("myList", values).
```

- Configuración de la petición en given()
- Tipos de parámetros
  - El tipo de parámetro se determina en base al método de la petición GET o POST. Pero se puede ser explícito e indicar el tipo de parámetro

```
given().  
    formParam("formParamName", "value1").  
    queryParam("queryParamName", "value2").
```

- Configuración de la petición en given()
- Parámetros en la ruta

```
given().  
    pathParam("hotelId", "My Hotel").  
    pathParam("roomNumber", 23).  
when().  
    post("/reserve/{hotelId}/{roomNumber}").
```

```
given().  
    pathParam("hotelId", "My Hotel").  
when().  
    post("/reserve/{hotelId}/{roomNumber}", 23).
```

```
given().  
when().  
    post("/reserve/{hotelId}/{roomNumber}", "MyHot", 23);
```

- Configuración de la petición en given()
- Cabeceras (Headers)

```
given().  
    header("MyHeader", "Something").
```

```
given().  
    header("MyHeader", "Value1", "Value2").
```

```
given().  
    headers("MyHeader", "Value1", "OtherHeader", "Value2")
```

- Configuración de la petición en `given()`

- `ContentType`

```
given().  
    contentType(ContentType.TEXT).
```

```
given().  
    contentType("application/json").
```

- Request Body

```
given().  
    body("some body")
```

```
given().  
    request().body("some body")
```

```
given().  
    body(new byte[]{42})
```

- Método en when()

```
given().  
    parameters(...).  
    contentType(...).  
when().  
    get("/greeting").  
then().
```

```
when().  
    post("/greeting").
```

```
when().  
    put("/greeting").
```

```
when().  
    delete("/greeting").
```

- **Verificar la respuesta en then()**

- Status code

```
then().statusCode(400).
```

```
then().assertThat().statusCode(400).
```

```
then().assertThat().statusLine("something").
```

```
then().assertThat().statusLine(containsString("some")).
```

- Headers

```
then().header("headerName", "headerValue").
```

```
then().assertThat().headers("header1", "v1", "header2", "v2").
```

```
then().assertThat().header("headerName", containsString("v2")).
```

- Verificar la respuesta en then()
  - ContentType

```
then().assertThat().contentType(ContentType.JSON).
```

```
then().assertThat().contentType("application/json").
```

- Body

```
then().assertThat().body(equalTo("something")).
```



- Verificar la respuesta en then()
  - JSON Body

```
{
  "lotto":{
    "lottoId":5,
    "winners":[
      { "winnerId":23, "numbers":[2,45] },
      { "winnerId":54, "numbers":[45,34] }
    ]
  }
}
```

```
then()
  .body("lotto.lottoId", equalTo(5));
```

```
then()
  .body("lotto.winners.winnerId", hasItems(23, 54));
```

Gpath para seleccionar partes del JSON

- Verificar la respuesta en then()
  - JSON Body

```
then().  
    body("lotto.lottoId", equalTo(5));
```

```
then().  
    body("lotto.winners.winnerId", hasItems(23, 54));
```

Seleccionar partes  
del JSON  
**GPath Groovy**

Verificar contenido  
**Hamcrest Matchers**

[http://groovy-lang.org/processing-xml.html#\\_gpath](http://groovy-lang.org/processing-xml.html#_gpath)

<http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/Matchers.html>

- Verificar la respuesta en then()
  - JSON Body
    - Búsquedas avanzadas usando sintaxis Groovy

```
{
  "store":{
    "book":[
      {
        "author":"Nigel Rees",
        "category":"reference",
        "price":8.95,
        "title":"Sayings of the Century"
      },
      {
        "author":"Evelyn Waugh",
        "category":"fiction",
        "price":12.99,
        "title":"Sword of Honour"
      },
      {
        "author":"Herman Melville",
        "category":"fiction",
        "isbn":"0-553-21311-3",
        "price":8.99,
        "title":"Moby Dick"
      }
    ]
  }
}
```

```
then().body("store.book.findAll { it.price < 10 }.title",
  hasItems("Sayings of the Century", "Moby Dick"));
```

```
then().body("store.book.author.collect { it.length() }.sum()",
  greaterThan(50));
```

- Verificar la respuesta en then()
  - JSON Body
    - Podemos usar las expresiones Groovy obtener los valores y manejarlos como queramos

```
import static io.restassured.path.json.JsonPath.from;
...

// Get the response body as a String
String response = get("/store").asString();

List<String> bookTitles = from(response).getList(
    "store.book.findAll { it.price < 10 }.title");

int sum = from(response).getInt("store.book.author*.length().sum()");

// We can also assert that the sum is equal to 53 as expected.
assertThat(sum, is(53));
```

- **Obtener datos de una petición**
  - Ideal para encadenar peticiones en los tests

```
import static io.restassured.path.json.JsonPath.*;
import static io.restassured.RestAssured.*;
....

Response response = given().
    body("{ ... }").
    contentType(ContentType.JSON).
    post("/items/").andReturn();

Integer id = from(response.getBody().asString()).get("id");

//Use id variable in new requests
```

- **Opciones avanzadas**

- Procesamiento de respuestas en XML
- Verificación de que la respuesta se ajusta a un esquema JSON o XML
- Gestión de Cookies
- Medir el tiempo de la respuesta
- Verificar que la respuesta tiene el mismo valor en dos partes
- Autenticación
- Conversión de body a objetos Java

<https://github.com/rest-assured/rest-assured/wiki/Usage#specifying-request-data>

<http://www.hascode.com/2011/10/testing-restful-web-services-made-easy-using-the-rest-assured-framework/>

<https://github.com/basdijkstra/workshops/tree/master/rest-assured>

- Ejemplo testeando la API de GoogleBooks

books-rest-test-ejem1

```
@Test
public void test() {
    given().
        param("q", "intitle:javascript").
    when().
        get("https://www.googleapis.com/books/v1/volumes").
    then().
        statusCode(200).
        body(
            "items.volumeInfo.title",
            hasItems(containsString("Javascript")))
    );
}
```

- Introducción a Rest Assured
- **Pruebas en Spring con Rest Assured**
- Pruebas en Quarkus con Rest Assured



# Pruebas en Spring con Rest Assured

- Ejemplo testeando la API de Items

spring-rest-test-ejem2

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)  
public class RestAPITest {
```

```
    @LocalServerPort  
    int port;
```

```
    @BeforeEach  
    public void setUp() {  
        RestAssured.port = port;  
    }
```

```
    @Test  
    public void itemAddTest() {
```

```
        given().  
            contentType("application/json").  
            body("{\"description\":\"milk\",\"checked\":false }").  
        when().  
            post("/items/").  
        then().  
            statusCode(201).  
            body("description", equalTo("milk")).  
            body("checked", equalTo(false));
```

```
    }
```

```
}
```

Lanza la aplicación  
para las pruebas de  
esta clase en un  
puerto aleatorio

# Pruebas en Spring con Rest Assured

- Ejemplo testeando la API de Items spring-rest-test-ejem2

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)  
public class RestAPITest {
```

```
    @LocalServerPort  
    int port;
```

```
    @BeforeEach  
    public void setUp() {  
        RestAssured.port = port;  
    }
```

```
    @Test  
    public void itemAddTest() {  
  
        given().  
            contentType("application/json").  
            body("{\"description\":\"milk\",\"checked\":false }").  
        when().  
            post("/items/").  
        then().  
            statusCode(201).  
            body("description", equalTo("milk")).  
            body("checked", equalTo(false));  
    }
```

Con **@LocalServerPort**  
inyectamos el puerto  
aleatorio en una variable  
con la que configuramos  
RestAssured

```
}
```

- **Ejercicio 1: Tests de API REST de Items**
  - Implementa los tests restantes con REST Assured de la API de items
    - Recuperar un item
    - Recuperar todos los items
    - Borrar un item

- Introducción a Rest Assured
- Pruebas en Spring con Rest Assured
- **Pruebas en Quarkus con Rest Assured**

# Pruebas en Spring con Rest Assured

## quarkus-rest-test\_ejem1

```
@QuarkusTest
@TestHTTPEndpoint(PostResource.class)
public class PostResourceTest {

    @Test
    public void testCreatePost() {

        String body = "{\"user\":\"Michel\",\"title\":\"Vendo Opel
Corsa\",\"text\":\"Bueno, bonito y barato\"}";

        given().
            contentType("application/json").
            body(body).
        when().
            post().
        then().
            statusCode(201).
            body("user", equalTo("Michel")).
            body("title", equalTo("Vendo Opel Corsa")).
            body("text", equalTo("Bueno, bonito y barato"));
    }
}
```

Lanza el controlador en  
el contexto del test

No es necesario indicar el path  
de la clase

```
@Path("/posts")
public class PostResource {
```