Assignment 3: GIT Version Control and Apache ANT

Niall Rodgers

October 2018

1 Download the Repository

This assignment aimed to teach the fundamentals of GIT, a version control system, and to use ANT to write a build file for a java based application project.

Unfortunately as one of the students who was unable to gain access to the TAU server. I was forced to download the alternative file from the course home page. I will however still go through the steps to clone a pre-existing repository.

The command is as follows:

```
$ git clone [insert URL address here]
```

The GIT VCS also contains a number of features which we will explore further on in this document. After cloning the directory named `Project` was created. Our goal after cloning this repository was to rewrite the `build.xml`

2 rewriting build.xml

The build.xml needed to contain at least 4 targets.

- INIT-To create all the needed directories for the project. this was done using the textttmkdir tag. Creating a directory named `build` with two subdirectories of: `classes` and `lib`
- COMPILE-To compile all the java code in the source directory to the created `classes` subdirectory.
- JAR-To create a Java ARchive(JAR) with a Main-Class and store this in the `lib` subdirectory.
- RUN-To execute the created JAR file within the `lib` directory.
- CLEAN-To remove all files and directories created by the program.

3 The Re-Written build.xml

Below is a copy of the rewritten build.xml

```
<project name="compareShapes" default="clean" basedir=".">
<description>
Application to compare the area of a number of different shapes
</description>


<!-- set global properties for this build -->
<property name="src" location="."/>
<property name="build" location="build"/>


<target name="initialise">
<!-- Create the required directories: /build, /build/lib, /build/classes -->
<mkdir dir="${build}/lib"/>
<mkdir dir="${build}/classes"/>
</target>

<target name="compile" depends="initialise"
description="compile the source">
<!-- Compile class files from ${src}to ${build}/lib -->
<javac srcdir="${src}" destdir="${build}/classes"/>
</target>

<target name="jar" depends="compile"
description="create executable jar">
<!--create a JAR with the Main-Class and store file in /lib subdirectory -->
<jar jarfile="${build}/lib/mainProgram.jar" basedir="${build}/classes">
<manifest>
<attribute name="Main-Class" value="TestShape"/>
</manifest>
</jar>
</target>

<target name ="run" depends="jar"
description="runs main program">
<!--will run mainProgram.jar -->
```

```
<java jar="${build}/lib/mainProgram.jar" fork="true"/>
</target>

<target name="clean" depends="run"
description="clean up">
<!-- Delete ${build} and it's subdirectories-->
<delete dir="${build}"/>
</target>

</project>
```

4 Inserting The New build.xml Into The Repository

In order to check the new `build.xml` into the Git repository the following command needed to be used:

```
$ git add build.xml
```

This command will insert build.xml into the local git repository's staging area. To see what files are currently within the staging area use:

```
$ git status
```

Once the status of the staging area is assessed, we must commit the changes within the staging area to the local repository:

```
$ git commit
```

This command will return a document within the terminal (using VI) which list the changes to be committed to the local repository. The document will ask the user to delete all comment characters which are excluding the wanted changes. Once saved this document will pass the files to the local repository.

If the repository has a remote origin, the changes to the local repository can also be transmitted to the origin's location.

```
$ git push -u origin master
```

As I have an alternative version of this GIT repository on my GitHub page, the output is from me pushing a duplicate repository to that page.