# Java 9 改进的 try-with-resources

try-with-resources 是 JDK 7 中一个新的异常处理机制，它能够很容易地关闭在 try-catch 语句块中使用的资源。所谓的资源（resource）是指在程序完成后，必须关闭的对象。try-with-resources 语句确保了每个资源在语句结束时关闭。所有实现了 java.lang.AutoCloseable 接口（其中，它包括实现了 java.io.Closeable 的所有对象），可以使用作为资源。

try-with-resources 声明在 JDK 9 已得到改进。如果你已经有一个资源是 final 或等效于 final 变量,您可以在 try-with-resources 语句中使用该变量，而无需在 try-with-resources 语句中声明一个新变量。

## 实例

import java.io.BufferedReader; import java.io.IOException; import java.io.Reader; import java.io.StringReader; public class Tester { public static void main(String[] args) throws IOException { System.out.println(readData("test")); } static String readData(String message) throws IOException { Reader inputString = new StringReader(message); BufferedReader br = new BufferedReader(inputString); try (BufferedReader br1 = br) { return br1.readLine(); } } }

输出结果为：

test

以上实例中我们需要在 try 语句块中声明资源 br1，然后才能使用它。

在 Java 9 中，我们不需要声明资源 br1 就可以使用它，并得到相同的结果。

## 实例

import java.io.BufferedReader; import java.io.IOException; import java.io.Reader; import java.io.StringReader; public class Tester { public static void main(String[] args) throws IOException { System.out.println(readData("test")); } static String readData(String message) throws IOException { Reader inputString = new StringReader(message); BufferedReader br = new BufferedReader(inputString); try (br) { return br.readLine(); } } }

执行输出结果为：

test

在处理必须关闭的资源时，使用try-with-resources语句替代try-finally语句。 生成的代码更简洁，更清晰，并且生成的异常更有用。 try-with-resources语句在编写必须关闭资源的代码时会更容易，也不会出错，而使用try-finally语句实际上是不可能的。