Mockito

Introducción

Mockito es una popular biblioteca de Java diseñada específicamente para la creación de objetos simulados, conocidos como ***"mocks"***, en el contexto de pruebas unitarias. La principal ventaja de usar *"mocks"* es la capacidad de **simular el comportamiento de objetos complejos y sus dependencias**, permitiendo a los desarrolladores centrarse en probar la funcionalidad específica de la clase bajo prueba sin preocuparse por el estado o el comportamiento de sus colaboradores.

Mockito se ha convertido en una herramienta indispensable para los desarrolladores de Java debido a su facilidad de uso y su capacidad para mejorar significativamente la eficacia y la precisión de las pruebas unitarias. A través de **Mockito**, los desarrolladores pueden crear simulaciones controladas de objetos y definir cómo estos objetos deben comportarse durante la ejecución de las pruebas. Esto es especialmente útil en escenarios donde las dependencias externas, como bases de datos, servicios web o componentes de infraestructura, pueden introducir variabilidad y complejidad en las pruebas.

El uso de **Mockito** proporciona varios beneficios clave:

- **Aislamiento y Control**: Mockito permite aislar la clase bajo prueba de sus dependencias. Esto es esencial para asegurar que las pruebas unitarias sean precisas y se enfoquen únicamente en la funcionalidad de la clase bajo prueba. Al controlar el comportamiento de los *"mocks"*, los desarrolladores pueden simular diferentes escenarios, incluyendo casos extremos y condiciones de error, para verificar cómo responde la clase bajo prueba.

- **Simplificación de Pruebas**: En aplicaciones complejas, configurar un entorno de prueba que incluya todas las dependencias reales puede ser una tarea ardua y propensa a errores. Mockito simplifica este proceso al permitir la creación de objetos simulados que replican el comportamiento de las dependencias reales, pero sin la necesidad de configuraciones complicadas.

- **Verificación de Interacciones**: Más allá de simplemente simular comportamientos, Mockito también ofrece la capacidad de verificar que ciertas interacciones ocurrieron como se esperaba. Esto incluye la verificación de llamadas a métodos específicos, con ciertos parámetros, en los *"mocks"*. Esta característica es fundamental para asegurar que la clase bajo prueba interactúa correctamente con sus dependencias.

- **Mejora de la Calidad del Código**: Al facilitar la creación de pruebas unitarias efectivas y específicas, Mockito contribuye a la mejora de la calidad del código. Las pruebas unitarias bien escritas ayudan a detectar y corregir errores en etapas tempranas del desarrollo, lo que reduce significativamente los costos asociados con la corrección de defectos en etapas posteriores del ciclo de vida del software.

- **Facilidad de Uso**: La sintaxis intuitiva y las potentes capacidades de Mockito lo convierten en una herramienta accesible tanto para desarrolladores novatos como para expertos. La comunidad activa y la abundancia de recursos y documentación también hacen que sea fácil encontrar ayuda y ejemplos para resolver problemas específicos.

En resumen, Mockito es una herramienta esencial en el arsenal de pruebas unitarias de cualquier desarrollador Java. Su capacidad para crear y gestionar *"mocks"*, junto con su facilidad de uso y la profundidad de sus funcionalidades, la convierten en una opción preferida para garantizar que las clases se comporten correctamente en aislamiento, contribuyendo así a la creación de software de alta calidad y robustez.

En las siguientes secciones, exploraremos en detalle cómo configurar Mockito, crear y utilizar *"mocks"*, definir comportamientos, verificar interacciones y aplicar las mejores prácticas para maximizar el valor de nuestras pruebas unitarias.

> Sección generada por ChatGPT

Configuración y dependencias

Para comenzar a usar Mockito en tu proyecto de Java, necesitas añadir las dependencias adecuadas en tu archivo de configuración. En esta sección, te mostraremos cómo hacerlo tanto con Maven como con Gradle. También cubriremos cómo utilizar el BOM de Mockito para una gestión más sencilla de las versiones.

Dependencias para Maven

Si tu proyecto utiliza Maven, añade las dependencias de **Mockito** y **JUnit 5** en tu archivo `pom.xml` de la siguiente manera:

```
<dependencies>
    <!-- JUnit 5 -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>

    <!-- Mockito -->
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>4.0.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <version>4.0.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Para simplificar la gestión de versiones y asegurar la compatibilidad entre las diferentes partes de Mockito, puedes utilizar el BOM (Bill of Materials) de Mockito. Añade el BOM en la sección `<dependencyManagement>` de tu archivo `pom.xml`:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-bom</artifactId>
            <version>4.0.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <!-- JUnit 5 -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
```

```
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>

    <!-- Mockito -->
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Usar el BOM garantiza que todas las dependencias relacionadas con Mockito usen versiones compatibles entre sí.

Dependencias para Gradle

Si tu proyecto utiliza Gradle, añade las dependencias de **Mockito** y **JUnit 5** en tu archivo `build.gradle` como sigue:

```
dependencies {
    // JUnit 5
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.2'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.2'

    // Mockito
    testImplementation 'org.mockito:mockito-core:4.0.0'
    testImplementation 'org.mockito:mockito-junit-jupiter:4.0.0'
}
```

En Gradle, puedes utilizar el BOM de Mockito para manejar las versiones de manera más sencilla. Añade el BOM en la configuración de tu proyecto:

```
dependencies {
    // Importar el BOM de Mockito
    implementation platform('org.mockito:mockito-bom:4.0.0')

    // JUnit 5
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.2'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.2'

    // Mockito
    testImplementation 'org.mockito:mockito-core'
    testImplementation 'org.mockito:mockito-junit-jupiter'
}
```

Utilizando el BOM de Mockito, no necesitas especificar las versiones individuales para `mockito-core` y `mockito-junit-jupiter`, ya que el BOM se encarga de establecer las versiones compatibles automáticamente.

- Mockito and JUnit 5 – Using ExtendWith - Baeldung

Creación de Mocks

TODO

Definición de Comportamiento

TODO

Verificación de Interacciones

TODO

Anotaciones de Mockito

TODO

Argument Matchers

TODO

Simulación de Excepciones

TODO

Ejemplos Completos

TODO

Resumen y Buenas Prácticas

TODO

Referencias

- https://site.mockito.org/
- https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html
- https://www.baeldung.com/mockito-series

Licencia