♥ Reading a text file in Java with the Scanner

Posted on July 6, 2015 by Arulkumaran Kumaraswamipillai

As a Java developer it is a very common task to read file contents to a String object. It is also very common in pre-interview written tests read the contents of a file and apply regex to split string, etc.

4 things to watch-out for in File processing

1) Files must be closed once read. “Try with resources” feauture in java 7 is used to auto close the file once read.
2) Favor reading from a classpath over loading from an absolute path.
3) Scanner is for ASCII files, and a line-oriented scanner cannot be used for binary files. You have no guarantee that

the binary file even has “lines” delimited by newline characters.

4) Reading large files directly into memory can cause memory issues. Read Processing large files efficiently in Java

Java Scanner class in action

#1. Scanner class reading from an absolute file path

readme.txt

```
A big brown fox
jumped over the fence
```

readme.txt

JDK 7 or later must be used to take advantage of the try with resources that auto closes the file.

```
1  package com.read.file;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.util.Scanner;
6
7  public class MyFileReader {
8
9      public static void main(String[] args) {
10         String path = "C:\\Users\\akumaras\\work
11
12         //try with resource auto close the file
13         try (Scanner sc = new Scanner(new File(p
14             String readContent = sc.useDelimiter
15             System.out.println(readContent);
16         } catch (FileNotFoundException e) {
17             e.printStackTrace();
18         }
19     }
20 }
```

The “java.util.regex.Pattern class states:

“\Z” The end of the input but for the final terminator, if any.
“\z” The end of the input.

The extra “\” is added to escape, since backslash is a special character in Java String. For e.g. to print a \ or ” which are

As a Java Architect

Java architecture & design concepts interview Q&As with diagrams | What should be a typical Java EE architecture?

special in string literal you have to escape it with another \ which gives us \\ and \". Similarly, you need to escape "\" in "\Z" with another "\" which becomes "**\\Z**".

#2. Scanner class dynamically constructing an absolute path

```
1  package com.read.file;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.util.Scanner;
6
7  public class MyFileReader {
8
9      public static void main(String[] args) {
10
11         //gives the current working folder
12         String currentWorkDir = System.getProper
13         //convert the package name to path
14         String packagePath = MyFileReader.class.
15
16         try (Scanner sc = new Scanner(new File(c
17             String readContent = sc.useDelimiter
18             System.out.println(readContent);
19         } catch (FileNotFoundException e) {
20             e.printStackTrace();
21         }
22     }
23 }
```

#3. Scanner class reading from the **classpath relatively**

The approaches **#1** and **#2** of reading a file via absolute path is not recommended because if the you move the deployed files to some other location then you will get "FileNotFoundException". A better approach is to read from your classpath. The method "getResourceAsStream" in the java.lang.Class API to the rescue.

```
1  package com.read.file;
2
3  import java.util.Scanner;
4
5  public class MyFileReader {
6
7      public static void main(String[] args) {
8          //getResourceAsStream to the rescue
9          try (Scanner sc = new Scanner( MyFileRea
10             String readContent = sc.useDelimiter
11             System.out.println(readContent);
```

```
12          }
13      }
14  }
```

Why is this a better approach?

If you build a jar or war packaging of the above "MyFileReader" and "readme.txt" it can be deployed anywhere. For example, let's build a jar file with Maven.

Step 1: Create a Maven Jar project

```
1 mvn archetype:generate -DgroupId=com.read.file -D
```

Step 2: Create the files MyFileReader.java & readme.txt

src/main/java: com.read.file.MyFileReader.java
src/main/resources: com.read.file.readme.txt

The "src/main/resources" folder can be created with right mouse click on "readFileWithScanner" and then "new –> Source Folder" and then typing "src/main/resources" as the source folder.

After importing into eclipse looks like:

Java Scanner to read File

Step 3: Ensure that the **pom.xml** uses Java 7 or later

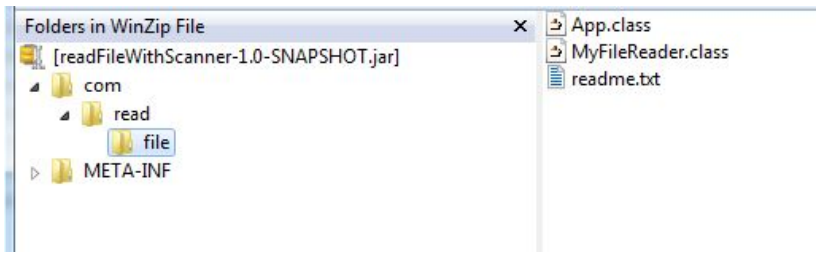
to take advantage of the “Try with resources” feature.

```
<project xmlns="http://maven.apache.org/POM/4.0.
    xsi:schemaLocation="http://maven.apache.org/
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.read.file</groupId>
    <artifactId>readFileWithScanner</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>readFileWithScanner</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</pro
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugin
                <artifactId>maven-compiler-plugi
            </plugin>
        </plugins>

        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.pl
                    <artifactId>maven-compiler-p
                    <version>3.2</version>
                    <configuration>
                        <source>1.7</source>
                        <target>1.7</target>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

Step 4: Build the jar file

```
mvn clean package
```

Step 5: The built jar file looks like



jar file with the readme.txt and reader class file

Step 6: Copy this built jar file to say c:\temp folder and run

```
1 C:\Users\akumaras\projects>java -classpath c:\Tem
2 A big brown fox
3 jumped over the fence
```

So, you can run this jar file in any folder as the lookup of the file is relative to the classpath.

#4. 2 Scanners: one for reading the file line by line & the other for tokenizing the line on spaces

```
1  package com.read.file;
2
3  import java.util.Scanner;
4
5  public class MyFileReader {
6
7      public static void main(String[] args) {
8          Scanner fileScanner = new Scanner(MyFile
9          try {
10             while (fileScanner.hasNextLine()) {
11                 String line = fileScanner.nextLi
12                 Scanner lineScanner = new Scanne
13                 while (lineScanner.hasNext()) {
14                     String token = lineScanner.n
15                     System.out.println(token); /
16                 }
17                 lineScanner.close();
18             }
19         } finally {
20             fileScanner.close();
21         }
22     }
23 }
```

The **output:**

```
1 A
2 big
3 brown
4 fox
5 jumped
6 over
7 the
8 fence
```

You may also like other File I/O posts:

1) Processing large files efficiently in Java

2) 15 Java I/O interview Q&A

Popular Posts

♦ 11 Spring boot interview questions & answers

892 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

851 views

18 Java scenarios based interview Questions and Answers

456 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

411 views

♦ 7 Java debugging interview questions & answers

315 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

313 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

307 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

289 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

267 views

8 Git Source control system interview questions & answers

216 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

‹ Setting up Cygwin with Git Tutorial

06: ♥ Java 8 way of File reading and functionally processing the data ›

Posted in IO

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

 ↑