

( / )

# Afirmar que se produce una excepción en



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

**Pruebas** (<https://www.baeldung.com/category/testing/>)

**JUnit** (<https://www.baeldung.com/tag/junit/>) **JUEGO 5** (<https://www.baeldung.com/tag/junit-5/>)

Acabo de anunciar el nuevo curso *Learn Spring*, centrado en los fundamentos de Spring 5 y Spring Boot 2:

**>> VER EL CURSO** ([/ls-course-start](#))

## 1. Introducción

En este tutorial rápido, veremos cómo probar si se produjo una excepción, utilizando la biblioteca JUnit. Por supuesto, nos aseguraremos de cubrir las versiones JUnit 4 y JUnit 5.

## 2. JUnit 5

La API de aserciones JUnit 5 Jupiter introduce el método *afirmarThrows* para afirmar excepciones.

Esto toma el tipo de la excepción esperada y una interfaz funcional *ejecutable* donde podemos pasar el código bajo prueba a través de una expresión lambda:



```
1  @Test
2  public void whenExceptionThrown_thenAssertionSucceeds() {
3      Exception exception = assertThrows(NumberFormatException.class, () -> {
4          Integer.parseInt("1a");
5      });
6
7      String expectedMessage = "For input string";
8      String actualMessage = exception.getMessage();
9
10     assertTrue(actualMessage.contains(expectedMessage));
11 }
```

Si se produce la excepción esperada, *afirmarThrows* devuelve la excepción, lo que nos permite también afirmar en el mensaje.



que la aserción tenga éxito ya que *Exception* es el supertipo para todas las excepciones.

Si cambiamos la prueba anterior para esperar una *RuntimeException*, esto también pasará:

```
1  @Test
2  public void whenDerivedExceptionThrown_thenAssertionSucceeds() {
3      Exception exception = assertThrows(RuntimeException.class, () -> {
4          Integer.parseInt("1a");
5      });
6
7      String expectedMessage = "For input string";
8      String actualMessage = exception.getMessage();
9
10     assertTrue(actualMessage.contains(expectedMessage));
11 }
```

El método *ClaimThrows()* permite un control más detallado para la lógica de aserción de excepciones porque podemos usarlo alrededor de partes específicas del código.

### 3. JUnit 4

Al usar JUnit 4, simplemente podemos **usar el atributo *esperado de la anotación @Test*** para declarar que esperamos que se **arroje** una excepción en cualquier parte del método de prueba anotado.

Como resultado, la prueba fallará si no se lanza la excepción especificada cuando se ejecuta la prueba y pasará si se lanza:

```
1  @Test(expected = NullPointerException.class)
2  public void whenExceptionThrown_thenExpectationSatisfied() {
3      String test = null;
4      test.length();
5  }
```

En este ejemplo, hemos declarado que esperamos que nuestro código de prueba dé como resultado una *NullPointerException*.



Veamos un ejemplo de verificación de la propiedad del *mensaje* de una excepción:

```
1  @Rule
2  public ExpectedException exceptionRule = ExpectedException.none();
3
4  @Test
5  public void whenExceptionThrown_thenRuleIsApplied() {
6      exceptionRule.expect(NumberFormatException.class);
7      exceptionRule.expectMessage("For input string");
8      Integer.parseInt("1a");
9  }
```

En el ejemplo anterior, primero declaramos la regla *ExpectedException*. Luego, en nuestra prueba, estamos afirmando que el código que intenta analizar un valor *entero* dará como resultado una *excepción* *NumberFormatException* con el mensaje "Para la cadena de entrada".

## 4. Conclusión

En este artículo, nos enfocamos y cubrimos las excepciones de afirmación con JUnit 4 y JUnit 5.

El código fuente completo para los ejemplos está disponible en GitHub (<https://github.com/eugenp/tutorials/tree/master/testing-modules/junit-5-basics>).

**Acabo de anunciar el nuevo curso *Learn Spring*, centrado en los fundamentos de Spring 5 y Spring Boot 2:**

**>> VER EL CURSO (/ls-course-end)**




## ¿Estás aprendiendo a construir tu API con Spring ?

Ingrese su dirección de correo electrónico

**>> Obtenga el libro electrónico**

¡Los comentarios están cerrados en este artículo!

 **ezoic** (<https://www.ezoic.com/what-is-ezoic/>)

[reportar este anuncio](#)

### CATEGORIAS

PRIMAVERA ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/](https://www.baeldung.com/category/spring/))  
DESCANSO ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/](https://www.baeldung.com/category/rest/))  
JAVA ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/](https://www.baeldung.com/category/java/))  
SEGURIDAD ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/](https://www.baeldung.com/category/security-2/))  
PERSISTENCIA ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/](https://www.baeldung.com/category/persistence/))  
JACKSON ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/](https://www.baeldung.com/category/json/jackson/))  
HTTP DEL LADO DEL CLIENTE ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/](https://www.baeldung.com/category/http/))  
KOTLIN ([HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/](https://www.baeldung.com/category/kotlin/))

SERIE

- TUTORIAL DE JAVA "VOLVER A LO BÁSICO" (/JAVA-TUTORIAL)
- JACKSON JSON TUTORIAL (/JACKSON)
- HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
- RESTO CON SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
- TUTORIAL SPRING PERSISTENCE (/PERSISTENCE-WITH-SPRING-SERIES)
- SEGURIDAD CON PRIMAVERA (/SECURITY-SPRING)

ACERCA DE

- META BAELDUNG (HTTP://META.BAELDUNG.COM/)
- EL ARCHIVO COMPLETO (/FULL\_ARCHIVE)
- ESCRIBIR PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
- EDITORES (/EDITORS)
- NUESTROS COMPAÑEROS (/PARTNERS)
- ANUNCIE EN BAELDUNG (/ADVERTISE)
  
- TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
- POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
- INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)
- CONTACTO (/CONTACT)