Pruebas unitarias de JUnit 5 con IntelliJ IDEA

a.micleusanu@ edu.gva.es

1. INTRODUCCIÓN
2. CARACTERÍSTICAS CLAVE DE JUNIT 5
3. CONFIGURACIÓN DEL PROYECTO EN INTELLIJ IDEA
4. ESTRUCTURA DE LAS PRUEBAS
5. CONCLUSIONES

1. INTRODUCCIÓN

Breve explicación

importancia de las pruebas unitarias

Garantizar el correcto funcionamiento de cada parte del código es esencial para el desarrollo de software eficiente.

Esto implica comprender a fondo cada componente y aplicar estrategias efectivas de prueba.

Las pruebas unitarias juegan un papel clave al evaluar individualmente unidades de código, contribuyendo a la calidad al identificar y corregir errores tempranamente.

Actúan como documentación viva, facilitando la colaboración y el mantenimiento a largo plazo del software

JUnit 5 –un estándar para las pruebas unitarias en Java.

-mencionar –

soporte para anotaciones y extensiones más flexibles.

JUnit 5 es el estándar actual para pruebas unitarias en Java, destacándose por mejoras como un sólido soporte para anotaciones y extensiones más flexibles.

Estas características facilitan la escritura y estructuración de pruebas, así como la personalización del marco de prueba según las necesidades del proyecto, lo que contribuye a un desarrollo más eficiente y adaptable

2. Características clave de JUnit 5

Las anotaciones en JUnit proporcionan una manera efectiva de definir métodos de configuración y limpieza antes y después de cada prueba.

-`@Before` se utiliza para marcar un método que se ejecuta antes de cada prueba, permitiendo la preparación de recursos o el establecimiento de condiciones iniciales.

-`@After` señala un método que se ejecuta después de cada prueba, facilitando la limpieza de recursos o la restauración del entorno a su estado original.

-`@Before` para inicializar una conexión a la base de datos antes de cada prueba

- `@After` para cerrar la conexión después de la ejecución de la prueba, asegurando así un entorno controlado y consistente para cada caso de prueba.

2. Características clave de JUnit 5

Las extensiones en JUnit 5 son una característica clave que permite la extensibilidad del marco de pruebas para satisfacer requisitos personalizados. Esta arquitectura JUnit 5 permite crear extensiones personalizadas que determinan el ciclo de vida de las pruebas.

Esta extensión personalizada podría medir el tiempo de ejecución de cada prueba y generar informes detallados, proporcionando así una visión más profunda del rendimiento del código.

Las extensiones en JUnit 5 ofrecen la capacidad de adaptar el marco de pruebas a requisitos particulares, permitiendo a los desarrolladores crear soluciones personalizadas.

2. Características clave de JUnit 5

Las aserciones mejoradas en JUnit 5 se refiere a una mejora significativa en la forma en que se comparan valores en las pruebas unitarias.

A diferencia de las aserciones tradicionales - ofrecen mensajes de **error** más informativos, facilitando la identificación y corrección de problemas.

-permiten **incluir mensajes** personalizados, brindando información específica sobre los valores esperados y los valores reales en caso de que la aserción falle.

Por ejemplo - `assertEquals(expected, actual, "El resultado no coincide con el valor esperado")`, - facilita la comprensión del problema en caso de fallo

3. Configuración del Proyecto en IntelliJ IDEA

Creación de un proyecto en IntelliJ IDEA:

Guía paso a paso sobre cómo iniciar un nuevo proyecto en IntelliJ IDEA.

-Configuración del proyecto para usar JUnit 5:

Mostrar cómo agregar las dependencias necesarias de JUnit 5 al archivo de configuración del proyecto.

Asegurarse de que los ajustes de ejecución reconozcan JUnit 5.

4. ESTRUCTURA DE LAS PRUEBAS

-Creación de un primer conjunto de pruebas:

Crear y ejecutar pruebas simples para asegurarse de que la configuración sea correcta.

- Estructura de las Pruebas

- Organización de las pruebas en paquetes y clases:

La buena organización del código es como tener un libro bien estructurado con capítulos y títulos claros, -facilita encontrar información rápidamente.

En programación, una organización ordenada ayuda a entender, modificar y corregir el código más eficientemente.

En el mantenimiento del código, una buena estructura es como tener un índice bien hecho, haciendo que las tareas de corrección y actualización sean más simples y rápidas.

Veremos ejemplos durante las clases en dentro el marco común

¿Qué son las anotaciones y las extensiones de Junit 5?

son conceptos clave que permiten personalizar y extender el comportamiento de las pruebas unitarias.

Anotaciones en JUnit 5:

- **@Test:**
  - Utilizada para marcar un método como una prueba unitaria.
  - El código dentro del método anotado se ejecutará como parte de la prueba.
- **@BeforeEach y @AfterEach:**
  - @BeforeEach indica que un método debe ejecutarse antes de cada prueba.
  - @AfterEach indica que un método debe ejecutarse después de cada prueba.
  - Útiles para establecer y limpiar condiciones previas a la ejecución de cada prueba.

Anotaciones en JUnit 5:

- **@DisplayName:**
  - Permite asignar un nombre más descriptivo a una prueba para mejorar la legibilidad de los informes de prueba.
- **@Tag:**
  - Permite etiquetar pruebas para organizar y filtrar ejecuciones de prueba según categorías específicas.
- **@Disabled:**
  - Desactiva temporalmente una prueba, lo que puede ser útil cuando una prueba no es aplicable o está en desarrollo.
- **@ParameterizedTest:**
  - Utilizada para indicar que un método es un caso de prueba parametrizado, permitiendo la ejecución de la misma prueba con diferentes conjuntos de datos.

5. CONCLUSIONES

JUnit5: Herramienta esencial para pruebas eficientes.

- Accesible y Gratuita:
- Ideal para usuarios de todos los niveles.

- Recomendación sólida:
- Fácil de aprender y bien integrada en IDEs como IntelliJ IDEA, Eclipse y NetBeans.

- Mejora la calidad del código: -Automatiza pruebas unitarias, contribuyendo a la eficiencia en el desarrollo y mantenimiento del software.