ANT - CREATING JAR FILES

The next logical step after compiling your java source files, is to build the java archive, i.e., the JAR file. Creating JAR files with Ant is quite easy with the **jar** task. The commonly used attributes of the jar task are as follows:

Attributes	Description
basedir	The base directory for the output JAR file. By default, this is set to the base directory of the project.
compress	Advises Ant to compress the file as it creates the JAR file.
keepcompression	While the **compress** attribute is applicable to the individual files, the **keepcompression** attribute does the same thing, but it applies to the entire archive.
destfile	The name of the output JAR file.
duplicate	Advises Ant on what to do when duplicate files are found. You could add, preserve, or fail the duplicate files.
excludes	Advises Ant to not include these comma separated list of files in the package.
excludesfile	Same as above, except the exclude files are specified using a pattern.
inlcudes	Inverse of excludes.
includesfile	Inverse of excludesfile.
update	Advises Ant to overwrite files in the already built JAR file.

Continuing our **Hello World** Fax Application project, let us add a new target to produce the jar files. But before that, let us consider the jar task given below.

```
<jar destfile = "${web.dir}/lib/util.jar"
   basedir = "${build.dir}/classes"
   includes = "faxapp/util/**"
   excludes = "**/Test.class" />
```

Here, the **web.dir** property points to the path of the web source files. In our case, this is where the util.jar will be placed.

The **build.dir** property in this example points to the build folder where the class files for the util.jar can be found.

In this example, we create a jar file called **util.jar** using the classes from the **faxapp.util.*** package. However, we are excluding the classes that end with the name Test. The output jar file will be placed in the web application lib folder.

If we want to make the util.jar an executable jar file we need to add the **manifest** with the **Main-Class** meta attribute.

Therefore, the above example will be updated as:

```
<jar destfile = "${web.dir}/lib/util.jar"
   basedir = "${build.dir}/classes"
   includes = "faxapp/util/**"
   excludes = "**/Test.class">
```

```
   <manifest>
      <attribute name = "Main-Class" value = "com.tutorialspoint.util.FaxUtil"/>
   </manifest>

</jar>
```

To execute the jar task, wrap it inside a target, most commonly, the build or package target, and execute them.

```
<target name="build-jar">
   <jar destfile="${web.dir}/lib/util.jar"
      basedir="${build.dir}/classes"
      includes="faxapp/util/**"
      excludes="**/Test.class">

      <manifest>
         <attribute name="Main-Class" value="com.tutorialspoint.util.FaxUtil"/>
      </manifest>

   </jar>
</target>
```

Running Ant on this file creates the util.jar file for us.

The following outcome is the result of running the Ant file:

```
C:\>ant build-jar
Buildfile: C:\build.xml

BUILD SUCCESSFUL
Total time: 1.3 seconds
```

The util.jar file is now placed in the output folder.