ExtractAndDecompressGzipFiles

Back to the VfsCookbook

Overview

Try using VFS to read the content of a compressed (gz) file inside of a tar file. Extract tar file objects. If they are gzip files, decompress them. Any directory structure in the tarfile is not being preserved, the contents are pulled out to the same location regardless of directory hierarchy (for the purposes of this example, all objects in the tar file have unique names, so there are no file name conflicts).

Use a two phase approach.

1. look at each of the files in the tar file 2. if it's a directory, recursively process it, otherwise
    - if it's a non-gzipped file, extract it to a file
    - if it's a gzipped file, decompress gzipped content to file

Conceptually there is a tar file:

```
archive.tar
 +- tardir/
     +- content.txt.gz
     +- non-gzip.txt
```

I'd like to end up with an uncompressed file "content.txt" and "non-gzip.txt".

Sample data file

Create this sample `archive.tar` file with some (unix) commands along the lines of:

```
ls -l > content.txt
gzip content.txt
ls -l > non-gzip.txt
mkdir tardir
mv content.txt.gz non-gzip.txt tardir
tar cvf archive.tar tardir
rm -r tardir
```

The contents of the `content.txt` and `non-gzip.txt` files are just a directory listings, dump in anything you want here. For this example the sample `archive.tar` is located in the `/extra/data/tryVfs` directory. You can see that hardcoded in the java example below. The `content.txt` and `non-gzip.txt` files will be extracted into the same location.

Key Concepts

Building the resolveFile 'name' String

An essential ingredient for this "recipe" is the **name** argument for the `FileSystemManager.resolveFile(String name)` method. See this in the lines defining and using `String gzName`, line numbers 99-101 in the ExtractFromGzipInTar.java code listing below. The important work of connecting to the content.txt file inside the content.txt.gz file inside the archive.tar file is performed by

```
FileSystemManager fsManager = VFS.getManager();
FileObject file = fsManager.resolveFile( "gz:tar:file:///extra/data/tryVfs/archive.tar!/tardir/content.txt.gz!
content.txt" );
```

In order to build similar strings for your own purposes, you will need to understand what is going on here. The paths to the file of interest are chained together with the "!" character as a separator. At the same time the corresponding file system scheme designators ("file:", "tar:" and "gz:") should be prepended onto the front in reverse order. Taking this one step at a time, we have the full path to the `archive.tar` file (/extra/data/tryVfs/archive.tar), which is accessed through the normal file system *file:*

file:+///extra/data/tryVfs/archive.tar+

Now we will treat the file as a **tar:** file and navigate inside this archive by appending a "!" and specifying the path `/tardir/content.txt.gz`.

tar:`file:///extra/data/tryVfs/archive.tar`!`/tardir/content.txt.gz`

Finally we will switch to the **gz:** file system to read the uncompressed `content.txt` (again using the "!" separator character)

gz:`tar:`file:///extra/data/tryVfs/archive.tar!/tardir/content.txt.gz!`content.txt`

Generic Drill-down

On line 90 I'm giving special attention to gzip files

```
if (extractFile.getName().getExtension().equals("gz"))
```

and other types of compression like zip and bzip2 (as well as nested archives like jar and tar) will not be expanded. To generically drill down and expand zip, bzip2, jar, tar files to arbitrary depth, eliminate the "gz" specific code and use instead

```
if (manager.canCreateFileSystem(extractFile))
{
    FileObject innerFile = manager.createFileSystem(extractFile);
}
```

pom.xml Project file

This example uses Maven2. There is a **`pom.xml`** to define the project

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>gov.noaa.eds</groupId>
    <artifactId>tryVfs</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>Try apache commons vfs</name>
    <url>http://maven.apache.org</url>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.5</source>
                    <target>1.5</target>
                </configuration>
            </plugin>
            <plugin>
                <!-- Usage: mvn assembly:assembly -->
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                    <archive>
                        <manifest>
                            <mainClass>gov.noaa.eds.tryVfs.ExtractFromGzipInTar</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>commons-vfs</groupId>
            <artifactId>commons-vfs</artifactId>
            <version>1.0</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Source Code

Content of **`src/main/java/gov/noaa/eds/tryVfs/ExtractFromGzipInTar.java`**

```
/*
 * ExtractFromGzipInTar.java
 */
package gov.noaa.eds.tryVfs;

import org.apache.commons.vfs.AllFileSelector;
import org.apache.commons.vfs.FileName;
import org.apache.commons.vfs.FileObject;
import org.apache.commons.vfs.FileSystemException;
import org.apache.commons.vfs.FileSystemManager;
import org.apache.commons.vfs.FileType;
import org.apache.commons.vfs.FileTypeSelector;
import org.apache.commons.vfs.VFS;
```

```
import org.apache.commons.vfs.provider.local.LocalFile;

/**
 * Try using VFS to read the content of a compressed (gz) file inside of
 * a tar file. Extract tar file objects. If they are gzip files, decompress them.
 * Any directory structure in the tarfile is not being preserved, the contents
 * are pulled out to the same location regardless of directory hierarchy (for
 * the purposes of this example, all objects in the tar file have unique names,
 * so there are no file name conflicts).
 *
 * @author Ken Tanaka
 */
public class ExtractFromGzipInTar
{
    FileSystemManager fsManager = null;
    static String extractDirname = "/extra/data/tryVfs";

    /**
     * Extract files from a tar file. If the file extracted is gzipped,
     * decompress it and remove the gzipped version.
     * @param args command line arguments are currently not used
     */
    public static void main( String[] args )
    {
        ExtractFromGzipInTar extract = new ExtractFromGzipInTar();

        try {
            extract.fsManager = VFS.getManager();
        } catch (FileSystemException ex) {
            throw new RuntimeException("failed to get fsManager from VFS", ex);
        }


        /* Create a tarFile FileObject to connect to the tarfile on disk */
        FileObject tarFile;
        try {
            String tarName = new String("tar:file://" + extractDirname + "/archive.tar");
            System.out.println("Resolve " + tarName);
            tarFile = extract.fsManager.resolveFile(tarName);

            FileName tarFileName = tarFile.getName();
            System.out.println("  Path     : " + tarFileName.getPath());
            System.out.println("  URI      : " + tarFileName.getURI());
        } catch (Exception ex) {
            throw new RuntimeException("failed to open tar file ", ex);
        }

        /* Work on files inside tarFile */
        FileObject[] children;
        try {
            children = tarFile.getChildren();
        } catch (FileSystemException ex) {
            throw new RuntimeException("failed to get contents of tarfile ", ex);
        }

        for (FileObject f : children) {
            extract.processChild(f);
        }
    } // main( String[] args )

    private void processChild(FileObject f) {
        try {
            if (f.getType() == FileType.FOLDER) {
                // Recursively process files in this folder
                FileObject[] children = f.getChildren();
                for (FileObject subfile : children) {
                    processChild(subfile);
                }
            } else {
                FileName fname = f.getName();
                String extractName = new String("file://" + extractDirname + "/"
```

```
                        + fname.getBaseName());
                System.out.println("Extracting " + extractName);
                LocalFile extractFile = (LocalFile) this.fsManager.resolveFile(extractName);

                // if the file is gzipped, decompress it
/* line  90 */  if (extractFile.getName().getExtension().equals("gz")) {
                    System.out.println("Decompressing " + extractName);

                    // The uncompressed filename we seek
                    // content.txt
                    String fileName = extractFile.getName().getBaseName().replaceAll(".gz$", "");

                    // Build the direct path to the uncompressed content of the
                    // gzip file in the tar file.
                    // gz:tar:file:///archive.tar!/tardir/content.txt.gz!content.txt
/* line 100 */      String gzName = new String("gz:" + fname.getURI() + "!" + fileName);
                    FileObject gzFile = this.fsManager.resolveFile(gzName);

                    // The decompressed path we want
                    String decompName = new String("file://" + extractDirname + "/"
                            + fileName);
                    LocalFile decompFile = (LocalFile) this.fsManager.resolveFile(decompName);

                    // Some debug lines
                    System.out.println("fileName   =" + fileName);
                    System.out.println("decompName =" + decompName);
                    System.out.println("gzName=" + gzName);

                    // Extracting
                    decompFile.copyFrom(gzFile, new FileTypeSelector(FileType.FILE));
                } else {
                    // just extract the non-gzip file
                    extractFile.copyFrom(f, new AllFileSelector());
                }
            }
        } catch (FileSystemException ex) {
            ex.printStackTrace();
            throw new RuntimeException("Error working on tarfile object " + f.getName());
        }
    } // processChild(FileObject f)
}
```

Compiling

Compile the source code with

```
mvn assembly:assembly
```

This will create an executable jar file in the standard `target` directory.

Running

Use a command like this to run the example

```
java -jar target/tryVfs-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Sample Output

```
Nov 7, 2007 12:22:01 PM org.apache.commons.vfs.VfsLog info
INFO: Using "/tmp/vfs_cache" as temporary files store.
Resolve tar:file:///extra/data/tryVfs/archive.tar
  Path     : /
  URI      : tar:file:///extra/data/tryVfs/archive.tar!/
Extracting file:///extra/data/tryVfs/non-gzip.txt
Extracting file:///extra/data/tryVfs/content.txt.gz
Decompressing file:///extra/data/tryVfs/content.txt.gz
fileName   =content.txt
decompName =file:///extra/data/tryVfs/content.txt
gzName=gz:tar:file:///extra/data/tryVfs/archive.tar!/tardir/content.txt.gz!content.txt
```

In addition to the `archive.tar` file, there should now be `content.txt` and `non-gzip.txt` files in the same location.