# Ant 入门学习教程及实例

1. Ant应用场景，优点
2. Ant在PC上的部署
3. Ant详细解析----属性的介绍及使用
4. Android打包实例

**属性的介绍及使用**

project

target 继承关系

antcall

date type

arg参数的形式 p85

available

time

exec

condition

## 一个简单的Ant例子

Apache Ant，是一个将软件编译、测试、部署等步骤联系在一起加以自动化的一个工具，大多用于Java环境中的软件开发。

本人使用Ant的场景是拿它在后台执行编译、打包Android工程的操作，在Ant工作期间，仍可正常使用Ecelipse，避免痛苦的等待。

在使用之前应先[下载Ant](http://ant.apache.org/bindownload.cgi)并部署执行环境，细节上来说就是个添加系统环境变量的过程，具体可参考：[ant安装、环境变量配置及验证](http://www.cnblogs.com/yuzhongwusan/archive/2013/03/26/2982411.html)。

下面直接给出一个Ant的例子：

代码1.1 Hello World

<?xml version="1.0" encoding="UTF-8"?>

<project name="Hello\_World" default="Echo" basedir=".">

<target name="Echo" description="Just print 'Hello world'''.">

<echo message="Hello world! I'm ant."></echo>

</target>

</project>

Ant并没有定义它自己的语法，它的执行文件用XML编写。

以上代码存储在文件build\_01\_hello\_world.xml中。执行该文件则需命令：

ant –f build\_01\_hello\_world.xml，运行效果如下：

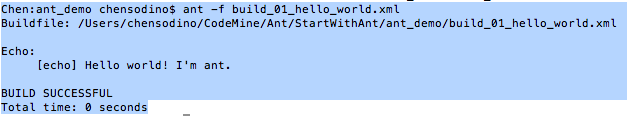


图1.1 Hello World

执行命令后，第一行显示编译的文件绝对路径;空格一行后的“Echo:”则表示当前执行的target名为“Echo”;“[echo]”表示当前输出是由Ant的echo任务(Task)发起的。最后给出“BUILD SUCCESSFUL”的提示及执行的消耗时长。

执行Ant构建文件还有一个更简单的方式，把构建文件命名为build.xml,则只要在构建文件所在目录下执行命令行ant，即可运行。

## Project Target Task 三个臭皮匠，顶个诸葛亮

通过以上操作，Ant的大门已经为你打开。现在解释介绍一下Ant中重要的三个元素。

**project:**

project元素有三个属性。

name表示当前工程名称;

当Ant被调用时，如果没有指定执行具体的目标(target)，则default将指定要执行的目标;

basedir用于指定Ant的基础目录，Ant中所有的相对路径都将以此值为基础进行计算。

这三个元素都不是必备的，但为了方便工程的执行，通常都会对default进行赋值。在代码1.1 中，直接就支接运行了Echo目标打印“Hello world!”。

**target:**

target是一系列task任务的有序集合。

1. target的分类：

属性description为当前目标(target)处理的业务逻辑进行描述。有被赋值的目标(target)称为Main Target。未被声明的则为Subtarget。通过命令行：ant –f buildfile.xml –projecthelp可以将target区分开来。

<?xml version="1.0" encoding="UTF-8"?>

<project name="Hello\_World" default="Echo" basedir=".">

<target name="Echo" description="Just print 'Hello world'.">

<echo message="Hello world! I'm ant."></echo>

</target>

<target name="EchoSubTarget">

</target>

</project>

代码2.1 Target分类

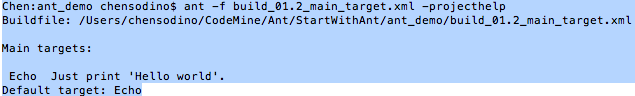


图2.1 Target分类

由上图可看出，Main Target清单中有”Echo”，并做出了描述;但未对description赋值的target：EchoSubTarget并没有被打印出来。

2. target具有依赖特性

先给出一段代码，如下：

代码2.2 target的依赖特性

<target name="A"/>

<target name="B" depends="A"/>

<target name="C" depends="B"/>

<target name="D" depends="C,B,A"/>

2.1 target的依赖关系

属性”depends”可以决定target之间依赖关系;并且不限定依赖个数，即一个子target可以同时依赖多个父target。

2.2 target多重依赖的调用顺序

假若要执行D，由于C依赖自B，B依赖自A，所以这些target的调用顺序会是:A🡪B🡪C🡪D。

2.3 target多次依赖的调用次数

D依赖了A、B、C，但由于C与A、B及B与A的依赖关系，在执行D时前的初始化操作中，A、B、C都仅会且仅会被执行一次。并不会因为D中刻意声明依赖自B和A会再单独执行B、A，因为C的初始化操作中已经执行过一次了。

2.4 依赖的传承

在A声明的property或tstamp仍可以在B、C、D中使用。

在下面的代码中，默认目标target为D。当Ant执行此文件，由于D的依赖关系，会依次调用A、B、C，并且A、B、C都只会被调用一次。而D作为最底层的依赖目标target，可以直接使用上层被依赖target内的所有属性。

代码2.3 依赖关系示例

<?xml version="1.0" encoding="UTF-8"?>

<project name="TargetDepends" default="D" basedir=".">

<target name="A" description="First Class A">

<property name="a.value" value="Apple" />

<echo message="I'm a ${a.value}."></echo>

</target>

<target name="B" description="First Class B" depends="A">

<property name="b.value" value="Bee" />

<echo message="I'm a ${b.value}."></echo>

</target>

<target name="C" description="First Class C" depends="B">

<property name="c.value" value="Car" />

<echo message="I'm a ${c.value}."></echo>

</target>

<target name="D" description="First Class D" depends="C,B,A">

<echo message="a.value=${a.value}"/>

<echo message="b.value=${b.value}"/>

<echo message="c.value=${c.value}"/>

</target>

</project>

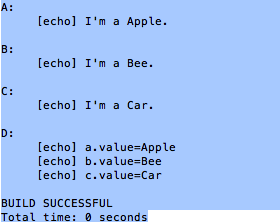


图2.3 依赖关系示例

**3. project与target的小关系：**

每个Ant文件由单个的project元素组成，该元素可以包含一个或多个target元素。一个目标(target)是生成过程中已定义的一个步骤，可以定义任意数量的操作，比如编译源代码，对Apk进行签名等。这些操作本身由其他专用任务(Task)标签执行。通常为了编写的灵活性，把数量较多的操作根据其业务逻辑划分到单独的目标(target)元素中，这样可以有选择性的执行整体生成过程中的单独部分。例如，可以调用某些目标元素，直接对安装包进行重新签名，而不需要从头执行编译操作。

**4. task**

task是指一系列内嵌在target的任务元素。用它来处理具体的业务操作。在上面的例子中，已经使用到打印输出的任务task：“echo”。在后面的章节中，会对具体的任务task用法进行详细解析及给出实例。

## 属性property

自定义构建过程中，属性property提供获取高频度使用的字符串的简便方式。一个属性property拥有一个区分大小写的name和一个value，要直接使用某一属性property的具体赋值时，需要引用这样的格式：*”${name}”*。假设有一属性property名为*”builddir”*并被赋值为*”build”*，则可以这样引用到到一个新值中去，如*”${builddir}/classes”*，在运行时环境中该值将被解析为*”build/classes”*。

**1. 属性property的不变性**