Java_Ant 详解

文章分类: Java 编程

本文转自：百度空间

Java_Ant 详解

1,什么是 ant
ant 是构建工具
2,什么是构建
概念到处可查到，形象来说，你要把代码从某个地方拿来，编译，再拷贝到某个地方去等等操作，当然不仅与此，但是主要用来干这个
3,ant 的好处
跨平台　--因为 ant 是使用 java 实现的，所以它跨平台
使用简单--与 ant 的兄弟 make 比起来
语法清晰--同样是和 make 相比
功能强大--ant 能做的事情很多，可能你用了很久，你仍然不知道它能有多少功能。当你自己开发一些 ant 插件的时候，你会发现它更多的功能。
4,ant 的兄弟 make
ant 做的很多事情，大部分是曾经有一个叫 make 的所做的，不过对象不同，make 更多应用于 c/c++,ant 更多应用于 Java。当然这不是一定的，但大部分人如此。
一,构建 ant 环境
要使用 ant 首先要构建一个 ant 环境，步骤很简单：
1),安装 jdk，设置 JAVA_HOME ,PATH ,CLASS_PATH(这些应该是看这篇文章的人应该知道的)
2),下载 ant 地址 http://www.apache.org/找一个你喜欢的版本,或者干脆最新的版本
3),解压 ant 你得到的是一个压缩包，解压缩它，并把它放在一个尽量简单的目录，例如 D:\ant-1.6 虽然你不一 定要这么做，但这么做是有好处的。
4),设置 ANT_HOME PATH 中添加 ANT_HOME 目录下的 bin 目录
5),测试一下你的设置,开始-->运行-->cmd 进入命令行-->键入 ant 回车,如果看到
Buildfile: build.xml does not exist!
Build failed
那么恭喜你你已经完成 ant 的设置
二,体验 ant
就像每个语言都有 HelloWorld 一样，一个最简单的应用能让人感受一下 Ant
1,首先你要知道你要干什么，我现在想做的事情是：
编写一些程序
编译它们
把它打包成 jar 包
把他们放在应该放置的地方
运行它们
这里为了简单起见只写一个程序，就是 HelloWorld.java 程序代码如下：

```
package test.ant;
public class HelloWorld{
public static void main(String[] args){
   System.out.println("Hello world1");
}
};
```
2，为了达到上边的目的，你可以手动的用 javac 、copy 、jar、java 来完成，但是考虑一下如果你有成百上千个类，在多次调试，部署的时候，一次次的 javac 、copy、jar、
java 那将是一份辛苦的工作。现在看看 ant 怎么优雅的完成它们。
要运行 ant 需要有一个 build.xml 虽然不一定要叫这个名字,但是建议你这么做
下边就是一个完整的 build.xml，然后我们来详细的解释每一句
```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="HelloWorld" default="run" basedir=".">
<property name="src" value="src"/>
<property name="dest" value="classes"/>
<property name="hello_jar" value="hello1.jar"/>
<target name="init">
   <mkdir dir="${dest}"/>
</target>
<target name="compile" depends="init">
   <javac srcdir="${src}" destdir="${dest}"/>
</target>
<target name="build" depends="compile">
   <jar jarfile="${hello_jar}" basedir="${dest}"/>
</target>
<target name="run" depends="build">
   <java classname="test.ant.HelloWorld" classpath="${hello_jar}"/>
</target>
<target name="clean">
   <delete dir="${dest}" />
   <delete file="${hello_jar}" />
</target>
<target name="rerun" depends="clean,run">
   <ant target="clean" />
   <ant target="run" />
</target>
</project>
```
解释：
```
<?xml version="1.0" encoding="UTF-8" ?>
```
build.xml 中的第一句话，没有实际的意义
```
<project name="HelloWorld" default="run" basedir=".">
</project>
```
ant 的所有内容必须包含在这个里边，name 是你给它取的名字，basedir 故名思

意就是工作的根目录 .代表当前目录。default 代表默认要做的事情。
<property name="src" value="src"/>
类似程序中的变量，为什么这么做想一下变量的作用
<target name="compile" depends="init">
  <javac srcdir="${src}" destdir="${dest}"/>
</target>
把你想做的每一件事情写成一个 target ，它有一个名字，depends 是它所依赖的 target，在执行这个 target 例如这里的 compile 之前 ant 会先检查 init 是否曾经被执行过，如果执行
过则直接直接执行 compile，如果没有则会先执行它依赖的 target 例如这里的 init，然后在执行这个 target
如我们的计划
编译：
<target name="compile" depends="init">
<javac srcdir="${src}" destdir="${dest}"/>
</target>
做 jar 包:
<target name="build" depends="compile">
<jar jarfile="${hello_jar}" basedir="${dest}"/>
</target>
运行：
<target name="run" depends="build">
<java classname="test.ant.HelloWorld" classpath="${hello_jar}"/>
</target>
为了不用拷贝，我们可以在最开始定义好目标文件夹，这样 ant 直接把结果就放在目标文件夹中了
新建文件夹:
<target name="init">
<mkdir dir="${dest}"/>
</target>
为了更多一点的功能体现，又加入了两个 target
删除生成的文件
<target name="clean">
<delete dir="${dest}" />
<delete file="${hello_jar}" />
</target>
再次运行，这里显示了如何在一个 target 里边调用其他的 target
<target name="rerun" depends="clean,run">
<ant target="clean" />
<ant target="run" />
</target>
好了，解释完成了，下边检验一下你的 ant 吧
新建一个 src 的文件夹，然后把 HelloWorld.java 按照包目录放进去
做好 build.xml 文件

在命令行下键入 ant ,你会发现一个个任务都完成了。每次更改完代码只需要再次键入 ant
有的时候我们可能并不想运行程序，只想执行这些步骤中的某一两个步骤，例如我只想重新部署而不想运行，键入
ant build
ant 中的每一个任务都可以这样调用 ant + target name
好了，这样一个简单的 ant 任务完成了。

一,什么时候使用 ant
也许你听到别人说起 ant，一时冲动准备学习一下 ant，当你看完了上边的第一个实例，也许你感觉 ant 真好，也许你感觉 ant 不过如此，得出这些结论都不能说错，虽然 ant 很好用，
但并不是在任何情况下都是最好的选择，例如 windows 上有更多更简单，更容易使用的工具，比如 eclipse+myeclipse eclipse+wtp 等等，无论是编译，部署，运行使用起来比 ant 更
容易，方便但有些情况则是 ant 发挥的好地方：
1,服务器上部署的时候
当你的程序开发完成，部署人员要部署在服务器上的时候，总不能因为因为安装一个程序就配置一个 eclipse+myeclipse 吧，ant 在这个时候是个很好的选择，因为它小巧，容易配
置，你带着你写好的 build.xml 到任何一台服务器上，只需要做简单的修改（一些设定，例如目录），然后一两个命令完成，这难道不是一件美好的事情吗。
2,linux 上，很多时候是这样的，程序开发是在 windows 下，但是程序要在 linux 或者 unix 上运行，在 linux 或者
在 unix(特别是 unix 上)部署是个麻烦的事情，这个时候 ant 的特点又出来了，因为 ant 是跨平台的，你在 build.xml 可以在大多数操作系统上使用，基本不需要修改。
3,当服务器维护者不懂编程的时候
很多人都有过这样的经历，使用你们程序的人，并不懂得写程序。你得程序因为版本更新，因为修正 bug 需要一次又一次得重新部署。这个时候你会发现教一个人是如此得困难。但
是有 ant 后，你只需要告诉他，输入 ant xxx 等一两个命令，一切 ok.
以上是我遇到得一些情况。
看完以上得情况，好好考虑一下，你是否需要使用 ant，如果是继续。

进一步学习一个稍微复杂一点点的 ant
在实际的工作过程中可能会出现以下一些情况，一个项目分成很多个模块，每个小组或者部门负责一个模块，为了测试，他们自己写了一个 build.xml,而你负责把这些模块组合到
一起使用，写一个 build.xml
这个时候你有两种选择：
1,自己重新写一个 build.xml ，这将是一个麻烦的事情
2,尽量利用他们已经写好的 build.xml，减少自己的工作

举个例子：
假设你下边有三个小组，每个小组负责一个部分，他们分别有一个 src 和一个写好的 build.xml
这个时候你拿到他们的 src，你需要做的是建立三个文件夹 src1 ,src2, src3 分别把他们的 src 和 build.xml 放进去，然后写一个 build.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="main" default="build" basedir=".">
<property name="bin" value="${basedir}\bin" />
<property name="src1" value="${basedir}\src1" />
<property name="src2" value="${basedir}\src2" />
<property name="src3" value="${basedir}\src3" />
<target name="init">
   <mkdir dir="${bin}" />
</target>
<target name="run">
   <ant dir="${src1}" target="run" />
   <ant dir="${src2}" target="run" />
   <ant dir="${src3}" target="run" />
</target>
<target name="clean">
   <ant dir="${src1}" target="clean" />
   <ant dir="${src2}" target="clean" />
   <ant dir="${src3}" target="clean" />
</target>
<target name="build" depends="init,call">
   <copy todir="${bin}">
    <fileset dir="${src1}">
     <include name="*.jar" />
    </fileset>
    <fileset dir="${src2}">
     <include name="*.jar" />
    </fileset>
    <fileset dir="${src3}">
     <include name="*.jar" />
    </fileset>
   </copy>
</target>
<target name="rebuild" depends="build,clean">
   <ant target="clean" />
   <ant target="build" />
</target>
</project>
```

ok 你的任务完成了。

ok,上边你完成了任务，但是你是否有些感触呢，在那些 build.xml 中，大多数是重复的,而且更改一次目录需要更改不少东西。是否能让工作做的更好一点呢，答案是肯定的。
引入两个东西：
1,propery
2,xml include
这两个东西都有一个功能，就是能把 build.xml 中<propery />中的内容分离出来，共同使用
除此之外它们各有特点：
propery 的特点是维护简单,只需要简单的键值对,因为并不是所有人都喜欢 xml 的格式
xml include 的特点是不单可以提取出属性来，连 target 也可以。
还是以前的例子：
例如我们想把 src1 src2 src3 这三个属性从 xml 中提出来，可以新建一个文件叫 all.properties
里边的内容

```
src1=D:\\study\\ant\\src1
src2=D:\\study\\ant\\src2
src3=D:\\study\\ant\\src3
```

然后你的 build.xml 文件可以这样写，别人只需要更改配置文件，而不许要更改你的 build.xml 文件了

```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="main" default="build" basedir=".">
<property file="all.properties" />
<property name="bin" value="${basedir}\bin" />
<target name="init">
   <mkdir dir="${bin}" />
</target>
<target name="run">
   <ant dir="${src1}" target="run" />
   <ant dir="${src2}" target="run" />
   <ant dir="${src3}" target="run" />
</target>
<target name="clean">
   <ant dir="${src1}" target="clean" />
   <ant dir="${src2}" target="clean" />
   <ant dir="${src3}" target="clean" />
</target>
<target name="build" depends="init,call">
   <copy todir="${bin}">
    <fileset dir="${src1}">
     <include name="*.jar" />
    </fileset>
    <fileset dir="${src2}">
```

```
     <include name="*.jar" />
    </fileset>
    <fileset dir="${src3}">
     <include name="*.jar" />
    </fileset>
   </copy>
</target>
<target name="rebuild" depends="build,clean">
   <ant target="clean" />
   <ant target="build" />
</target>
<target name="test">
   <ant dir="${src1}" target="test" />
   <ant dir="${src2}" target="test" />
   <ant dir="${src3}" target="test" />
</target>
</project>
```

如果你自己看的话你会看到这样一个 target

```
<target name="test">
<ant dir="${src1}" target="test" />
<ant dir="${src2}" target="test" />
<ant dir="${src3}" target="test" />
</target>
```

有的时候你想给每个小组的 build.xml 加入几个 target，一种做法是每个里边写，然后在这里调用

但是有一种更好的方法。

你可以写一个 include.xml 文件，内容如下

```
<?xml version="1.0" encoding="UTF-8" ?>
<property name="src" value="src"/>
<property name="dest" value="classes"/>
<target name="test" >
<ant target="run" />
</target>
```

然后更改你三个小组的 build.xml 文件,每个里边加入如下内容

```
<!--include a xml file ,it can be common propery ,can be also a target   -->
<!DOCTYPE project [
<!ENTITY share-variable SYSTEM "file:../include.xml">
]>
&share-variable;
```

变成如下的样子

这个时候，你只要在 include.xml 添加 propery， 添加 target，三个 build.xml 会同时添加这些 propery 和 target

而且不会让三个组的 build.xml 变得更复杂。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--include a xml file ,it can be common propery ,can be also a
target   -->
<!DOCTYPE project [
<!ENTITY share-variable SYSTEM "file:../include.xml">
]>
<project name="HelloWorld" default="run" basedir=".">
<!--use the include   -->
&share-variable;
<!--defined the property-->
<!--via include
<property name="src" value="src"/>
<property name="dest" value="classes"/>
-->
<property name="hello_jar" value="hello1.jar"/>
<!--define the op-->
<target name="init">
   <mkdir dir="${dest}"/>
</target>
<target name="compile" depends="init">
   <javac srcdir="${src}" destdir="${dest}"/>
</target>
<target name="build" depends="compile">
   <jar jarfile="${hello_jar}" basedir="${dest}"/>
</target>
<target name="run" depends="build">
   <java classname="test.ant.HelloWorld" classpath="${hello_jar}"/>
</target>
<target name="clean">
   <delete dir="${dest}" />
   <delete file="${hello_jar}" />
</target>
<target name="rerun" depends="clean,run">
   <ant target="clean" />
   <ant target="run" />
</target>
</project>
```

掌握了上边的那些内容之后，你就知道如何去写一个好的 ant，但是你会发现当你真的想去做的时候，你不能马上作出好的 build.xml，因为你知道太少的 ant 的默认提供的命令.这
个时候如果你想完成任务，并提高自己，有很多办法：
1,很多开源的程序都带有 build.xml，看看它们如何写的
2,ant 的 document，里边详细列写了 ant 的各种默认命令，及其丰富

3,google，永远不要忘记它
ok,在这之后随着你写的 ant build 越来越多，你知道的命令就越多，ant 在你的手里也就越来越强大了。
这个是一个慢慢积累的过程。

ant 的例子很好找，各种开源框架都会带有一个 build.xml 仔细看看，会有很大收获
另外一个经常会用到的，但是在开源框架的 build.xml 一般没有的是 cvs
如果使用的是远程的 cvs，可以这样使用

```
<xml version="1.0" encoding="utf-8"?>
<project>
      <property name="cvsroot"
value=":pserver:wang:@192.168.1.2:/cvsroot"/>
      <property name="basedir" value="/tmp/testant/"/>
      <property name="cvs.password" value="wang"/>
      <property name="cvs.passfile" value="${basedir}/ant.cvspass"/>
      <target name="initpass">
              <cvspass cvsroot="${cvsroot}" password="${cvs.password}"
passfile="${cvs.passfile}"/>
      </target>
      <target name="checkout" depends="initpass">
              <cvs cvsroot="${cvsroot}" command="checkout"
cvsrsh="ssh" package="myproject" dest="${basedir}"
               passfile="${cvs.passfile}"/>
       </target>
</project>
```

在 eclipse 里边先天支持 ant，所以你可以在 eclipse 里边直接写 build.xml
因为 eclipse 提供了提示功能，自动补充功能，它能让你事半功倍。
使用方法，只需要建立一个工程，然后建立一个叫 build.xml 的文件。然后就可以在里边写你的 ant build 了
但是时刻记住 http://www.apache.org/永远能找到你需要的东西