Pruebas para código Java: conceptos básicos y las mejores formas de realizarlas.

Introducción

Las pruebas unitarias son muy importantes para crear software, Le permiten comprobar si cada parte de un programa funciona bien por sí sola, lo que garantiza que el código sea bueno y le ayuda a encontrar problemas a tiempo. Resultado: en este trabajo, veremos los conceptos básicos de las pruebas unitarias en Java, como por qué son importantes, cómo hacerlo y algunos consejos.

¿Qué son las pruebas unitarias?

Salida: Las pruebas unitarias son pruebas que verifican si cada parte del código, como un método o una función, funciona correctamente en un programa. Las pruebas verifican pequeñas partes del código sin necesidad de que otras partes del sistema funcionen.

Las pruebas unitarias son realmente importantes. Las pruebas unitarias son excelentes por varias razones:

Búsqueda temprana de errores: cuando prueba cada parte de su código por sí sola, puede detectar y solucionar problemas rápidamente antes de que afecten a todo el sistema.

Las pruebas unitarias son como documentación viva de cómo debería funcionar cada parte del código, lo que simplifica el cambio y la comprensión del código más adelante.

Salida: escribir pruebas para cada parte del código hace que sea más fácil de entender, probar y corregir.

Cuando probamos una pequeña parte de nuestro código, a menudo tenemos que fingir que está conectada a otras cosas fuera de nuestro código, como un sitio web o una base de datos, para asegurarnos de que nuestro código funcione bien. Para

hacer esto, puedes usar técnicas como burlarse y golpear. Esta oración trata sobre cómo Mockito es una herramienta que te ayuda a crear partes falsas de tu código en Java.

Esta oración significa que necesitamos trabajar juntos con herramientas que nos ayuden a realizar un seguimiento de los cambios en nuestro código.

Salida: Puede simplemente agregar pruebas unitarias a su proceso de integración continua con herramientas como Jenkins, Travis CI o CircleCI. Salida: puede configurar estas herramientas para ejecutar pruebas automáticamente cada vez que alguien realice un nuevo cambio en el código, de modo que pueda asegurarse de que los cambios no arruinen el sistema.

Puede utilizar herramientas como Selenium WebDriver para probar automáticamente la interfaz de usuario de las aplicaciones web Java.

Resultado de Pensamientos adicionales: Probar su código en Java es una parte muy importante de la creación de software. Además de las cosas básicas que necesita saber sobre la escritura de pruebas, también necesita saber algunas cosas interesantes como cómo medir la cantidad de su código que se prueba, cómo probar diferentes entradas y salidas, cómo falsificar algunas partes de su código. y usarlos como sustitutos, y cómo vincular sus pruebas con herramientas que verifican automáticamente si su código funciona. Si hace estas cosas, puede hacer que su software funcione mejor y no se rompa con tanta frecuencia.

Si quieres saber más sobre cómo probar el código Java o tienes alguna pregunta, pregúntame. dar una mano.

Cobertura de código

La cobertura de código es una forma de medir la cantidad de código que realmente has intentado ejecutar cuando lo pruebas. Se pueden utilizar herramientas como JaCoCo para medir y mejorar la cobertura del código.

4. "Trámites que son más complejos".

palabras = oración.split()

Salida: puede usar JUnit para probar su código con diferentes entradas mediante pruebas parametrizadas.

Mockito es un marco que le brinda herramientas para probar cómo se comportan los objetos externos, lo que simplifica descubrir qué partes de su código están funcionando.

5. La mejor manera de hacer algo.

Salida:ly, 5.1 Nombre descriptivo de las pruebas.

Asigne nombres descriptivos a sus pruebas para que pueda descubrir fácilmente qué están verificando y qué se supone que debe hacer.

Sigue probando por tu cuenta. No haga que sus pruebas dependan unas de otras, o podrían estropearse.

el proceso de automatización del desarrollo de software e integración continua. Al agregar pruebas unitarias a su proceso de integración continua, puede encontrar y corregir errores fácilmente mientras mantiene su código en buen estado.

6. Duh, eso es lo que haces al final. Resultado:

"Las pruebas unitarias son una parte crucial de la creación de software en Java". Si escribe buenas pruebas y hace las cosas bien, sus aplicaciones funcionarán bien y no fallarán.

7. Producción:. Desarrollo basado en pruebas: con el ejemplo. Profesional de Addison-Wesley

Producción:, S. y Pryce, N. (2009). "Crear software que sea fácil de entender y trabajar, mediante pruebas posteriores". Profesional de Addison-Wesley

Producción:. Resultado: Esta es una oración que obtuve de un sitio web llamado junit.org.

Producción:.