在使用 **TestNG** 进行测试时，有些场景，我们通过向测试方法传入某些异常的参数，期望代码抛出异常时，我们可以通过 @Test( expectedExceptions, expectedExceptionsMessageRegExp)实现，并且可以实现异常信息的断言。

**运行时异常与检查异常**

Java中 对于异常分为运行时异常与检查异常。

运行时异常，编译时不被检查的异常，不需要用throws 声明抛出 异常对象所属类，也可以不用throw 抛出异常对象或异常引用。对于调用该方法，也不需要放于 try-catch 代码块中。（避免程序代码错误被掩盖在运行中无法察觉)

检查异常，编译时被检测的异常，一旦用throw 抛出异常，如果当前方法可处理异常，那么需要在该方法内通过 try-catch 处理异常。如果当前方法不具备该异常的处理能力，那么必须在参数列表后方法体 前使用 throws 声明异常所属类，交给方法的调用方处理 。

**运行时异常测试（RuntimeException）**

首先先创建一个自定义运行时异常类 CustomRuntimeException，如下

package testng.base.demo;

​

public class CustomRuntimeException extends RuntimeException {

​

   //无参构造方法

   public CustomRuntimeException(){

       super();

  }

​

   //含参的构造方法，指定异常的详细信息

   public CustomRuntimeException(String message){

       super(message);

  }

​

   // 含参的构造方法，指定异常的详细信息和原因

   public CustomRuntimeException(String message, Throwable cause){

       super(message,cause);

  }

​

   // 含参的构造方法，指定异常的原因

   public CustomRuntimeException(Throwable cause) {

       super(cause);

  }

​

}

创建一个测试类：runtimeExceptionTest.jav ，其代码如下所示 -

public class runtimeExceptionTest {

   @Test

   public void testExceptionDemo() {

       throw new CustomException("TestNG custom RuntimeException.");

  }

​

}

​

执行该测试用例，将抛出如下异常信息

testng.base.demo.CustomException: TestNG custom RuntimeException.

​

at testng.base.demo.runtimeExceptionTest.testExceptionDemo(runtimeExceptionTest.java:7)

at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)

at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:86)

at com.intellij.rt.testng.IDEARemoteTestNG.run(IDEARemoteTestNG.java:66)

at com.intellij.rt.testng.RemoteTestNGStarter.main(RemoteTestNGStarter.java:110)

我们期望结果就是抛出该异常信息，但此时用例执行失败了。与我们想要的期望在抛出异常时，用例执行成功的结果不一致，此时我们可以在 @Test注解中通过expectedExceptions参数实现，如下：

package testng.base.demo;

import org.testng.annotations.Test;

​

public class runtimeExceptionTest {

   @Test(expectedExceptions = CustomRuntimeException.class)

   public void testExceptionDemo() {

       throw new CustomRuntimeException("TestNG custom RuntimeException.");

  }

​

}

此时，执行测试用例时，执行结果是成功的，如下

​

===============================================

Default Suite

Total tests run: 1, Failures: 0, Skips: 0

===============================================

我们还可以对抛出异常的信息进行判断，同样在@Test注解中通过expectedExceptionsMessageRegExp参数实现，如下：

package testng.base.demo;

import org.testng.annotations.Test;

​

public class runtimeExceptionTest {

   @Test(expectedExceptions = CustomRuntimeException.class, expectedExceptionsMessageRegExp="TestNG custom RuntimeException.")

   public void testExceptionDemo() {

       throw new CustomRuntimeException("TestNG custom RuntimeException.");

  }

}

​

当抛出的异常信息与expectedExceptionsMessageRegExp参数不一致时，用例将运行失败。

**检查异常测试（Exception）**

首先先创建一个自定义检查异常类 CustomException ，如下

package testng.base.demo;

​

public class CustomException extends Exception {

​

   //无参构造方法

   public CustomException(){

       super();

  }

​

   //有参的构造方法

   public CustomException(String message){

       super(message);

  }

​

   // 用指定的详细信息和原因构造一个新的异常

   public CustomException(String message, Throwable cause){

       super(message,cause);

  }

​

   //用指定原因构造一个新的异常

   public CustomException(Throwable cause) {

       super(cause);

  }

​

}

​

当进行检查类异常测试时，只需要在测试方法的参数列表后方法体 前用 throws 声明异常所属类，代码如下

package testng.base.demo;

import org.testng.annotations.Test;

​

public class exceptionTest {

​

   @Test(expectedExceptions = CustomException.class, expectedExceptionsMessageRegExp="TestNG custom Exception.")

   public void testExceptionDemo() throws CustomException {

       throw new CustomException("TestNG custom Exception.");

  }

​

}

​