

Ant 是一个 Apache 基金会下的跨平台的构件工具，它可以实现项目的自动构建和部署等功能。在本文中，主要让读者熟悉怎样将 Ant 应用到 Java 项目中，让它简化构建和部署操作。

## 一. 安装与配置

下载地址: <http://ant.apache.org/>，在本文中下载的是 1.7.0 版本。  
解压到某个目录（例如 E:"apache-ant-1.7.0"），即可使用。

添加系统环境变量: ANT\_HOME，该变量指向 Ant 解压后的根目录，在此为 E:"apache-ant-1.7.0"。

安装与配置完毕后，读者可以测试一下 Ant 是否可用，首先进入 Ant 的 bin 目录，运行命令 `ant -version`，若安装和配置成功，则会显示 Ant 版本信息，如下图所示：

由上可以看出，读者运行 Ant 的命令时，需要进入到 Ant 的 bin 目录，如何才能让系统自动找到 Ant 呢？这时需要读者在系统环境变量 path 中添加 Ant 的 bin 目录。设置完成后，我们就可以在任何目录（例如 C:"Documents and Settings"AmigoXie 目录）输入 Ant 的命令，来获得命令的运行结果。

## 二. Ant 的关键元素

Ant 的构件文件是基于 XML 编写的，默认名称为 build.xml。为了更清楚的了解 Ant，在这里编写一个简单的 Ant 程序，用来展现 Ant 的功能，让读者

对 Ant 有一个初步的了解。首先在 E 盘下建立一个 build.xml 文件，内容如下：

```
<?xml version="1.0"?>
<project name="helloWorld">
    <target name="sayHelloWorld">
        <echo message="Hello,Amigo"/>
    </target>
</project>
```

读者可以进入 E 盘，然后运行 `ant sayHelloWorld`，可以看到如下的运行结果：

其中 `sayHelloWorld` 为需要执行的任务的名称。如果文件名不为 `build.xml`，而为 `hello.xml` 时，读者运行同样的命令时，命令窗口会出现如下错误：

```
Buildfile: build.xml does not exist!
```

```
Build failed
```

由上面的命令的错误提示可以看出，`ant` 命令默认寻找 `build.xml` 文件。若文件名为 `hello.xml` 时，读者还需要对命令做少许改变，改为：`ant -f hello.xml sayHelloWorld`、`ant -buildfile hello.xml sayHelloWorld` 或 `ant -file hello.xml sayHelloWorld`。

接下来开始向读者讲解本节的重点：Ant 的关键元素 `project`、`target`、`property` 和 `task`。

## 1. `project` 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。

### 1) `name` 属性

用于指定 `project` 元素的名称。

### 2) `default` 属性

用于指定 `project` 默认执行时所执行的 `target` 的名称。

### 3) `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。

下面给读者一个简单的例子来展示 `project` 的各元素的使用。修改 `E:\build.xml` 文件，修改后的内容如下：

```
<?xml version="1.0"?>
<project name="projectStudy" default="sayBaseDir" basedir
="E:\apache-ant-1.7.0">
    <target name="sayBaseDir">
```

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

    </target>
</project>
```

从上面的内容我们可以看出，在这里定义了 default 属性的值为 sayBaseDir，即当运行 ant 命令时，若未指明执行的 target 时，默认执行的 target 的 sayBaseDir，同时还定义了 basedir 属性的值为 E:"apache-ant-1.7.0，进入 E 盘后运行 ant 命令，可看到运行的结果，如下图所示：

因为设定了 basedir 的值，所以 basedir 属性的值变成了读者设置的值。读者可以自行将 project 元素的 basedir 属性去掉后运行 ant 看看运行结果，此时 basedir 的值变成了 E:"，即为 Ant 构件文件的父目录。

有的时候，读者可能有这种需求，即想得到某个 project 下所有的 target 的名称，读者可以通过在 ant 命令里加上 -projecthelp 来达到该目的。例如针对上述的例子我们运行 ant -projecthelp，输出结果如下：

```
Buildfile: build.xml
```

```
Main targets:
```

```
Other targets:
```

```
sayBaseDir
```

```
Default target: sayBaseDir
```

## 2. **target** 元素

它为 Ant 的基本执行单元，它可以包含一个或多个具体的任务。多个 target 可以存在相互依赖关系。它有如下属性：

### 1) **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。

### 2) **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。

### 3) **if** 属性

用于验证指定的属性是否存在，若不存在，所在 target 将不会被执行。

### 4) **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。

### 5) **description** 属性

该属性是关于 target 功能的简短描述和说明。

下面带领读者来看一个各属性综合使用的例子。修改 E:"build.xml 文件，修改后的内容如下：

```
<?xml version="1.0"?>
<project name="targetStudy">
    <target name="targetA" if="ant.java.version">
        <echo message="Java Version: ${ant.java.ver
sion}"/>
    </target>
    <target name="targetB" depends="targetA" unless="a
migo">
        <description>
            a depend example!
        </description>
        <echo message="The base dir is: ${basedir}"
/>
    </target>
</project>
```

进入 E 盘后运行 ant targetB，可看到如下图所示的运行结果：

读者分析结果后可以看到，我们运行的是名为 targetB 的 target，因该 target 依赖于 targetA，所以 targetA 将首先被执行，同时因为系统安装了 java 环境，所以 ant.java.version 属性存在，执行了 targetA 这个 target，输出信息：[echo] Java Version: 1.5，targetA 执行完毕后，接

着执行 targetB, 因为 amigo 不存在, 而 unless 属性是在不存在时进入所在的 target 的, 由此可知 targetB 得以执行, 输出信息: The base dir is: E:"。

### 3. **property** 元素

该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入: <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 "\${" 和 "}" 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, 该属性在讲解 project 元素时有详细说明, 不再赘述;

ant.file: buildfile 的绝对路径, 如上面的各例子中, ant.file 的值为 E:"build.xml";

ant.version: Ant 的版本, 在本文中, 值为 1.7.0;

ant.project.name: 当前指定的 project 的名字，即前文说到的 project 的 name 属性的值；

ant.java.version: Ant 检测到的 JDK 的版本，在上例运行结果中可看到为 1.5。

下面让读者来看一个 property 元素使用的简单例子。修改 E:\build.xml 文件，内容如下：

```
<?xml version="1.0"?>
<project name="propertyStudy" default="example">
  <property name="name" value="amigo"/>
  <property name="age" value="25"/>
  <target name="example">
    <echo message="name: ${name}, age: ${age}"/>
  </target>
</project>
```

该例的运行结果如下图所示：

由此读者可以看出，通过如下两个语句：

```
<property name="name" value="amigo"/>
```

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

我们设置了名为 name 和 age 的两个属性，这两个属性设置后，读者在下文中可以通过 \${name} 和 \${age} 分别取得这两个属性的值。



### 三. **Ant** 的常用任务

在 Ant 工具中每一个任务封装了具体要执行的功能，是 Ant 工具的基本执行单位。在本小节中，主要引导读者来看下 Ant 的常用任务及其使用举例。

#### 1. **copy** 任务

该任务主要用来对文件和目录的复制功能。举例如下：

Eg1.复制单个文件：<copy file="file.txt" tofile="copy.txt" />

Eg2.对文件目录进行复制：

```
<copy todir="../newdir/dest_dir">
```

```
<fileset dir="src_dir"/>
```

```
</copy>
```

Eg3. 将文件复制到另外的目录：

```
<copy file="file.txt" todir="../other/dir"/>
```

#### 2. **delete** 任务

对文件或目录进行删除，举例如下：

Eg1. 删除某个文件：<delete file="photo/amigo.jpg"/>

Eg2. 删除某个目录：<delete dir="photo"/>

Eg3. 删除所有的备份目录或空目录:

```
<delete includeEmptyDirs="true">  
  
    <fileset dir="." includes="**/*.bak"/>  
  
</delete>
```

### 3. **mkdir** 任务

创建目录。eg: `<mkdir dir="build"/>`

### 4. **move** 任务

移动文件或目录，举例如下:

Eg1. 移动单个文件: `<move file="fromfile" tofile="tofile"/>`

Eg2. 移动单个文件到另一个目录: `<move file="fromfile" todir="movedir"/>`

Eg3. 移动某个目录到另一个目录:

```
<move todir="newdir">  
  
    <fileset dir="olddir"/>  
  
</move>
```

### 5. **echo** 任务

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="Hello,Amigo" file="logs/system.log" append="true">
```

#### 四. 利用 **Ant** 构建和部署 **Java** 工程

Ant 可以代替使用 javac、java 和 jar 等命令来执行 java 操作，从而达到轻松的构建和部署 Java 工程的目的。下面来看几个知识点。

##### 1. 利用 **Ant** 的 **javac** 任务来编译 **java** 程序

Ant 的 javac 任务用于实现编译 Java 程序的功能。下面来看一个简单的例子：

首先我们建立名为 antstudy 的 Java 工程，建立 src 目录为源代码目录，在 src 目录下建立 HelloWorld.java 这个类文件。该类文件的内容如下：

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello,Amigo");  
    }  
}
```

同时在 antstudy 工程的根目录下建立 build.xml 文件，在该文件中编译 src 目录下的 java 文件，并将编译后的 class 文件放入 build/classes 目录中，在编译前，需清除 classes 目录，该文件的内容如下：

```
<?xml version="1.0"?>
<project name="javacTest" default="compile" basedir=". ">
    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile" depends="clean">
        <mkdir dir="build/classes"/>
        <javac srcdir="src" destdir="build/classes"/>
    </target>
</project>
```

运行该 build.xml 文件,可在工程中看到新增了 build/classes 目录,并在该目录中生成了编译后的 HelloWorld.class 文件。

## 2. 使用 Ant 的 java 任务运行 Java 程序

Ant 中可以使用 java 任务实现运行 Java 程序的功能。下面在 1 的例子中进行如下的修改,修改后的 build.xml 文件的内容如下:

```
<?xml version="1.0"?>
<project name="javaTest" default="jar" basedir=". ">
    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile" depends="clean">
        <mkdir dir="build/classes"/>
```

```

        <javac srcdir="src" destdir="build/classes"/>
    </target>

    <target name="run" depends="compile">
        <java classname="HelloWorld">
            <classpath>
                <pathelement path="build/classes"/>
            </classpath>
        </java>
    </target>
</project>

```

运行该 build.xml 文件，可在控制台看到 HelloWorld 的 main 方法的输出。

### 3. 使用 Ant 的 jar 任务生成 jar 文件

读者可以在上例的基础上更进一步，来生成 jar 包，可在 run 这个 target 下再加上如下 target:

```

<target name="jar" depends="run">
    <jar destfile="helloworld.jar" basedir="build/classes">
        <manifest>
            <attribute name="Main-class" value="HelloWorld"/>
        </manifest>
    </jar>
</target>

```

```
        </jar>
    </target>
```

此时将 ant 的 project 的 default 属性设置为 jar,同时运行该 build.xml 文件,运行完毕后,可看到在工程目录下生成了一个 jar 包 HelloWorld.jar。

#### 4. 使用 Ant 的 war 任务打包 J2EE Web 项目

建立一个 J2EE Web 工程,其目录结构如下图所示:

其中 src 为源代码目录,WebRoot 为各 jsp 存放目录,lib 为工程的包目录。在 antwebproject 工程目录下建立了 build.xml 文件,该文件为该工程的 Ant 构件文件。读者可以在 src 目录下放入在前续例子中开发的 HelloWorld.java 文件,并在 WebRoot 下建立 index.jsp 文件,其内容很简单,就是输出 Hello 信息,代码如下所示:

```
<%@ page language="java" contentType="text/html; charset=
"UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional
//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/htm
l; charset=ISO-8859-1">

        <title>ant 打包测试</title>
```

```
</head>

<body>

    Hello,Ant

</body>

</html>
```

接下来编写 build.xml 文件，其内容如下：

```
<?xml version="1.0"?>
<project name="antwebproject" default="war" basedir=".">
    <property name="classes" value="build/classes"/>
    <property name="build" value="build"/>
    <property name="lib" value="WebRoot/WEB-INF/lib"/>

    <!-- 删除 build 路径-->

    <target name="clean">
        <delete dir="build"/>
    </target>

    <!-- 建立 build/classes 路径，并编译 class 文件到 build/classes 路径下-->

    <target name="compile" depends="clean">
        <mkdir dir="${classes}"/>

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

    <!-- 打 war 包-->
```

```

    <target name="war" depends="compile">
<war destfile="${build}/antwebproject.war" webxml="WebRoot/WEB-INF/web.xml">

    <!-- 拷贝 WebRoot 下除了 WEB-INF 和 META-INF 的两个文件夹-->

    <fileset dir="WebRoot" includes="**/*.jsp"/>

    <!-- 拷贝 lib 目录下的 jar 包-->

    <lib dir="${lib}"/>

    <!-- 拷贝 build/classes 下的 class 文件-->

    <classesdir="${classes}"/>

    </war>

</target>
</project>

```

各 target 的作用在内容中已经进行说明，在此不再赘述。运行该 build 文件，更新目录后，可看到在 build 目录下生成了 antwebproject.war 文件，解开后可看到其目录结构如下：

```
--META-INF
```

```
--MANIFEST.MF
```

```
--index.jsp
```

```
--WEB-INF
```

```
--lib
```



```
--log4j-1.2.9.jar

--classes

--HelloWorld.class

--web.xml
```

读者可以将该 war 包拷贝到 Tomcat 的目录下看一下运行结果。

## 五. 总结

在本文中，笔者由浅至深详细描述了 Ant 的安装与配置、关键元素和常用任务。并通过实例讲述了 Ant 在我们 Java 项目中的应用，讲述了编译、运行 java 程序，以及打 jar 包、war 包等知识，引领读者进入 Ant 的奇妙世界。在本文中可以看到，Ant 在自动构建和部署 Java 程序方面方便易用，而且非常灵活，不失为我们 Java 开发者的绝佳帮手