Java 11 Single-File Applications

Replacing bash with a syntax I actually know

Steve Corwin
Corwin Technology LLC

Why use Java instead of shell script?

Frustration.

```
# ...working shell script...
VALUES=($RESULTS)
V500S=${VALUES[0]}
V504S=${VALUES[1]}

COUNT500S=$(echo "${V500S//[$'\t\r\n ']}")
COUNT504S=$(echo "${V504S//[$'\t\r\n ']}")

echo "\"$SERVER\", \"$COUNT500S\",
 \"$COUNT504S\""
```

Basics of single-file applications

JEP 330: https://openjdk.java.net/jeps/330

- Java 11
- Java app where everything is within a single file, including

```
public static void main(String[] args) {
```

- invoke using java or shebang

Begin with HelloWorld (where else?)

```
public class HelloWorld {
  public static void main(String[]
  args) {
    System.out.println("Hello,
  world!  I am a plain .java
  file!");
  }
}
$ /home/scorwin/apps/java/jdk-
  11.0.1/bin/java HelloWorld.java
```

- standard main() method
- my default is Java 8

Shebang

HelloWorldShebang:

```
#!/home/scorwin/apps/java/jdk-11.0.1/bin/java --
   source 11
public class HelloWorldShebang {
  public static void main(String[] args) {
    System.out.println("Hello, world!  I have a
   shebang but no file extension.");
  }
}
$ ./HelloWorldShebang
```

- shebang must include --source option
- file must be marked executable
- Eclipse doesn't recognize as Java code

Shebang with .java

HelloWorldShebang.java:

```
//#!/home/scorwin/apps/java/jdk-11.0.1/bin/java --
   source 11
public class HelloWorldShebang {
  public static void main(String[] args) {
    System.out.println("Hello, world!  I have a
   shebang and a file extension.");
  }
}
```

- with the shebang line commented out:

  ```
  $ /home/scorwin/apps/java/jdk-11.0.1/bin/java
   HelloWorldShebang.java
  ```

- with the shebang line uncommented:

  ```
  $ ./HelloWorldShebang.java
  ```
  -> fails, errors due to shebang line

  ```
  $ /home/scorwin/apps/java/jdk-11.0.1/bin/java
   HelloWorld.java
  ```
  -> fails, errors due to shebang line

Shebang with .jv

HelloWorldShebang.jv:

```
#!/home/scorwin/apps/java/jdk-11.0.1/bin/java --
   source 11
public class HelloWorldShebang {
  public static void main(String[] args) {
    System.out.println("Hello, world!  I have a
   shebang and a .jv file extension.");
  }
}
$ ./HelloWorldShebang.jv
```

- shebang must include --source option
- file must be marked executable
- Eclipse doesn't recognize as Java code

Workaround: symbolic link

```
$ ls -lh GrepFor500s504sCSV*
lrwxrwxrwx 1 scorwin scorwin   23 Nov 17
  21:40 GrepFor500s504sCSV ->
  GrepFor500s504sCSV.java
-rwxrwxr-x 1 scorwin scorwin 5.3K Nov 18
  12:01 GrepFor500s504sCSV.java
```

- symlink has same name but no file extension
- .java file must be marked executable
- open .java file in IDE
- either comment out the shebang line temporarily to get rid of the error, or just ignore that particular error

GrepFor500s504sCSV.java

```
#!/home/scorwin/apps/java/jdk-11.0.1/bin/java --source 10
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
/**
 * Usage:
 *                                    $ ~/apps/java/jdk-11.0.1/bin/java GrepFor500s504sCSV.java access.log 2>&1 | grep '^"'
 * OR using the symlink and with the shebang line uncommented (and GrepFor500s504sCSV.java marked executable):
 *                                    $ ./GrepFor500s504sCSV access.log 2>&1 | grep '^"'
 */
public class GrepFor500s504sCSV {
private static List<String> directoryNames = Arrays.asList(
"server1", "server2", "server3", "server4", "server5", "server6", "server7");
private static List<String> directory1 = Arrays.asList("server1");
private static List<String> serversToCheck = directoryNames;
//                  private static List<String> serversToCheck = directory1;
public static void main(String[] args) {
checkArgs(args);
String accessLogName = args[0];
System.out.println("accessLogName: " + accessLogName);
System.out.println("serversToCheck: " +serversToCheck);
boolean firstServer = true;
for (String serverName : serversToCheck) {
grepFor500s504s(serverName, firstServer, "/home/scorwin/data/demoData", accessLogName);
//check500s504s(serverName, firstServer, accessLogName);
firstServer = false;
}
}
private static void checkArgs(String[] args) {
if (args.length != 1) {
System.err.println("access log name argument like access.log-20181015 required, " + args.length + " argument(s) provided");
System.exit(-1);
}
```

GrepFor500s504sCSV.java - part 2

```
private static void grepFor500s504s(String serverName, boolean firstServer, String accessLogBasePath, String accessLogName) {
if (firstServer) {
System.out.println("\nCounting 500s and 504s in directories inside " + accessLogBasePath + ", access log " + accessLogName + "\n");
System.out.println("\"server name\", \"count of 500s\", \"count of 504s\"");
}
System.out.println("\nbeginning server " + serverName);
// grep "HTTP/1.1' 500 " /home/scorwin/data/demoData/server1/access.log | wc -l
String command500s = "grep \"HTTP/1.1' 500 \" " + accessLogBasePath + "/" + serverName + "/" + accessLogName + " | wc -l";
System.out.println("command500s: " + command500s);
List<String> outputList500s = executeCommand(command500s);
//outputList500s.forEach(line -> System.out.println(line));
String command504s = "grep \"HTTP/1.1' 504 \" " + accessLogBasePath + "/" + serverName + "/" + accessLogName + " | wc -l";
System.out.println("command504s: " + command504s);
List<String> outputList504s = executeCommand(command504s);
//outputList504s.forEach(line -> System.out.println(line));
System.out.println(String.format("\"%s\",\"%s\",\"%s\"", serverName, outputList500s.get(0), outputList504s.get(0)));
}
private static void check500s504s(String serverName, boolean firstServer, String accessLogName) {
if (firstServer) {
System.out.println("\nCounting 500s and 504s in access log " + accessLogName + "\n");
System.out.println("\"server name\", \"count of 500s\", \"count of 504s\"");
}
System.out.println("\nbeginning server " + serverName);
// ssh -t -o ConnectTimeout=15 server1 grep "HTTP/1.1' 500 " /var/log/tomcat7/access.log | wc -l;  grep "HTTP/1.1' 504 " /var/log/tomcat7/access.log | wc -l
String command = "ssh -t -o ConnectTimeout=15 " + serverName + " grep \"HTTP/1.1' 500 \" /var/log/tomcat7/" + accessLogName + " | wc -l; "
+ " grep \"HTTP/1.1' 504 \" /var/log/tomcat7/" + accessLogName + " | wc -l";
List<String> outputList = executeCommand(command);
System.out.println(String.format("\"%s\",\"%s\",\"%s\"", serverName, outputList.get(0), outputList.get(1)));
}
```

GrepFor500s504sCSV.java - part 3

```
private static List<String> executeCommand(String command) {
List<String> outputList = new ArrayList<>();
// any command which produces more than 64K of output will produce an apparent hang.  Creating BufferedReader with buffer size = 65536 didn't help.
Process process;
try {
List<String> commands = new ArrayList<String>();
// to get grep to work it needs to be invoked by '/bin/sh -c'.  Otherwise get errors like
//   'java.io.IOException: Cannot run program "grep -c "HTTP/1.1' 500 " /home/scorwin/data/demoData/server7/access.log": error=2, No such file or directory'
// using '/bin/sh -c' means pipes work, for example " | wc -l" returns a count instead of an error.
// 'wc -l access.log' and 'tail -5 access.log' still work when invoked by '/bin/sh -c'
// have yet to get ack to work.
            commands.add("/bin/sh");
            commands.add("-c");
            commands.add(command);
ProcessBuilder processBuilder = new ProcessBuilder(commands);
processBuilder.redirectErrorStream(true);  // so stderr is visible
process = processBuilder.start();
//process = Runtime.getRuntime().exec(command);  // this was sufficient until I tried to run grep by itself...
//System.out.println("process: " + process);
process.waitFor();
BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
String line = "";
while ((line = reader.readLine()) != null) {
//System.out.println(line.trim());
outputList.add(line.trim());
}
//System.out.println("process: " + process);
} catch (Exception e) {
e.printStackTrace();
}
return outputList;
}
}
```

Libraries beyond JDK built-ins

HelloWorldNeedsClasspath.java:

```
// #!/home/scorwin/apps/java/jdk-11.0.1/bin/java --source 11
   --class-path
   /home/scorwin/.m2/repository/org/apache/commons/commons-
   lang3/3.6/commons-lang3-3.6.jar
import org.apache.commons.lang3.StringUtils;
  public static void main(String[] args) {
    System.out.println(StringUtils.isBlank("bob"));
    System.out.println("Hello, world!  I also have a shebang
   and a .java file extension.");
  }
}
```

- with shebang commented out:

```
$/home/scorwin/apps/java/jdk-11.0.1/bin/java --source 11 --
   class-path
   /home/scorwin/.m2/repository/org/apache/commons/commons-
   lang3/3.6/commons-lang3-3.6.jar
   HelloWorldNeedsClasspath.java
```

- never got shebang to work

Limitations

- any command which produces more than 64K of output will produce an apparent hang when using my code
- using shebang with libraries beyond built-ins may not be possible

Q & A