We have learnt the different aspects of Ant using the **Hello World** Fax web application in bits and pieces.

Now it is time to put everything together to create a full and complete build.xml file. Consider **build.properties** and **build.xml** files as follows:

build.properties

```
deploy.path = c:\tomcat6\webapps
```

build.xml

```
<?xml version = "1.0"?>

<project name = "fax" basedir = "." default = "usage">

   <property file = "build.properties"/>
   <property name = "src.dir" value = "src"/>
   <property name = "web.dir" value = "war"/>
   <property name = "javadoc.dir" value = "doc"/>
   <property name = "build.dir" value = "${web.dir}/WEB-INF/classes"/>
   <property name = "name" value = "fax"/>

   <path id = "master-classpath">
      <fileset dir = "${web.dir}/WEB-INF/lib">
         <include name = "*.jar"/>
      </fileset>
      <pathelement path = "${build.dir}"/>
   </path>

   <target name = "javadoc">
      <javadoc packagenames = "faxapp.*" sourcepath = "${src.dir}"
         destdir = "doc" version = "true" windowtitle = "Fax Application">

         <doctitle><![CDATA[<h1> =  Fax Application  = </h1>]]>
         </doctitle>

         <bottom><![CDATA[Copyright © 2011. All Rights Reserved.]]>
         </bottom>

         <group title = "util packages" packages = "faxapp.util.*"/>
         <group title = "web packages" packages = "faxapp.web.*"/>
         <group title = "data packages" packages = "faxapp.entity.*:faxapp.dao.*"/>
      </javadoc>
   </target>

   <target name = "usage">
      <echo message = ""/>
      <echo message = "${name} build file"/>
      <echo message = "-----------------------------------"/>
      <echo message = ""/>
      <echo message = "Available targets are:"/>
      <echo message = ""/>
      <echo message = "deploy    --> Deploy application as directory"/>
      <echo message = "deploywar --> Deploy application as a WAR file"/>
      <echo message = ""/>
   </target>

   <target name = "build" description = "Compile main source tree java files">
      <mkdir dir = "${build.dir}"/>

      <javac destdir = "${build.dir}" source = "1.5" target = "1.5" debug = "true"
```

```
         deprecation = "false" optimize = "false" failonerror = "true">

         <src path = "${src.dir}"/>
         <classpath refid = "master-classpath"/>

      </javac>
   </target>


   <target name = "deploy" depends = "build" description = "Deploy application">
      <copy todir = "${deploy.path}/${name}" preservelastmodified = "true">

         <fileset dir = "${web.dir}">
            <include name = "**/*.*"/>
         </fileset>

      </copy>
   </target>


   <target name = "deploywar" depends = "build" description = "Deploy application as a
WAR file">

      <war destfile = "${name}.war" webxml = "${web.dir}/WEB-INF/web.xml">
         <fileset dir = "${web.dir}">
            <include name = "**/*.*"/>
         </fileset>
      </war>

      <copy todir = "${deploy.path}" preservelastmodified = "true">
         <fileset dir = ".">
            <include name = "*.war"/>
         </fileset>
      </copy>

   </target>


   <target name = "clean" description = "Clean output directories">
      <delete>
         <fileset dir = "${build.dir}">
            <include name = "**/*.class"/>
         </fileset>
      </delete>
   </target>

</project>
```

In this example:

- We first declare the path to the webapps folder in Tomcat in the build properties file as the **deploy.path** variable.
- We also declare the source folder for the java files in the **src.dir** variable.
- Then we declare the source folder for the web files in the **web.dir** variable. **javadoc.dir** is the folder for storing the java documentation, and **build.dir** is the path for storing the build output files.
- Then we declare the name of the web application, which is **fax** in our case.
- We also define the master class path which contains the JAR files present in the WEB-INF/lib folder of the project.
- We also include the class files present in the **build.dir** in the master class path.
- The Javadoc target produces the javadoc required for the project and the usage target is used to print the common targets that are present in the build file.

The above example shows two deployment targets : **deploy** and **deploywar.**

The deploy target copies the files from the web directory to the deploy directory preserving the last modified date time stamp. This is useful when deploying to a server that supports hot deployment.

The clean target clears all the previously built files.

The deploywar target builds the war file and then copies the war file to the deploy directory of the application server.