|  |
| --- |
|  |
| User Manual |
| Java GlusterFS |
|  |
| **Ian Royce Herbig and Maylem Gonzalez** |
| **12/2/2014** |

|  |
| --- |
|  |

*Maven is the property of the Apache Software Foundation. Java is the property of Oracle Corporation. GlusterFS is the property of Red Hat, Inc. All other software are the properties of their respective owners. This document is copyright Ian Herbig and Maylem Gonzalez.*

# 1. Introduction

The Java GlusterFS project seeks to bind the Gluster file system to the Java platform. The Gluster file system project supports utilization of a Gluster file system from within a program written in C alone. Other bindings have arisen to enable developers to utilize a Gluster file system from other platforms, like Python or Ruby – this project seeks to enable Java developers to utilize a Gluster file system without having to create their own native interface.

# 2. Hardware and Software Requirements

The hardware requirements for the use of this software are as follows:

* A 64-bit processor

The software requirements for the use of this software are as follows:

* A 64-bit, Linux-based operating system
* Java 7 or greater
* Apache Maven

# 3. Installation and Setup

Installation is easy. Create a Maven project in whatever manner you would prefer (either through your preferred IDE or through Maven’s command line interface; instructions for which can be found on Maven’s website [1]). Then add the following to your Maven’s pom.xml file:

<dependencies>

<dependency>

<groupId>com.peircean.glusterfs</groupId>

<artifactId>glusterfs-java-filesystem</artifactId>

<version>1.0.3</version>

</dependency>

</dependencies>

You can check the version at Maven Central by searching for keyword “glusterfs.”

If you do not wish to have a Maven project, then you can create an ordinary project and then generate a “shaded” JAR file using Maven. This would be done by cloning the GitHub repository [2] and then issuing the final commands in a terminal:

cd glusterfs-java-filesystem

mvn package shade:shade

Maven will output the path of the JAR file to the console, which you can then export to your CLASSPATH environment variable using:

export CLASSPATH=<path-to-shaded.jar>

If you do not want to use Maven at all, you may contact the repository’s maintainer and he will provide you with a shaded JAR.

# 4. Getting Started

Once installed, getting started is familiar if you have worked with files in Java since Java 7. Instead of specifying the path to a file as a string as you would under ordinary circumstances:

Path fooPath = Paths.get(<absolute-path-to-file>);

Instead, you must have a URI that specifies the gluster server’s IP address and the volume’s name. For example, let’s say you want to access a file called “bar” on your gluster server which is located at IP 172.31.31.31 and “bar” is located on a volume called “foo,” then you must do the following.

String glusterIdentifier = “gluster://”;

String glusterIP = “172.31.31.31”;

String volumeName = “foo/”;

URI barURI = new URI(glusterIdentifer + glusterIP + “:” + volumeName + “bar”);

Path barPath = Paths.get(barURI);

From this, you can see that you can simply create a single String that you build paths with:

public static final String glusterRoot = “gluster://172.31.31.31:foo/”;

URI barURI = new URI(glusterRoot + “bar”);

Path barPath = Paths.get(barURI);

Once you have the path to the file, you can perform almost any action on it that the Files API allows. Writing to it is as simple as putting the bytes you want written into an array and calling Files.write()!

byte[] bytes = “Hello, world!”.getBytes()

Files.createFile(barPath, PosixFilePermissions.asFileAttribute(PosixFilePermissions.fromString(“rw-rw-r--")));

Files.write(barPath, bytes, StandardOpenOption.WRITE);

byte[] hello = Files.readAllBytes(barPath);

Arrays.equals(bytes, hello); //this is true

Path fooPath = Paths.get(glusterRoot + “foo”);

Files.copy(barPath, fooPath, StandardCopyOptions.CopyAttributes);

Files.delete(barPath);

try {

hello = Files.readAllBytes(barPath);

} catch (IOException e) {

System.out.println(“You can’t read a file that doesn’t exist.”);

}

byte[] fooContents = Files.readAllBytes(fooPath);

Arrays.equals(hello, fooContents); //this is true as well

See Oracle’s Java tutorials [3] and the JavaDocs [4] for the Files class for more information.

# 5. Accessing Help Online

For help using the project, you may contact Louis Zuckerman:

GitHub: semiosis

IRC: semiosis in #gluster on Freenode IRC

Twitter: @pragmaticism

# 6. References

[1] Maven Central – <http://search.maven.org/>

[2] Java GlusterFS GitHub Repository - <https://github.com/semiosis/glusterfs-java-filesystem>

[3] Oracle’s Java Tutorials - <https://docs.oracle.com/javase/tutorial/essential/io/fileio.html>

[4] JavaDoc for Files - <http://docs.oracle.com/javase/7/docs/api/java/nio/file/Files.html>