Reading a resource file from within jar

Ask Question

I would like to read a resource from within my jar like so:

```
File file;
file = new File(getClass().getResource("/file.txt").toURI());
BufferredReader reader = new BufferedReader(new FileReader(file));

//Read the file
```

and it works fine when running it in Eclipse, but if I export it to a jar the run it there is an IllegalArgumentException:

```
Exception in thread "Thread-2"
java.lang.IllegalArgumentException: URI is not hierarchical
```

and I really don't know why but with some testing I found if I change

```
file = new File(getClass().getResource("/file.txt").toURI());
```

to

```
file = new File(getClass().getResource("/folder/file.txt").toURI());
```

then it works the opposite (it works in jar but not eclipse).

I'm using Eclipse and the folder with my file is in a class folder.

java file jar resources embedded-resource

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS

edited Oct 7 at 13:47

Drew MacInnis
4,901 1 13 15

asked Dec 5 '13 at 0:48

PrinceCJC
758 2 6 5

If you want to read files from a directory in jar with any numbers for

`getResourceAsStream` is still a simpler and more portable solution to the problem. – Drew MacInnis Dec 15 '16 at 3:56

11 Answers

Rather than trying to address the resource as a File just ask the ClassLoader to return an InputStream for the resource instead via getResourceAsStream:

```
InputStream in = getClass().getResourc
BufferedReader reader = new BufferedRe
```

As long as the `file.txt` resource is available on the classpath then this approach will work the same way regardless of whether the `file.txt` resource is in a `classes/` directory or inside a `jar`.

The `URI is not hierarchical` occurs because the URI for a resource within a jar file is going to look something like this: `file:/example.jar!/file.txt`. You cannot read the entries within a `jar` (a `zip` file) like it was a plain old File.

This is explained well by the answers to:

- How do I read a resource file from a Java jar file?
- Java Jar file: use resource errors: URI is not hierarchical

edited Jun 13 at 0:26

answered Dec 5 '13 at 1:05

Drew MacInnis
4,901 1 13 15

3 Thank you, this was very helpful and the code works perfectly, but I do have one problem, I need to determine whether the `InputStream` exists (like `File.exists()`) so my game can tell whether to use the default file or not. Thanks. – PrinceCJC Dec 5 '13 at 15:23

3 `getResourceAsStream` returns null if the resource does not exist so that can be your "exists" test. – Drew MacInnis Dec 5 '13 at 19:05

1 BTW, you have a typo: it should be BufferedReader, not BufferredReader (notice the extra 'r' in the later) – mailmindlin Sep 6 '14 at 5:56

1 And of course... don't forget to close the inputStream and BufferedReader – Noremac May 15 '15 at 13:38

To access a file in a jar you have two options:

- Place the file in directory structure matching your package name (after extracting .jar file, it should be in the same directory as .class file), then access it using `getClass().getResourceAsStream("file.txt")`
- Place the file at the root (after extracting .jar file, it should be in the root), then access it using `Thread.currentThread().getContextClassLoader().getResourceAsStream("file.txt")`

The first option may not work when jar is used as a plugin.

answered Sep 17 '16 at 8:55

Juozas Kontvainis
6,097 6 44 63

If you wanna read as a file, I believe there still is a similar solution:

```
ClassLoader classLoader = getClass
File file = new File(classLoader.g
```

answered Nov 16 '14 at 21:04

pablo.vix
1,119 1 10 11

3 URL.getFile() *does not* convert a URL to a file name. It returns the portion of the URL after the host, with all

15 this does not work inside once the program is build to a jar – Akshay Kasar Sep 20 '17 at 12:42

Doesn't work from jar unless you convert to string and save it locally first. – smoosh911 Aug 9 at 16:55

I had this problem before and I made fallback way for loading. Basically first way work within .jar file and second way works within eclipse or other IDE.

```
public class MyClass {

    public static InputStream accessFi
        String resource = "my-file-loc

        // this is the path within the
        InputStream input = MyClass.cl
        if (input == null) {
            // this is how we load fil
            input = MyClass.class.getC
        }

        return input;
    }
}
```

answered Jun 27 '17 at 15:31

MForm
129 3

Make sure that you work with the correct separator. I replaced all `/` in a relative path with a `File.separator`. This worked fine in the IDE, however did not work in the build JAR.

answered Feb 17 '17 at 13:29

Petterson
462 3 14

Up until now (December 2017), this is the only solution I found which works **both** inside and outside the IDE.

Use ***PathMatchingResourcePatternResolver***

```
try {
    // Get all the files under this ir
    String scannedPackage = "my_folder
    PathMatchingResourcePatternResolve
PathMatchingResourcePatternResolver();
    Resource[] resources = scanner.get

    if (resources == null || resources
        log.warn("Warning: could not f
scannedPackage);
    else {
        for (Resource resource : resou
            log.info(resource.getFiler
            // Read the file content (
solutions for that):
            BufferedReader bufferedRea
InputStreamReader(resource.getInputStr
            String line = null;
            while ((line = bufferedRea
                // ...
                // ...
            }
            bufferedReader.close();
        }
    }
} catch (Exception e) {
    throw new Exception("Failed to rea
}
```

answered Dec 20 '17 at 10:42

Naor Bar
574 5 5

You could also just use java.nio. Here is an example to slurp in text from a file at `resourcePath` in classpath:

```
new String(Files.readAllBytes(Paths.ge
```

edited Aug 10 '15 at 23:16

Community ♦
1 1

answered Jul 1 '15 at 19:40

Ayush Gupta
3,845 1 16 16

6 A URI referring to a resource inside a .jar file is not a `file:` URI, so your call to Paths.get will fail. – VGR Mar 5 '16 at 18:08

13 This indeed will fail if the resource is located inside a jar file.I am curious if anyone is aware of a proper way to

read from a jar using the Files class as illustrated in this example, i.e. not

After a lot of digging around in Java, the only solution that seems to work for me is to manually read the jar file itself unless you're in a development environment(IDE):

```
/** @return The root folder or jar fil
public static final File getClasspathF
    return new
File(YourMainClass.class.getProtectior
}

/** @param resource The path to the re
 * @return An InputStream containing t
 *         <b><code>null</code></b> if
public static final InputStream getRes
    resource = resource.startsWith("/'
    if(getClasspathFile().isDirectory(
        return YourMainClass.class.get
    }
    final String res = resource;//Jar
    return AccessController.doPrivileg
        @SuppressWarnings("resource")
        @Override
        public InputStream run() {
            try {
                final JarFile jar = ne
                String resource = res.
                if(resource.endsWith('
normal getResourceAsStream("someFolder
                    ByteArrayOutputStr
                    Enumeration<JarEnt
                    while(entries.hasM
                        JarEntry entry
                        if(entry.getNa
entry.getName().length() > resource.le
                            String nam
                            if(name.co
(name.indexOf("/") == name.lastIndexOf
the children's folders, only the parer
                                name =
name.length() - 1) : name;
                                baos.w
                                baos.w
                                baos.w
                            }
                        }
                    }
                    jar.close();
                    return new ByteArr
                }
                JarEntry entry = jar.g
                InputStream in = entry
                if(in == null) {
                    jar.close();
                    return in;
                }
                final InputStream stre
resources or close jar until the
                return new InputStream
closes all associated InputStreams):
                    @Override
                    public int read()
                        return stream.
                    }

                    @Override
                    public int read(by
```

```
                    @Override
                    public long skip(l
                        return stream.
                    }

                    @Override
                    public int availab
                        return stream.
                    }

                    @Override
                    public void close(
                        try {
                            jar.close(
                        } catch(IOExce
                        }
                        stream.close()
                    }

                    @Override
                    public synchronize
                        stream.mark(re
                    }

                    @Override
                    public synchronize
                        stream.reset()
                    }

                    @Override
                    public boolean mar
                        return stream.
                    }
                };
            } catch(Throwable e) {
                e.printStackTrace();
                return null;
            }
        }
    });
}
```

Note: The above code only seems to work correctly for jar files if it is in the main class. I'm not sure why.

edited Mar 21 at 20:38

answered Mar 21 at 19:21

Brian_Entei
311 1 3 11

You can use class loader which will read from classpath as ROOT path (without "/" in the beginning)

```
InputStream in = getClass().getClassLo
BufferedReader reader = new BufferedRe
```

I think this should be works in java as well. The following code I use is using kotlin.

```
val resource = Thread.currentThread().
```

answered Nov 14 at 7:15

Irvi Firqotul Aini
86 7

If you are using spring, then you can use the the following method to read file from src/main/resources:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import org.springframework.core.io.Cla

  public String readFileToString(Strir

    StringBuilder resultBuilder = new
    ClassPathResource resource = new (

    try (
        InputStream inputStream = resc
        BufferedReader bufferedReader
InputStreamReader(inputStream))) {

      String line;

      while ((line = bufferedReader.re
        resultBuilder.append(line);
      }

    }

    return resultBuilder.toString();
  }
```

edited Nov 27 at 12:39

Tristan
5,222 2 27 56

answered May 3 '17 at 14:53

Sujan M.
11

Welcome to SO. This does not provide an answer to the question. Once you have sufficient reputation you will be able to comment on any post. Also check this what can I do instead. – thewaywewere May 3 '17 at 15:36

Well, yes it does, even before my edit. – Tristan Nov 27 at 12:37