(/)

Invocar un método privado en Java

Última modificación: 8 de marzo de 2021

por baeldung (https://www.baeldung.com/author/baeldung/)
(https://www.baeldung.com/author/baeldung/)

Java (https://www.baeldung.com/category/java/) +

Reflexión (https://www.baeldung.com/tag/reflection/)

1. Información general

Si bien los métodos se hacen *privados* en Java para evitar que sean llamados desde fuera de la clase propietaria, es posible que debamos invocarlos por alguna razón.

Para lograr esto, debemos evitar los controles de acceso de Java. Esto puede ayudarnos a llegar a un rincón de una biblioteca o permitirnos probar algún código que normalmente debería permanecer privado.

En este breve tutorial, veremos cómo podemos verificar la funcionalidad de un método independientemente de su visibilidad. Vamos a considerar dos enfoques diferentes: el API de reflexión de Java (/java-reflection) y de la primavera *ReflectionTestUtils* (/spring-reflection-test-utils) .

2. Visibilidad fuera de nuestro control

Para nuestro ejemplo, usemos una clase de utilidad *LongArrayUtil* que opera en matrices *largas* . Nuestra clase tiene dos métodos *indexOf* :

```
public static int indexOf(long[] array, long target) {
    return indexOf(array, target, 0, array.length);
}

private static int indexOf(long[] array, long target, int start, int end)
{
    for (int i = start; i < end; i++) {
        if (array[i] == target) {
            return i;
        }
    }
    return -1;
}
```

Supongamos que la visibilidad de estos métodos no se puede cambiar y, sin embargo, queremos llamar al método privado *indexOf* .

3. API de reflexión de Java

Si bien el compilador nos impide llamar a una función que no es visible para nuestra clase, podemos invocar funciones a través de la reflexión. Primero, necesitamos acceder al objeto *Método* que describe la función que queremos llamar:

```
Method indexOfMethod = LongArrayUtil.class.getDeclaredMethod(
  "indexOf", long[].class, long.class, int.class, int.class);
```

Tenemos que usar *getDeclaredMethod* para acceder a métodos no privados. Lo llamamos en el tipo que tiene la función, en este caso, *LongArrayUtil* , y pasamos los tipos de los parámetros para identificar el método correcto.
La función puede fallar y lanzar una excepción si el método no existe.

3.2. Permitir que se acceda al método

Ahora necesitamos elevar la visibilidad del método temporalmente:

```
indexOfMethod.setAccessible(true);
```


Este cambio durará hasta que la JVM se detenga o la propiedad *accesible* se establezca de nuevo en *falso*.

3.3. Invocar el método con reflexión

```
int value = (int) indexOfMethod.invoke(
  LongArrayUtil.class, someLongArray, 2L, 0, someLongArray.length);
```

Ahora hemos accedido con éxito a un método privado.

El primer argumento a *invocar* es el objeto de destino, y los argumentos restantes deben coincidir con la firma de nuestro método. Como en este caso, nuestro método es *estático* y el objeto de destino es la clase principal, *LongArrayUtil* . Para llamar a métodos de instancia, pasaríamos el objeto cuyo método estamos llamando.

También debemos tener en cuenta que *invoke* devuelve *Object* , que es *nulo* para funciones *nulas* , y que necesita conversión al tipo correcto para poder usarlo.

4. Spring *ReflectionTestUtils*

Llegar a los internos de las clases es un problema común en las pruebas. La biblioteca de pruebas de Spring proporciona algunos atajos para ayudar a que las pruebas unitarias lleguen a las clases. Esto a menudo resuelve problemas específicos de las pruebas unitarias, donde una prueba necesita acceder a un campo privado que Spring podría instanciar en tiempo de ejecución.

Primero, necesitamos agregar la dependencia de la *prueba de primavera* (https://search.maven.org/classic/#search%7Cgav%7C1%7Cg%3A%22org.springframework%22%20AND%20a%3A%22spring-test%22) en nuestro pom.xml:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.3.4</version>
    <scope>test</scope>
</dependency>
```

Ahora podemos usar la función *invokeMethod* en *ReflectionTestUtils* , que usa el mismo algoritmo que el anterior y nos ahorra escribir tanto código:

Como se trata de una biblioteca de prueba, no esperaríamos usarla fuera del código de prueba.

5. Consideraciones

El uso de la reflexión para evitar la visibilidad de la función conlleva algunos riesgos y es posible que ni siquiera sea posible. Debemos considerar:

- Si Java Security Manager permitirá esto en nuestro tiempo de ejecución
- Si la función que estamos llamando, sin verificación en tiempo de compilación, seguirá existiendo para que la llamemos en el futuro.
- Refactorizar nuestro propio código para hacer las cosas más visibles y accesibles

6. Conclusión

En este artículo, analizamos cómo acceder a métodos privados usando la API de Java Reflection y usando Spring's *ReflectionTestUtils* .

Como siempre, el código de ejemplo de este artículo se puede encontrar en GitHub (https://github.com/eugenp/tutorials/tree/master/core-java-modules/core-java-reflection-2) .

Acceso (https://www.baeldung.com/wp-login.php?redirect_to=https%3A%2F%2Fwww.baeldung.com%2Fjava-call-private-method)

Be the First to Comment!

{} [+]

0 COMENTARIOS

CATEGORIAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
LADO DEL CLIENTE HTTP (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (/JAVA-TUTORIAL)
TUTORIAL DE JACKSON JSON (/JACKSON)
TUTORIAL DE HTTPCLIENT 4 (/HTTPCLIENT-GUIDE)

SOBRE

SOBRE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJOS (/TAG/ACTIVE-JOB/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBE PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
ANÚNCIESE EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)
CONTACTO (/CONTACT)