Introducción al testing

¿Por que debemos realizar test a nuestro software?

Cuando realizamos pruebas unitarias a nuestro código apostamos a mejorar en los siguientes puntos:

- Mejoramos nuestra calidad de código.
- Probamos que nuestro código hace lo que pensamos que debería hacer.
- Detenemos el "arreglar uno y dañar otro"
- Aportamos conocimientos de buenas practicas a nuestros desarrollos.
- Aprendemos a modificar código con confianza y en funcionamiento.

Terminología testing

Code under test o código bajo prueba: se refiere al código que esta siendo probado.

Test fixture: es una configuración que permite que nuestras pruebas puedan ser realizadas, un ejemplo podría ser

- Mocks objects o carga de objetos.
- Carga de data a una base de datos

Unit tests / unit testing: escribir pruebas para probar alguna porción de código. Los unit test poseen las siguientes características:

- Son diseñados para probar una sección de código especifica.
- La cobertura de código debe estar sobre un 70 a un 80% La cobertura de código se refiere a que porcentaje de nuestro código esta siendo evaluado por pruebas automáticas y
- Deberá ser una unidad y ejecutarse rápido.
- No deben tener dependencias externas. Es decir, sin bases de datos, sin spring context, etc.

Integration testing: diseñadas para probar comportamientos entre objetos y partes del sistema global.

- Alcance mucho mayor
- Pueden incluir el Contexto Spring, la base de datos y los corredores de mensajes
- Se ejecutarán mucho más lentamente que las pruebas unitarias

Pruebas funcionales: normalmente significa que está probando la aplicación en ejecución

- La aplicación está activa, probablemente desplegada en un entorno conocido
- Se prueban puntos de contacto funcionales (es decir, uso de un controlador web, llamada a servicios web, envío/recepción de mensajes, etc.).

- enviar/recibir mensajes

TDD: desarrollo basado en pruebas. Escribe primero las pruebas, codifique para "arreglar" las pruebas, refactorice el código para limpiarlo, mejorarlo, etc.
BDD: desarrollo guiado por el comportamiento. Muy similar a TDD

- Describe el comportamiento esperado del software
- A menudo se expresa como: cuando/entonces; dado/cuando/entonces

¿Qué es mejor utilizar?
Utilice ambos.

Recomendaciones

Entender sobre el code average
HERE'S WHY YOU SHOULD WRITE UNIT TESTS (Lectura recomendada)
10 Reasons Why Unit Testing Matters (Lectura recomendada)