Testing in Spring Boot

# Best practices

## All about unit testing: 11 best practices and overview

Mar 30, 2022 - 10 min read

Erica Vartanian

<https://www.educative.io/blog/unit-testing-best-practices-overview>

## Unit Testing Best Practices: 9 to Ensure You Do It Right

By Testim, March 11, 2021

<https://www.testim.io/blog/unit-testing-best-practices/>

## Unit testing best practices with .NET Core and .NET Standard

11/29/2021

14 minutes to read

By [John Reese](https://reese.dev/) with special thanks to [Roy Osherove](https://osherove.com/)

<https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices>

# Mocking

<https://en.wikipedia.org/wiki/Mock_object#Use_in_test-driven_development>

Qué significa mocking en el contexto de software testing.

# Test unitario con Spring Boot, Mockito y JUnit 5 | 1 - Introducción y presentación del proyecto

<https://youtu.be/2lGvzVtA9hU>

5 minutos.

Claro, simple.

Qué son pruebas unitarias.

Qué son pruebas de integración.

Qué son pruebas de aceptación.

Qué son pruebas de stress.

Qué es el framework Mockito.

# TutorialesVIP

## Test unitario con Spring Boot, Mockito y JUnit 5 | 2 - Configuración y funcionamiento del proyecto

<https://youtu.be/iy1Wfsr3Xrs>

6 minutos.

Cloné el repositorio en mi PC. Le cambié el nombre. El proyecto está en la carpeta

C:\GIT\50-testing\TutorialesVIP

URL de prueba:

<http://localhost:8080/country/do>

La consola de H2 está en:

<http://localhost:8080/h2-console>

Aparecen errores de encoding en el browser. Se puede arreglar configurando la base de datos, pero eso no lo vamos a ver por ahora.

## Test unitario con Spring Boot, Mockito y JUnit 5 | 3 - Estructura y funcionamiento del código

<https://youtu.be/x9auKwPoPjI>

8 minutos.

Para qué sirve el archivo imports.sql que está en la carpeta resources.

## Test unitario con Spring Boot, Mockito y JUnit 5 | 4 - Primer test unitario con JUnit

<https://youtu.be/jr8o6m-5WLM>

14 minutos.

La opción de test coverage está disponible solo en la Enterprise Edition. Nosotros usamos la Community. Yo instalé una extensión:

**Coverage Gutters**v2.10.1, by ryanluker

Es simple, cómoda, hace lo que dice. Las explicaciones del tutorial sirven tal cual para esta extensión. Capaz que el autor procuró que fuera así.

¿Se considera una buena práctica imprimir o grabar logs en pruebas unitarias?

Por qué cada prueba unitaria debe ser lo más corta posible.

Qué es coverage.

Como es la convención de nomenclatura de las clases de test.

## Test unitario con Spring Boot, Mockito y JUnit 5 | 5 - Implementación de mocks con mockito framework

<https://youtu.be/boX6H6AARZ0>

12 minutos.

Explicar la relación entre Mockito, pruebas unitarias y pruebas de integración.

## Test unitario con Spring Boot, Mockito y JUnit 5 | 6 - Utilizando los mocks creados con Mockito Mock

<https://youtu.be/1wrno3H5ziU>

7 minutos.

# Code With Arho

## Spring Boot Testing Mini-Series Part #1: Unit Testing

<https://youtu.be/Ae5ukd136pc>

9 minutos.

Qué son pruebas unitarias.

Qué cosas no hace una prueba unitaria.

Qué son pruebas de integración.

Explicar la diferencia entre Spring y Spring Boot para escribir unit tests.

Cuál es el inconveniente de usar Mockito para correr unit tests.

Por qué lo normal es que las pruebas unitarias corran extremadamente rápido.

## Spring Boot Testing Mini-Series Part #2: Web Controllers

<https://youtu.be/NW8i2gna4qA>

## Spring Boot Testing Mini-Series Part #3: Database Tests

<https://youtu.be/wRpxKDi79mk>

## Spring Boot Testing Mini-Series Part #4: Serialization Tests

<https://youtu.be/NQHQ3YVMWaM>

## Spring Boot Testing Mini-Series Part #5: WebClient Tests

<https://youtu.be/GBKY8QyfNDk>

## Spring Boot Testing Mini-Series #6: Integration Tests

<https://youtu.be/N6ZaSNhzsGo>

# Baeldung

Como de costumbre, no es para principiantes. Mejor empezar por otro.

<https://www.baeldung.com/spring-boot-testing>

Descargué el código fuente de GitHub. Está en la carpeta baeldung. Las explicaciones quizá sean útiles para complementar lo que entienda de otros materiales.

# Testing the Web Layer

<https://spring.io/guides/gs/testing-web/>

Documentación oficial del Spring.

El código original tiene algunos pequeños errores, que he corregido. Algunas anotaciones estaban deprecadas, faltaban algunos imports, y cosas por el estilo.

Genero una aplicación Maven. Uso Java 11 porque en este momento tengo mi entorno configurado en esa versión.

Creo un paquete controllers y pongo dentro la clase HomeController.

Las explicaciones, en particular de las anotaciones, son muy interesantes. Vale la pena leerlas con todo cuidado.

Este es un documento de alta calidad. Dice todo lo necesario, no omite nada importante, no da nada por supuesto (al menos, que yo lo haya notado), va al grano, es concreto.

Hay un aviso de anotación deprecada. Actualizar el import según la recomendación y se arregla.

# Discusión de la función de testing

El objeto mockMvc es de tipo MockMvc[[1]](#footnote-1). Entre los métodos de la clase MockMvc está perform:

ResultActions perform(RequestBuilder requestBuilder)

El método perform() recibe un parámetro de tipo (interfaz) RequestBuilder[[2]](#footnote-2) y retorna un objeto de tipo ResultActions[[3]](#footnote-3).

Los métodos estáticos de la interfaz RequestBuilder están implementados en la clase MockMvcRequestBuilders[[4]](#footnote-4). Uno de los métodos de esta clase es get:

static MockHttpServletRequestBuilder get(String urlTemplate, Object... uriVars)

Este es el método que estamos usando en nuestro proyecto. Toma un argumento que es una String que contiene una URL, y retorna un objeto de tipo MockHttpServletRequestBuilder[[5]](#footnote-5). Este objeto se lo retorna al método que lo llamó, que es perform(); este, a su vez, retorna un objeto de tipo (interfaz) ResultActions.

El método print(), que está como argumento del método andDo(), es de la clase MockMvcResultHandlers[[6]](#footnote-6), que es una de las dos que implementan la interfaz ResultActions[[7]](#footnote-7). Este método print() retorna un objeto de tipo interface ResultHandler[[8]](#footnote-8).

1. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/MockMvc.html> [↑](#footnote-ref-1)
2. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/RequestBuilder.html> [↑](#footnote-ref-2)
3. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/ResultActions.html> [↑](#footnote-ref-3)
4. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/request/MockMvcRequestBuilders.html> [↑](#footnote-ref-4)
5. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/request/MockHttpServletRequestBuilder.html> [↑](#footnote-ref-5)
6. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/result/MockMvcResultHandlers.html> [↑](#footnote-ref-6)
7. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/ResultActions.html> [↑](#footnote-ref-7)
8. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/ResultHandler.html> [↑](#footnote-ref-8)