

Blog sobre Java EE

Estás aquí: [Inicio](#) / [Java EE](#) / [REST](#) / Spring REST Test utilizando Rest Assured

Spring REST Test utilizando Rest Assured

1 febrero, 2018 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)

[JAVA SE](#)[JAVA SE +](#)[SPRING](#)[JAVA EE](#)[JAVASCRIPT](#)[FRAMEWORKS JS](#)[ARQUITECTURA](#)[MIS LIBROS](#)

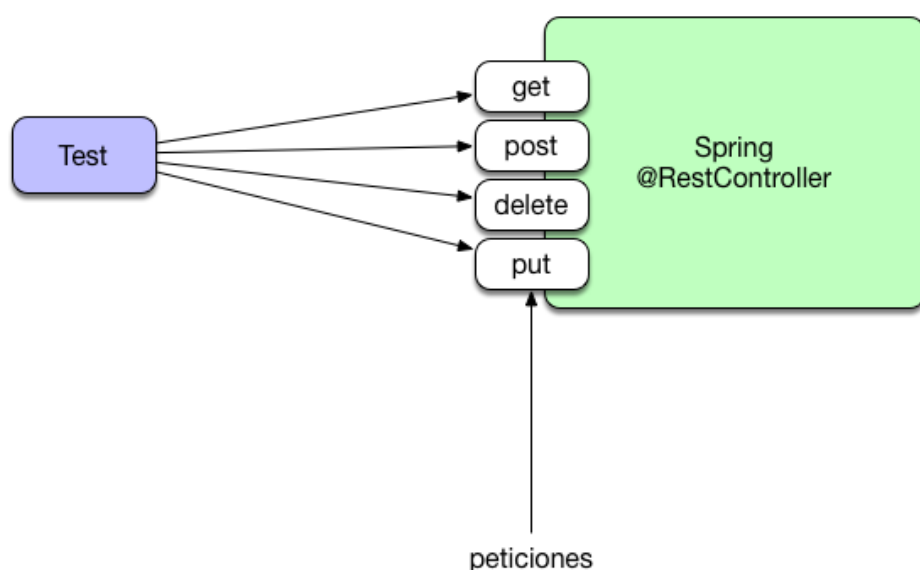


**G Suite de Google Cloud**

Abra y edite documentos rápidamente desde cualquier lugar: el taxi, el tren o el avión.



La necesidad de crear **Spring REST Test** cada día aumenta. Nuestras arquitecturas avanzan el uso de servicios REST se multiplica. Tenemos que comenzar a construir pruebas unitarias que de una forma sencilla puedan testear servicios REST . Ya sean servicios existentes o Mocks puros.



Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[ACEPTAR](#)[plugin cookies](#)

para llevar a cabo esta tarea. A continuacion se muestra el fichero Maven de partida.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>com.arquitecturajava</groupId>
6   <artifactId>rest</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>rest</name>
11  <description>Demo project for Spring Boot</description>
12
13  <parent>
14    <groupId>org.springframework.boot</groupId>
15    <artifactId>spring-boot-starter-parent</artifactId>
16    <version>1.5.10.RELEASE</version>
17    <relativePath/> <!-- lookup parent from repository -->
18  </parent>
19
20  <properties>
21    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEn
23    <java.version>1.8</java.version>
24  </properties>
25
26  <dependencies>
27    <dependency>
28      <groupId>org.springframework.boot</groupId>
29      <artifactId>spring-boot-starter-web</artifactId>
30    </dependency>
31
32    <dependency>
33      <groupId>org.springframework.boot</groupId>
34      <artifactId>spring-boot-starter-test</artifactId>
35      <scope>test</scope>
36    </dependency>
37
38    <dependency>
39      <groupId>io.rest-assured</groupId>
40      <artifactId>rest-assured</artifactId>
41      <version>3.0.6</version>
42      <scope>test</scope>
43    </dependency>
44  </dependencies>
```

Clases y Servicios

Acabamos de definir las dependencias para SpringBoot. Es momento de ver nuestras clases Java y como trabajar con ellas. En este caso únicamente generaremos la clase Ordenador y un Servicio REST que nos devuelva información sobre ella.

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra política de cookies, pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR

```
10 public Ordenador(String modelo, String marca, double precio) {
11     super();
12     this.modelo = modelo;
13     this.marca = marca;
14     this.precio = precio;
15 }
16 public Ordenador() {
17     super();
18 }
19 public String getModelo() {
20     return modelo;
21 }
22 public void setModelo(String modelo) {
23     this.modelo = modelo;
24 }
25 public String getMarca() {
26     return marca;
27 }
28 public void setMarca(String marca) {
29     this.marca = marca;
30 }
31 public double getPrecio() {
32     return precio;
33 }
34 public void setPrecio(double precio) {
35     this.precio = precio;
36 }
37
38 }
```

```
1 package com.arquitecturajava.rest;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import org.springframework.http.HttpStatus;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.RestController;
12
13 @RestController
14 public class OrdenadorController {
15     static List<Ordenador> lista = new ArrayList<>();
16     static {
17
18         lista.add(new Ordenador("Yoga", "Lenovo", 800));
19         lista.add(new Ordenador("Air", "Apple", 1000));
20     }
21
22     @GetMapping("/ordenadores")
23     public List<Ordenador> buscarTodos() {
24
25         return lista;
```

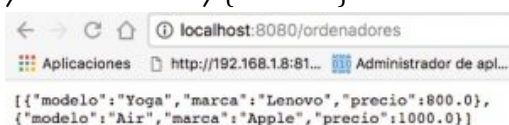
```

36         return new ResponseEntity<Ordenador>(HttpStatus.NOT_FOUND);
37     }
38
39
40     }
41 }

```

Construcción de @RestController

Acabamos de construir un servicio REST con dos URL de acceso /ordenadores y /ordenadores/{modelo}. Si arrancamos la aplicación veremos que accedemos a los datos.

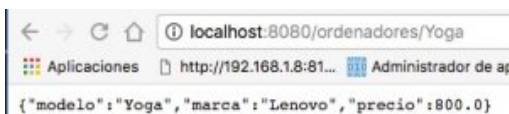


```

[{"modelo": "Yoga", "marca": "Lenovo", "precio": 800.0},
{"modelo": "Air", "marca": "Apple", "precio": 1000.0}]

```

Si pedimos una url concreta con un ordenador específico nos devolverá el ordenador:



```

{"modelo": "Yoga", "marca": "Lenovo", "precio": 800.0}

```

Spring REST Test y Rest Assured

Vamos a utilizar pruebas unitarias con Rest Assured para validar los datos que nos llegan desde ambas url de forma sencilla. El primer paso es obtener las dependencias de Maven.

1

Una vez lo tenemos vamos a construir nuestras primeras pruebas unitarias con este framework que nos compruebe la lista de ordenadores.

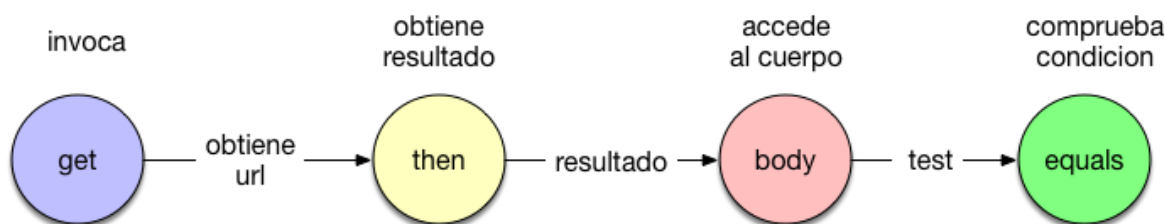
```

1 package com.arquitecturajava.rest;
2
3 import static io.restassured.RestAssured.get;
4 import static org.hamcrest.Matchers.equalTo;
5
6 import org.junit.Test;
7
8 public class RESTTest {
9
10     @Test
11     public void testLista() {
12
13         // ...

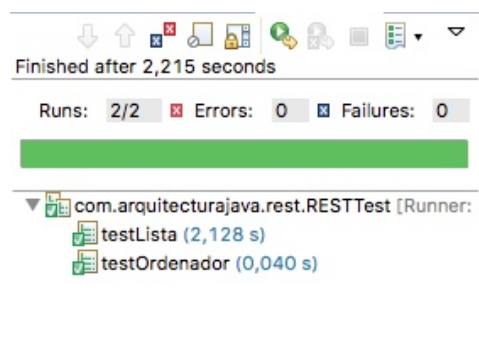
```

Resultado Test

Como podemos ver el DSL de **Rest Assured** es muy claro y permite ejecutar las pruebas unitarias de una forma muy sencilla. En este caso es suficiente con invocar el método `get()` que soporta programación fluida. Una vez ejecutado el método `get()` el método `then()` nos devolverá el resultado. Por último el método `body()` nos permite acceder a la respuesta y comprobar su contenido-



Nos queda por último ejecutar los Test y comprobar que funcionan



Acabamos de usar Rest Assured para lanzar pruebas unitarias contra servicios REST.

1. [Spring REST Client con RestTemplates](#)
2. [¿ Que es REST ?](#)
3. [Spring REST Service con @RestController](#)
4. [REST JSON y Java](#)




0
COMPARTIR

Etiquetada con: [Design Patterns](#)

Leave a Reply

Be the First to Comment!

Notify of



BUSCAR

Mis Cursos de Java Gratuitos

Java Herencia

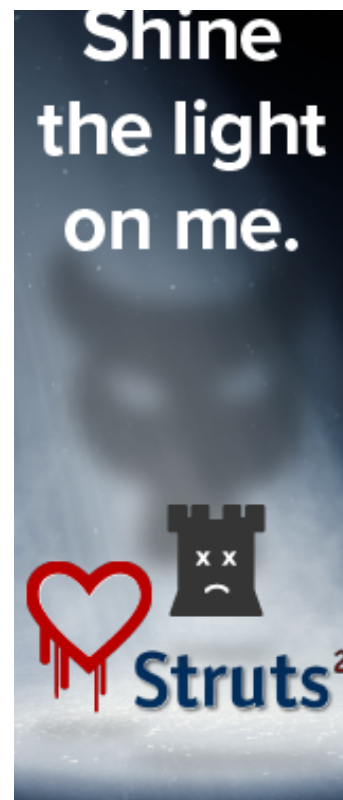


Java JDBC



Servlets





POPULAR

[Maven Parent POM y uso de librerías](#)

[Spring 5 Hello World](#)

[Java Generic Repository y JPA](#)

[Spring Boot WAR sin Microservicios](#)

[Java Interfaces y el concepto de simplicidad](#)

[Spring GetMapping ,PostMapping etc](#)

[Spring REST Client con RestTemplates](#)

[Java 9 Modules y el concepto de modularidad](#)

[PostMan App con Spring REST](#)

[Spring Boot Properties utilizando @Value](#)

CONTACTO

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

[ACEPTAR](#)

[Usando Java Session en aplicaciones web](#)[Java Constructores this\(\) y super\(\)](#)[Java Iterator vs ForEach](#)[Introducción a Servicios REST](#)[¿Cuales son las certificaciones Java?](#)[Java Composite Pattern y recursividad](#)[¿Qué es Gradle?](#)[Java Integer Wrapper y certificacion](#)[Ejemplo de JPA , Introducción \(I\)](#)[REST JSON y Java](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[Usando el patron factory](#)[Java Override y encapsulación](#)[Uso de Java Generics \(I\)](#)[Mis Libros](#)[¿Qué es un Microservicio?](#)[Comparando java == vs equals](#)[Spring MVC Configuración \(I\)](#)

Copyright © 2018 · [eleven40 Pro Theme](#) en [Genesis Framework](#) · [WordPress](#) · [Acceder](#)