JDK1.7的新特性

1. **try-with-resources语句**是一个声明一个或多个资源的 **try 语句**。一个资源作为一个对象，必须在程序结束之后随之关闭。 try-with-resources语句确保在语句的最后每个资源都被关闭 。任何实现了 **java.lang.AutoCloseable**的对象, 包括所有实现了 **java.io.Closeable** 的对象, 都可以用作一个资源。（java.io.Closeable是java.lang.AutoCloseable接口的子接口）
2. **示例1：**

**try** (InputStream is = **new** FileInputStream(**new** File("e:\\aaa.txt"));

OutputStream os = **new** FileOutputStream(**new** File("e:\\bbb.txt"));) {

**byte**[] buff = **new** **byte**[10];

**int** len = 0;

**while** (-1 != (len = is.read(buff))) {

os.write(buff, 0, len);

}

os.flush();

}

1. 官方文档翻译：

本文翻译自Oracle官方文档<http://docs.oracle.com/javase/7/docs/technotes/guides/language/try-with-resources.html>,如有不正确的地方，敬请指正，谢谢！

下面的例子读取文件的第一行。它使用了 BufferedReader 的一个实例来读取文件中的数据。BufferedReader 是一个资源，它必须在程序结束之后随之关闭:

static String readFirstLineFromFile(String path) throws IOException {

**try (BufferedReader br = new BufferedReader(new FileReader(path))) {**

**return br.readLine();**

**}**

}

       在这个例子中，**try-with-resources 语句**声明的资源是一个 BufferedReader。声明语句在紧跟在 try 关键字的**括号**里面。Java SE 7以及后续版本中，BufferedReader类实现了java.lang.AutoCloseable接口。 因为 BufferedReader 实例是在 try-with-resource 语句中声明的, 所以不管 try 语句正常地完成或是 发生意外 (结果就是 BufferedReader.readLine 方法抛出IOException)，BufferedReader都将会关闭。

       在 Java SE 7之前, 可以使用 finally 块来确保资源被关闭，不管 try 语句正常地完成或是发生意外。下面的例子使用 **finally 块**替换 try-with-resources 语句:

static String readFirstLineFromFileWithFinallyBlock(String path) throws IOException {

BufferedReader br = new BufferedReader(new FileReader(path));

try {

return br.readLine();

} **finally {**

**if (br != null) br.close();**

**}**

}

       然而，在这个例子中，如果 readLine 和 close 方法均抛出异常，**那么 readFirstLineFromFileWithFinallyBlock 方法将抛出从 finally 块中抛出**的异常**;  try 块中抛出的异常被抑制了**。与此相反, 在 readFirstLineFromFile 这个例子中, 如果 try 块和 try-with-resources 语句均抛出异常, 那么**readFirstLineFromFile 将抛出从 try 块中抛出的异常;  try-with-resources 块抛出的异常被抑制了**。在Java SE 7 以及后续的版本中, 你可以检索被抑制的异常；详情参见 [Suppressed Exceptions](http://docs.oracle.com/javase/7/docs/technotes/guides/language/try-with-resources.html#suppressed-exceptions)。

**可以在一个 try-with-resources 语句中声明一个或多个资源**。下面的例子检索zip文件 zipFileName 中所有文件的名称并创建一个包含那些文件名称的文本文件:

public static void writeToFileZipFileContents(String zipFileName, String outputFileName)

throws java.io.IOException {

java.nio.charset.Charset charset = java.nio.charset.Charset.forName("US-ASCII");

java.nio.file.Path outputFilePath = java.nio.file.Paths.get(outputFileName);

// Open zip file and create output file with try-with-resources statement

**try (**

**java.util.zip.ZipFile zf = new java.util.zip.ZipFile(zipFileName);**

**java.io.BufferedWriter writer = java.nio.file.Files.newBufferedWriter(outputFilePath, charset)**

**)** {

// Enumerate each entry

for (java.util.Enumeration entries = zf.entries(); entries.hasMoreElements();) {

// Get the entry name and write it to the output file

String newLine = System.getProperty("line.separator");

String zipEntryName = ((java.util.zip.ZipEntry)entries.nextElement()).getName() + newLine;

writer.write(zipEntryName, 0, zipEntryName.length());

}

}

}

       在这个例子中， try-with-resources 语句包含两个由分号隔开的声明: ZipFile 和 BufferedWriter。当代码块直接伴随着它正常地或由于一个异常终止时， BufferedWriter 和 ZipFile 对象的 close 方法以这种顺序自动地调用 。**注意：资源的 close 方法调用顺序与它们的创建顺序相反。**

       下面的例子使用一个 try-with-resources 语句来自动关闭一个 java.sql.Statement 对象:

public static void viewTable(Connection con) throws SQLException {

String query = "select COF\_NAME, SUP\_ID, PRICE, SALES, TOTAL from COFFEES";

**try (Statement stmt = con.createStatement())** {

ResultSet rs = stmt.executeQuery(query);

while (rs.next()) {

String coffeeName = rs.getString("COF\_NAME");

int supplierID = rs.getInt("SUP\_ID");

float price = rs.getFloat("PRICE");

int sales = rs.getInt("SALES");

int total = rs.getInt("TOTAL");

System.out.println(coffeeName + ", " + supplierID + ", " + price +

", " + sales + ", " + total);

}

} catch (SQLException e) {

JDBCTutorialUtilities.printSQLException(e);

}

}

       这个例子中使用的 java.sql.Statement 这个资源是JDBC 4.1以及后续版本API的一部分。

**注意: 一个 try-with-resources 语句可以像普通的 try 语句那样有 catch 和 finally 块。在try-with-resources 语句中, 任意的 catch 或者 finally 块都是在声明的资源被关闭以后才运行。**

**被抑制的异常**

       与 try-with-resources 语句关联的代码块可能会抛出异常。在 writeToFileZipFileContents这个例子中,  当试图关闭 ZipFile 和 BufferedWriter 对象时，try 块可能会抛出一个异常，并且 try-with-resources 语句可能抛出多达两个异常 。如果 try 块抛出异常并且 try-with-resources 语句抛出一个或多个异常，那么从 try-with-resources 语句中抛出的异常将会被抑制, 并且块抛出的异常是由 writeToFileZipFileContents 方法抛出的那一个。你**可以通过调用由 try块抛出的异常的Throwable.getSuppressed 方法检索这些被抑制的异常信息**。

**实现了AutoCloseable 或 Closeable 接口的类**

       参见 [AutoCloseable](http://docs.oracle.com/javase/7/docs/api/java/lang/AutoCloseable.html) 和 [Closeable](http://docs.oracle.com/javase/7/docs/api/java/io/Closeable.html) 接口的Javadoc可以看到实现了**两者当中的任何一个接口的类集。**Closeable 接口继承了 AutoCloseable 接口。**Closeable接口的 close 方法抛出IOException 类型的异常而 AutoCloseable 接口的 close 方法抛出 Exception 类型的异常**。因此, subclasses of the AutoCloseable 接口的子类可以重写 close 方法的这个行为来抛出指定的异常，例如 IOException, 或者没有异常。