从 Makefile 到 Buildfile:Java 工程利器 Ant

陈旭东（北方交通大学网络管理研究中心 100044）

摘要:本文通过一个典型的 Ant 工程文件,介绍如何使用 Ant 来 build 和配置 Java 工程。

关键词:Java 工程 XML Ant

Ant 是一种基于 Java 的 Java 工程自动 build 和配置工具,有些类似于 C/C++ 中的 make 。Makefile 目前存在一些问题,例如,如果 tab 前有一个空格或\ 后面有一个空格,make 命令就会出错,而且,make 依赖于操作系统命令。Ant 则不一样,使用 Ant 来 build 的纯 Java 工程是可以移植的,因为 Ant 本身就是使用 Java 语言编写的,而且 Ant 的 build 文件使用的是 XML。

本文通过一个典型的 Ant 工程文件,介绍如何使用 Ant 来 build 和配置 Java 工程。

1　准备工作

首先,需要下载和安装 jdk 和 ant。

Ant 是一个纯 Java 工具,要运行 ant,首先需要安装 Java 虚拟机 JVM。如果系统中尚未安装 JVM,可以从 http://java.sun.com/j2se/1.4.2/download.html 免费下载 j2sdk1.4.2 并安装。

Ant 可以从 http://jakarta.apache.org/ant/index.html 下载,我们可以直接下载其二进制版本 jakarta-ant-1.5.1-bin。安装时,只需解压缩到安装目录,如 windows 下的 C:\Ant 或 UNIX 下的 /usr/local/ant。

之后,需要设置一些环境变量,包括 PATH、JAVA_HOME 和 ANT_HOME。

如果是 Windows 系统,可以在 autoexec.bat 文件中增加如下几行:

```
set JAVA_HOME=<JAVA_HOME>
set ANT_HOME=<ANT_HOME>
set PATH=%PATH%;<JAVA_HOME>\bin;<ANT_HOME>\bin
```

其中,<JAVA_HOME>和<ANT_HOME>分别表示 jdk 和 Ant 的安装目录。要使设置生效,需要重新启动系统。对 WINDOWS2000/XP 系统,则可以直接在系统属性的环境变量中添加。

如果是 UNIX 或 LINUX,假如使用 Bash shell,可以在 $HOME/.bash_profile 文件(对于使用其他 shell,可以编辑对应的 profile 文件)中增加如下几行:

```
JAVA_HOME=<JAVA_HOME>
ANT_HOME=<ANT_HOME>
PATH=$PATH:<JAVA_HOME>/bin:<ANT_HOME>/bin
export JAVA_HOME ANT_HOME PATH
```

只需在命令提示符下输入 . $HOME/.bash_profile 就可以激活所定义的环境变量。

测试安装正确与否,可以在命令行输入 java 和 ant,看是否可以运行即可。

2　Ant 的工程文件

Ant 使用一个用 XML 编写的工程文件(buildfile)来确定其工作步骤。在没有指定任何参数时,Ant 会在当前目录下查询 build.xml 文件。如果找到了就用该文件作为工程文件。要想让 Ant 使用其他的工程文件,可以用参数 -buildfile file,这里 file 指定了你想使用的工程文件。如果你想在工程的任何位置调用 ant,可以使用 -find 选项,Ant 就会在文件系统目录中寻找工程文件。

下面是一个典型的 Java 工程,源代码存放在 src 目录,库文件(包括 JAR 文件)存放在 lib 目录,API 文档存放在 doc 目录,我们可以使用如下的 Ant 工程文件来 build 该 Java 工程:

```
<? xml version="1.0"? >
< project name = " typical " default = " all " basedir="."> 
<property name="name" value="typical" />
<property name="src" value="src" />
<property name="lib" value="lib" />
<property name="api" value="doc" />
<property name="tmp" value="tmp" />
<property name = "classpath" value = " ${lib}/${name}.jar" />
<property name="main" value="MyApp.Main" />
<target name="bin"
        description="编译源代码">
```

```
<javac srcdir="${src}"
       destdir="${tmp}"
       debug="on"
       deprecation="on" />
</target>
<target name="jar" depends="bin"
       description="生成 jar 文件 ">
<jar jarfile="${lib}/${name}.jar"
  basedir="${tmp}" />
</target>
<target name="run" depends="jar"
       description="运行应用程序">
<java classname="${main}"
      classpath="${classpath}" />
</target>
<target name="api"
       description="生成 API 文档">
<javadoc sourcepath="${src}"
        destdir="${api}"
        packagenames="MyApp.*" />
</target>
<target name="clean"
       description="清除临时文件">
<delete dir="${tmp}" />
<mkdir dir="${tmp}" />
</target>
<target name="all" depends="clean,jar,api" />
</project>
```

上面的语法看起来有点过于详细，如果使用 make 可以写出简短的多的工程文件，但是，Ant 的工程文件的优点是可读性强。开发人员无须过多了解 XML 的语法，也可以很容易读懂上述文件。下面对 Ant 的工程文件做一个比较详细的介绍。

Ant 工程文件的所有内容都包含在工程元素<project>中，工程元素有一个工程属性集。随后是一些目标段<target>，构成 Ant 的任务，包括编译源代码、生成 JAR 文件、运行应用程序、生成 API 文档、清除临时文件等。

2.1 工程的参数

工程的参数<property>位于工程元素<project>中，类似于环境变量。在工程内部和目标元素中的任何位置都可使用这些参数，其引用格式为 ${参数名}，例如上面文件中的${src}就代表在工程属性中定义的目录 src。需要注意的是，可以使用 Ant 的命令行参数来重新定义工程参数的值，其格式为 -D 属性名=值。例如，Ant -Dsrc=My_New_Src。工程参数一旦定义，就不能再改变它们的值。

2.2 目标与任务

目标<Target>元素定义了执行某种动作的指令集。例如上面文件中，目标 jar(包含属性 name="jar" 的<target>元素)包含一个<jar>任务，以完成从类文件生成 JAR 文件。

目标中可以引用其他目标。在目标 jar 中，就引用了目标 bin (depends="bin")，表示如果调用目标 jar，会首先调用目标 bin，除非目标 bin 已经执行过。

在运行 Ant 时，可以通过命令行传递目标参数，对于传递多个目标，中间以空格分隔。例如，ant jar，表示运行目标 jar。如果运行 Ant 时不指定目标，就会运行工程的默认目标，在上面的例子中就是目标 all。

3 Ant 工程中目标与任务的详细分析

我们已经知道，Ant 的工程文件是一个目标<target>的集合，它们规定了工程 build 和配置的各个步骤。下面，就上面例子文件中包含的目标作一个比较详细的描述，从而使读者可以尽快地自定义符合自己需要的工程文件。在我们的工程文件例子中，定义了五个基本目标：编译源代码、生成 jar 文件、运行应用程序、生成 API 文档和清除临时文件。

3.1 编译 Java 源代码文件

这个目标基本上是每个工程必备的目标之一。在我们的工程文件例子中，仅包含了一个 javac 任务，用于编译 Java 源代码文件，在 javac 任务中使用了如下属性：

srcdir="${src}"-- 源代码文件所在目录

destdir="${tmp}"-- 编译生成的类文件目录

debug="on"-- javac 的编译选项

deprecation="on"-- javac 的编译选项

如果编译时需要用到其他 Java 类库，那么就需要增加一个 classpath 属性，并使它等于对应的 JAR 文件或 classpath 目录。目录路径的写法可以是 Windows 格式(\)，也可以是 UNIX 格式(/)，多个目录之间可以使用冒号(:)或分号(;)来分隔，在不同的操作系统由 Ant 自动转换。例如，classpath = "lib/foo.jar:lib/bar.jar" 或 classpath = "libe --(右下线)---foo.jar;lib---(右下线)--bar.jar"都是可以的。

Javac 任务中使用的是 JDK 的编译器 javac，其本身是一个 Java 程序，使用的类库为 tools.jar 文件，位于 $JAVA_HOME/lib 目录。因此，classpath 中一定包含 tools.jar。

3.2 生成 Jar 文件

本目标的任务是从 Java 类和其他资源(如图片)来生成 JAR 文件。在我们的工程文件例子中,仅包含了一个 jar 任务,在 jar 任务中使用了如下属性:

jarfile="${lib}/${name}.jar"-- 生成的 JAR 文件

basedir="${tmp}"-- 类文件所在目录

如果需要包含其他文件,这时需要在任务中嵌入 fileset 元素来定义文件的集合。例如要包含一些存在于 img 目录下的图片文件(PNG 格式),就可使用如下的格式:

```
<target name="jar" depends="bin" description="生成 jar 文件">
  <jar jarfile="${lib}/${name}.jar"
       basedir="${tmp}">
    <fileset dir="img" includes="*.png" />
-- fileset 元素
  </jar>
</target>
```

fileset 元素中,dir 属性指明目录,includes 属性则定义了文件集的文件。字符 * 代表 0 到多个的任意字符,字符 ? 则表示一个任意字符,而 **/ 则表示所有子目录,例如<fileset dir="img" includes="**/*.png" /> 。另外,fileset 元素中还可以使用 excludes 属性来指定不包含的文件,例如,包含 img 目录下除了 *.jpg 的所有文件,可以写为:<fileset dir="img" excludes="**/*.dia" />。

默认情况下,某些特殊文件是不会包含在文件集<fileset>元素中的,如编辑程序产生的临时文件或版本控制产生的目录(如 **/CVS/*)。如果想要包含这些文件和目录,则<fileset>元素中需要增加一个属性:defaultexcludes="false"。

Ant 还可以处理 WAR 文件和 EAR 文件。WAR 任务需要定义<lib>、<classes>、<webinf>和<metainf>等嵌入元素来定义文件集<fileset>元素。EAR 任务则使用<metainf>元素和 appxml 属性。这方面的内容,在此不再叙述。

3.3 运行应用程序

我们使用 java 任务来运行 Java 应用程序。在我们的工程文件例子中,java 任务中使用了如下属性:

classname="${main}" -- 运行的类

classpath="${classpath}"-- 运行所需的类库

属性 classname 定义的类应包含有 main()方法以运行 Java 应用程序。属性 classpath 则包含运行该 Java 应用程序所需要的类库。也可以在<classpath>元素中嵌入<pathelement>或<fileset>子元素。例如,包括参数 classpath 中的路径、classes 目录、lib 目录下的所有 JAR 文件的语法为:

```
<java classname="${main}">
  <classpath>
    <pathelement path="${classpath}" />
-- 参数 classpath 中的路径
    <pathelement location="classes" /> --
classes 目录
    <fileset dir="lib" includes="*.jar" />
-- lib 目录下的所有 JAR 文件
  </classpath>
</java>
```

3.4 生成 API 文档

本目标的任务是使用 javadoc 来生成 API 文档。在我们的工程文件例子中,仅包含了一个 javadoc 任务,在 javadoc 任务中使用了如下属性:

sourcepath="${src}"-- Java 源代码目录

destdir="${api}"-- API 文档目录

packagenames="MyApp.*"-- 文档中的包

属性 packagenames 是文档中包(package)的列表,使用逗号分隔。上述属性将对 test、MyApp.foo、MyApp.foo.bar 包创建文档。

在本目标中,还可以使用 windowtitle、doctitle、header、footer、bottom、link 等属性来定义窗口和文档的标题、页眉、页脚、URL 链接等。

3.5 清除临时文件

本目标的任务是清除生成的类文件。在我们的工程文件例子中,包含了 delete 和 mkdir 两个任务,在 delete 和 mkdir 任务中都使用了属性:dir="${tmp}"。

为了避免将来编译的依赖性问题,因此,最好是清除生成的类文件。

4 结论

本文通过一个典型的 Ant 工程文件实例,介绍了 Ant 的使用。从 c/c++到 Java,从 Makefile 到 Buildfile,从 make 到 ant,您可以无需对 XML 有很深的了解,您也无需通读 Ant 文档,本文为开始 Ant 旅程打开了简捷之门。

参考文献

1 Apache Ant 1.5.1 Manual, http://jakarta.apache.org/ant/index.html

2 Java? 2 Platform Documentation , http://java.sun.com/j2se/1.4.2/docs/in dex.html