ant 脚本：使用节点、元素和属性、命令指令

<project default="all">

<property name="pro\_a" value="a value" />

<property name="pro\_b" value="b value" />

<path id="rt.path">

<pathelement location="${java.home}/jre/lib/rt.jar" />

</path>

<target name="all">

<javac srcdir=".">

<classpath refid="a.path" />

</javac>

</target>

</project>

注意：   
● 所有构建文件都要有<project>元素，而且至少有一个<target> 元素。   
● 对于<project> 元素的default 属性并不一定需要默认值。   
● 构建文件并不一定要被命名为build.xml。不过build.xml 是ant 要搜索的默认文件名。   
● 每个构建文件只能有一个<project> 元素。

在cmd窗口中进入到当前build.xml目录

ant   
在当前目录下的build.xml运行Ant，执行缺省的target。

ant -buildfile build-test.xml   
在当前目录下的build-test.xml运行Ant，执行缺省的target。

ant -buildfile build-test.xml clean   
在当前目录下的build-test.xml运行Ant，执行一个叫做clean的target。

ant -buildfile build-test.xml -Dbuild=build/classes clean   
在当前目录下的build-test.xml运行Ant，执行一个叫做clean的target，并设定build属性的值为build/classes。

1, project 节点元素

project 元素是 Ant 构件文件的根元素， Ant 构件文件至少应该包含一个 project 元素，否则会发生错误。在每个 project 元素下，可包含多个 target 元素。接下来向读者展示一下 project 元素的各属性。   
● name 属性：用于指定 project 元素的名称。   
● default 属性：用于指定 project 默认执行时所执行的 target 的名称。   
● basedir 属性：用于指定基路径的位置。该属性没有指定时，使用 Ant 的构件文件的附目录作为基准目录。

<?xml version="1.0" ?>

<project name="ant-project" default="print-dir" basedir=".">

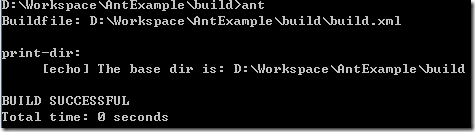
<target name="print-dir">

<echo message="The base dir is: ${basedir}" />

</target>

</project>

从上例可以看出，在这里定义了default 属性的值为print-dir，即当运行ant 命令时，如果没有指明执行的target，则将执行默认的target（print-dir）。此外，还定义了basedir 属性的值为 “.” ，.表示当前目录，进入当前目录后运行ant 命令，得一下结果：



2、target节点元素

target为ant的基本执行单元或是任务，它可以包含一个或多个具体的单元/任务。多个target 可以存在相互依赖关系。它有如下属性：   
● name 属性：指定 target 元素的名称，这个属性在一个 project 元素中是唯一的。我们可以通过指定 target 元素的名称来指定某个 target 。   
● depends 属性：用于描述 target 之间的依赖关系，若与多个 target 存在依赖关系时，需要以“,”间隔。 Ant 会依照 depends 属性中 target 出现的顺序依次执行每个 target ，被依赖的target 会先执行。   
● if 属性：用于验证指定的属性是存在，若不存在，所在 target 将不会被执行。   
● unless 属性：该属性的功能与 if 属性的功能正好相反，它也用于验证指定的属性是否存在，若不存在，所在 target 将会被执行。   
● description 属性：该属性是关于 target 功能的简短描述和说明。   
示例：

<?xml version="1.0" ?>

<project name="ant-target" default="print">

<target name="version" if="ant.java.version">

<echo message="Java Version: ${ant.java.version}" />

</target>

<target name="print" depends="version" unless="docs">

<description>

a depend example!

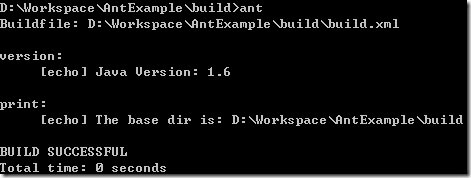
</description>

<echo message="The base dir is: ${basedir}" />

</target>

</project>

从以下结果后可以看到，我们运行的是名为 print的target ，由于它依赖于version这个target任务，所以 version将首先被执行，同时因为系统配置了JDK，所以 ant.java.version 属性存在，执行了version，输出信息："[echo] Java Version: 1.6 "，version执行完毕后，接着执行 print，因为docs不存在，而unless属性是在不存在时进入所在target 的，由此可知 print得以执行，输出信息："[echo] The base dir is:D:\Workspace\AntExample\build"。



3、property属性节点元素

property元素可看作参量或者参数的定义，project 的属性可以通过 property 元素来设定，也可在 Ant 之外设定。若要在外部引入某文件，例如 build.properties 文件，可以通过如下内容将其引：   
<property file="build.properties"/>   
property 元素可用作 task 的属性值。在 task 中是通过将属性名放在${属性名}之间，并放在 task 属性值的位置来实现的。   
Ant 提供了一些内置的属性，它能得到的系统属性的列表与 Java 文档中 System.getProperties() 方法得到的属性一致，这些系统属性可参考 sun 网站的说明。同时， Ant 还提供了一些它自己的内置属性，如下：   
basedir： project 基目录的绝对路径；     
ant.file： buildfile的绝对路径，上例中ant.file值为D:\Workspace\AntExample\build；   
ant.version： Ant 的版本信息，本文为1.8.1 ；   
ant.project.name： 当前指定的project的名字，即前文说到的project的name属性值；   
ant.java.version： Ant 检测到的JDK版本，本文为 1.6 。

举例说明如下：

<project name="ant-project" default="example">

<property name="name" value="jojo" />

<property name="age" value="25" />

<target name="example">

<echo message="name: ${name}, age: ${age}" />

</target>

</project>

上例中用户设置了名为name 和age的两个属性，这两个属性设置后，在下文中可以通过 ${name} 和 ${age} 分别取得这两个属性值。

4、copy命令

copy主要用来对文件和目录的复制功能。举例如下：   
● 复制单个文件：   
<copy file="old.txt" tofile="new.txt"/>

● 对文件目录进行复制：   
<copy todir="../dest\_dir">   
    <fileset dir="src\_dir"/>   
</copy>

● 将文件复制到另外的目录：   
<copy file="src.txt" todir="c:/base"/>

5、delete命令

对文件或目录进行删除，举例如下：

● 删除某个文件：      
<delete file="/res/image/cat.jpg"/>

● 删除某个目录：      
<delete dir="/res/image"/>

● 删除所有的jar文件或空目录：   
<delete includeEmptyDirs="true">   
       <fileset dir="." includes="\*\*/\*.jar"/>   
</delete>

6、 mkdir 命令

创建目录。   
<mkdir dir="/home/philander/build/classes"/>

7、 move 命令

移动文件或目录，举例如下：   
● 移动单个文件：   
<move file="sourcefile" tofile=”destfile”/>

● 移动单个文件到另一个目录：   
<move file="sourcefile" todir=”movedir”/>

● 移动某个目录到另一个目录：   
<move todir="newdir">   
    <fileset dir="olddir"/>   
</move>

8、echo 命令

该任务的作用是根据日志或监控器的级别输出信息。它包括 message 、 file 、 append 和 level 四个属性，举例如下   
<echo message="ant message" file="/logs/ant.log" append="true">

9、jar 标签节点元素

该标签用来生成一个JAR文件，其属性如下。   
● destfile表示JAR文件名。   
● basedir表示被归档的文件名。   
● includes表示别归档的文件模式。   
● exchudes表示被排除的文件模式。

● compress表示是否压缩。

示例：

<jar destfile="${webRoot}/${ash\_jar}" level="9" compress="true" encoding="utf-8" basedir="${dest}">

<manifest>

<attribute name="Implementation-Version" value="Version: 2.2"/>

</manifest>

</jar>

上面的mainfest是jar包中的MEAT-INF中的MANIFEST.MF中的文件内容

同样打包操作的的还有war、tgz，已经解压操作uzip

<!-- 创建zip -->

<zip basedir="${basedir}\classes" zipfile="temp\output.zip"/>

<!-- 创建tgz -->

<gzip src="classes\\*\*\\*.class" zipfile="output.class.gz"/>

<!-- 解压zip -->

<unzip src="output.class.gz" dest="extractDir"/>

<!-- 建立war包 -->

<war destfile="${webRoot}/ash.war" basedir="${basedir}/web" webxml="${basedir}/web/WEB-INF/web.xml">

<exclude name="WEB-INF/classes/\*\*"/>

<exclude name="WEB-INF/lib/\*\*"/>

<exclude name="WEB-INF/work/\_jsp/\*\*"/>

<lib dir="${lib.dir}" includes="\*\*/\*.jar, \*\*/\*.so, \*\*/\*.dll">

<exclude name="${webRoot}\${helloworld\_jar}"/>

</lib>

<lib file="${webRoot}/${helloworld\_jar}"/>

<classes dir="${dest}" includes="\*\*/\*.xml, \*\*/\*.properites, \*\*/\*.xsd"> </classes>

</war>

10、javac 标签节点元素

该标签用于编译一个或一组java文件，其属性如下。   
● srcdir表示源程序的目录。   
● destdir表示class文件的输出目录。   
● include表示被编译的文件的模式。   
● excludes表示被排除的文件的模式。   
● classpath表示所使用的类路径。   
● debug表示包含的调试信息。   
● optimize表示是否使用优化。   
● verbose 表示提供详细的输出信息。   
● fileonerror表示当碰到错误就自动停止。

示例

<javac srcdir="${src}" destdir="${dest}"/>

<!-- 设置jvm内存

<javac srcdir="src" fork="true"/>

<javac srcdir="src" fork="true" executable="d:\sdk141\bin\javac"

memoryMaximumSize="128m"/>

-->

11、java 标签节点元素

该标签用来执行编译生成的.class文件，其属性如下。   
● classname 表示将执行的类名。   
● jar表示包含该类的JAR文件名。   
● classpath所表示用到的类路径。   
● fork表示在一个新的虚拟机中运行该类。   
● failonerror表示当出现错误时自动停止。   
● output 表示输出文件。   
● append表示追加或者覆盖默认文件。

示例

<java classname="com.hoo.test.HelloWorld" classpath="${hello\_jar}"/>

12、arg 数据参数元素

由Ant构建文件调用的程序，可以通过<arg>元素向其传递命令行参数，如apply,exec和java任务均可接受嵌套<arg>元素，可以为各自的过程调用指定参数。以下是<arg>的所有属性。   
● values 是一个命令参数。如果参数中有空格，但又想将它作为单独一个值，则使用此属性。   
● file 表示一个参数的文件名。在构建文件中，此文件名相对于当前的工作目录。   
● line 表示用空格分隔的多个参数列表。   
● 表示路径，一个作为单个命令行变量的path-like的字符串；或作为分隔符，Ant会将其转变为特定平台的分隔符。   
● pathref 引用的path（使用path元素节点定义path）的id   
● prefix 前缀   
● suffix 后缀

例子   
<arg value="-l -a"/>   
是一个含有空格的单个的命令行变量。   
<arg line="-l -a"/>   
是两个空格分隔的命令行变量。   
<arg path="/dir;/dir2:\dir3"/>   
是一个命令行变量，其值在DOS系统上为\dir;\dir2;\dir3；在Unix系统上为/dir:/dir2:/dir3 。

13、ervironment 类型

由Ant构建文件调用的外部命令或程序，<env>元素制定了哪些环境变量要传递给正在执行的系统命令，<env>元素可以接受以下属性。   
● file表示环境变量值的文件名。此文件名要被转换位一个绝对路径。   
● path表示环境变量的路径。Ant会将它转换为一个本地约定。   
● value 表示环境变量的一个直接变量。   
● key 表示环境变量名。   
注意 file path 或 value只能取一个。

14、filelist 文件集合列表

filelist 是一个支持命名的文件列表的数据类型，包含在一个filelist类型中的文件不一定是存在的文件。以下是其所有的属性。   
● dir是用于计算绝对文件名的目录。   
● files 是用逗号分隔的文件名列表。   
● refid 是对某处定义的一个<filelist>的引用。   
注意 dir 和 files 都是必要的，除非指定了refid(这种情况下，dir和files都不允许使用)。

示例

<filelist id="docfiles" dir="${doc.src}" files="foo.xml,bar.xml"/>

文件集合 ${doc.src}/foo.xml和${doc.src}/bar.xml. 这些文件也许还是不存在的文件.

<filelist id="docfiles" dir="${doc.src}" files="foo.xml bar.xml"/>

<filelist refid="docfiles"/>

<filelist id="docfiles" dir="${doc.src}">

<file name="foo.xml"/>

<file name="bar.xml"/>

</filelist>

15、fileset 文件类型

fileset 数据类型定义了一组文件，并通常表示为<fileset>元素。不过，许多ant任务构建成了隐式的fileset,这说明他们支持所有的fileset属性和嵌套元素。以下为fileset 的属性列表。   
● dir表示fileset 的基目录。   
● casesensitive的值如果为false，那么匹配文件名时，fileset不是区分大小写的，其默认值为true.   
● defaultexcludes 用来确定是否使用默认的排除模式，默认为true。   
● excludes 是用逗号分隔的需要派出的文件模式列表。   
● excludesfile 表示每行包含一个排除模式的文件的文件名。   
● includes 是用逗号分隔的，需要包含的文件模式列表。   
● includesfile 表示每行包括一个包含模式的文件名。

示例

<fileset id="lib.runtime" dir="${lib.path}/runtime">

<include name="\*\*/\*.jar"/>

<include name="\*\*/\*.so"/>

<include name="\*\*/\*.dll"/>

</fileset>

<fileset id="lib.container" dir="${lib.path}/container">

<include name="\*\*/\*.jar"/>

</fileset>

<fileset id="lib.extras" dir="${lib.path}">

<include name="test/\*\*/\*.jar"/>

</fileset>

16、patternset 类型

fileset 是对文件的分组，而patternset是对模式的分组，他们是紧密相关的概念。

<patternset>支持4个属性：includes、excludex、includexfile、excludesfile，这些与fileset相同。

patternset 还允许以下嵌套元素：include,exclude,includefile 和 excludesfile.

示例

<!-- 黑白名单 -->

<patternset id="non.test.sources">

<include name="\*\*/\*.java"/>

<!-- 文件名包含Test的排除 -->

<exclude name="\*\*/\*Test\*"/>

</patternset>

<patternset id="sources">

<include name="std/\*\*/\*.java"/>

<!-- 判断条件 存在professional就引入 -->

<include name="prof/\*\*/\*.java" if="professional"/>

<exclude name="\*\*/\*Test\*"/>

</patternset>

<!-- 一组文件 -->

<patternset includesfile="some-file"/>

<patternset>

<includesfile name="some-file"/>

<patternset/>

<patternset>

<includesfile name="some-file"/>

<includesfile name="${some-other-file}" if="some-other-file"/>

<patternset/>

17、filterset 类型

filterset定义了一组过滤器，这些过滤器将在文件移动或复制时完成文件的文本替换。   
主要属性如下：   
● begintoken 表示嵌套过滤器所搜索的记号，这是标识其开始的字符串。   
● endtoken 表示嵌套过滤器所搜索的记号这是标识其结束的字符串。   
● id 是过滤器的唯一标志符。   
● refid 是对构建文件中某处定义一个过滤器的引用。

示例

<!-- 将目标文件build.dir目录中的version.txt文件内容中的@DATE@替换成TODAY当前日期的值，并把替换后的文件存放在dist.dir目录中 -->

<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">

<filterset>

<filter token="DATE" value="${TODAY}"/>

</filterset>

</copy>

<!-- 自定义变量的格式 -->

<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">

<!-- 从version.txt中的%位置开始搜索，到\*位置结束，进行替换内容中的@DATE@替换成TODAY当前日期的值-->

<filterset begintoken="%" endtoken="\*">

<filter token="DATE" value="${TODAY}"/>

</filterset>

</copy>

<!-- 使用外部的过滤定义文件 -->

<copy toDir="${dist.dir}/docs">

<fileset dir="${build.dir}/docs">

<include name="\*\*/\*.html">

</fileset>

<filterset begintoken="%" endtoken="\*">

<!-- 过来文件从外部引入，过来的属性和值配置在dist.properties文件中 -->

<filtersfile file="${user.dir}/dist.properties"/>

</filterset>

</copy>

<!-- 使用引用方式，重复利用过滤集 -->

<filterset id="myFilterSet" begintoken="%" endtoken="\*">

<filter token="DATE" value="${TODAY}"/>

</filterset>

<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">

<filterset refid="myFilterSet"/>

</copy>

18、path类型

path元素用来表示一个类路径，不过它还可以用于表示其他的路径。在用作几个属性时，路经中的各项用分号或冒号隔开。在构建的时候，此分隔符将代替当前平台中所有的路径分隔符，其拥有的属性如下。   
● location 表示一个文件或目录。Ant在内部将此扩展为一个绝对路径。   
● refid 是对当前构建文件中某处定义的一个path的引用。   
● path表示一个文件或路径名列表。

示例

<path id="buildpath">

<fileset refid="lib.runtime"/>

<fileset refid="lib.container"/>

<fileset refid="lib.extras"/>

</path>

<path id="src.paths">

<fileset id="srcs" dir=".">

<include name="src/\*\*/\*.java"/>

</fileset>

</path>

语法元素说明如下：   
-help   
显示描述ant 命令及其选项的帮助信息   
-projecthelp   
显 示包含在构建文件中的、所有用户编写的帮助文档。即为各个<target>中description 属性的文本，以及包含在<description>元素中的任何文本。将有description 属性的目标列为主目标（Main target），没有此属性的目标则列为子目标（Subtarget）。   
-version   
要求ant 显示其版本信息，然后退出。   
-quiet   
抑制并非由构建文件中的echo 任务所产生的大多数消息。   
-verbose   
显示构建过程中每个操作的详细消息。此选项与-debug 选项只能选其一。   
-debug   
显示Ant 和任务开发人员已经标志为调试消息的消息。此选项与-verbose 只能选其一。   
-emacs   
对日志消息进行格式化，使它们能够很容易地由Emacs 的shell 模式（shellmode）所解析；也就是说，打印任务事件，但并不缩排，在其之前也没有[taskname]。   
-logfile filename   
将日志输出重定向到指定文件。   
-logger classname   
指定一个类来处理Ant 的日志记录。所指定的类必须实现了org.apache.tools.ant.BuildLogger 接口。   
-listener classname   
为Ant 声明一个监听类，并增加到其监听者列表中。在Ant与IDE或其他Java程序集成时，此选项非常有用。可以阅读第六章以了解有关监听者的更多信息。必须将所指定的监听类编写为可以处理Ant 的构建消息接发。   
-buildfile filename   
指定Ant 需要处理的构建文件。默认的构建文件为build.xml。   
-Dproperty=value   
在命令行上定义一个特性名－值对。   
-find filename   
指 定Ant 应当处理的构建文件。与-buildfile 选项不同，如果所指定文件在当前目录中未找到，-find 就要求Ant 在其父目录中再进行搜索。这种搜索会继续在其祖先目录中进行，直至达到文件系统的根为止，在此如果文件还未找到，则构建失败。   
-atuoproxy jdk1.5以上的可以使用代理设置   
-nouserlib 运行ant时不使用用户lib中的jar包   
-nice 设计主线程优先级   
-logfile 使用指定的log日志   
-noinput 不允许交互输入   
-keep-going, -k 执行不依赖于所有目标   
-propertyfile 加载所有属性配置文件 -d 属性文件优先