PRUEBAS UNITARIAS Y DE INTEGRACION EN SPRING BOOT CON JUNIT 5

PRUEBAS UNITARIAS Y DE INTEGRACION

Aunque las pruebas unitarias y de integración tiene como finalidad el encontrar errores o defectos en nuestra aplicación, ambas tienen un alcance y objetivo diferente. Mientras que las pruebas unitarias corren pruebas sobre partes del código de forma independiente las pruebas de integración se hacen necesarias cuando queremos probar de una manera sistemica un proyecto como un todo. La combinación de ambas garantiza que el código planteado sea de buena calidad.

JUNIT

En Java existen varios frameworks para realizar pruebas unitarias, el más ampliamente utilizado es JUnit, el cual actualmente se encuentra en su versión 5. Este junto con Mockito conforman las librerías esenciales para pruebas unitarias con Spring.

Usando estos frameworks es posible probar todas las capas de una aplicación, desde la capa de controladores hasta los repositorios de base de datos.

MOCKITO

Mockito es un framework. Es una biblioteca basada en Java que se utiliza para crear APIs de prueba simples y básicas para realizar pruebas unitarias de aplicaciones Java. También se puede utilizar con otros frameworks como JUnit y TestNG.

Mocking es un proceso de desarrollo de los objetos que actúan como el simulacro o clon de los objetos reales. En otras palabras, la simulación es una técnica de prueba en la que se utilizan objetos simulados en lugar de objetos reales para realizar pruebas. Los objetos simulados proporcionan una salida específica (ficticia) para una entrada particular (ficticia) que se le pasa.

La técnica de mock no sólo se utiliza en Java, sino también en cualquier lenguaje de programación orientado a objetos. Hay muchos frameworks disponibles en Java para mocking, pero Mockito es el más popular entre ellos.

Para mockear los objetos, es necesario entender los tres conceptos clave de la burla, es decir, stub, fake y mock. Algunas de las pruebas unitarias implican sólo stubs, mientras que otras implican fake y mocks.

MOCKMVC

MockMvc se utiliza principalmente para las pruebas de la capa web. Las pruebas de la capa web consisten esencialmente en escribir pruebas de grano fino diseñadas específicamente para probar los controladores de su aplicación. Es muy similar a la escritura de pruebas unitarias regulares para las clases en las que se necesitan dependencias simuladas para probar métodos específicos.

TESTRESTTEMPLATE

Cuando se trata de pruebas de integración y validación de APIs basadas en RESTful, tanto TestRestTemplate como RestAssured ofrecen métodos convenientes para crear y ejecutar sus llamadas HTTP con cabeceras personalizadas, auth, tipos de contenido, parámetros de consulta, contenido de carga útil, cookies, etc. La principal diferencia -aparte de la sintaxis- es que TestRestTemplate forma parte del framework de pruebas de Spring que viene incluido en la dependencia spring-boot-starter-test.

En cuanto a las pruebas de integración, Spring proporciona dos formas de realizarlas, la primera de ellas usando una instancia de la clase MockMVC, la cual es un objeto de Spring que permite invocar un servicio dentro del contexto de Spring. La segunda forma es a través de un TestRestTemplate, el cual invoca un servicio de forma externa, como si se estuviera llamando desde un componente completamente independiente de Spring.