

Agile Java

阿左¹ Nobody²

October 9, 2012

¹感谢读者
²感谢国家

Contents

I 基础入门	4
1 开发环境	5
1.1 开发环境配置	5
1.2 编写第一个 ant 程序	5
2 起步	7
2.1 project 元素	7
2.2 target 元素	8
2.2.1 depends 属性	8
2.2.2 if 属性	8
2.2.3 unless 属性	9
2.3 property 元素	10
2.4 task 元素	11
2.5 命令行使用 ant	11
2.5.1 指定文件	11
2.5.2 显示任务描述	11
2.5.3 指定变量的值	11

3 核心类型	12
3.1 断言类型: Assertions Type	12
3.2 匹配模式: PatternSet	13
3.2.1 包含与排除	13
3.2.2 过滤的模式	14
3.2.3 例子	14
3.3 匹配目录: DirSet	15
3.4 匹配目录: FileSet	16
3.4.1 设置匹配	16
3.4.2 过滤筛选: selector	16
3.4.3 内容过滤: contains	16
3.4.4 时间过滤: date	17
3.4.5 比较过滤: depend	17
3.4.6 深度过滤: depth	18
3.4.7 差异过滤: different	18
3.4.8 文件名过滤: filename	19
3.4.9 目录过滤: parent	19
3.4.10 正则过滤: containsregexp	19
3.4.11 大小过滤: size	19
3.4.12 类型过滤: type	20
3.5 文件列表: FileList	20
3.6 文件过滤器: FilterSet	21
3.6.1 主要属性	21
3.6.2 替换版权与日期的例子	21

3.6.3 把替换的内容放在属性文件中	22
3.7 属性集合: PropertySet	23
3.7.1 属性与功能	23
3.7.2 引用已经存在的属性	23
3.7.3 使用属性集合的例子	24
3.8 文件映射: mapper	24
3.8.1 identity	25
3.8.2 flatten	25
3.8.3 glob	26
3.8.4 merge	26
3.8.5 regexp	26
3.8.6 package	27
3.8.7 composite	27
3.8.8 chained	28
3.8.9 filtermapper	28
3.9 压缩文件: zip	29
3.9.1 属性与功能	29
3.9.2 例子	29
3.10 过滤链与过滤读取器: FilterChains and FilterReader	30
3.11 定制与扩展	31
3.11.1 定制条件判断: Condition	31
3.11.2 定制选择器: Selector	32
3.11.3 定制过滤器: Filter	33

CONTENTS	4
II 工具整合	34
III 实践	35

List of Figures

List of Tables

Part I

基础入门

Chapter 1

开发环境

1.1 开发环境配置

基本环境变量的配置：

JAVA_HOME、ANT_HOME、PATH 这三个环境变量要配置正确。

1.2 编写第一个 ant 程序

```
1 <?xml version="1.0"?>
2
3 <project name="myproject" default="firstExample">
4
5   <target name="firstExample">
6     <echo message="This is my first targe" />
7     <echo message="          os: ${os.name}" />
8     <echo message="          baseidr: ${basedir}" />
9     <echo message=" ant fileis: ${ant.file}" />
```

```
10     <echo message=" ant version: ${ant.version}" />
11     <echo message="project name: ${ant.project.name}" />
12     <echo message="java version: ${ant.java.version}" />
13 </target>
14
15 </project>
```

Chapter 2

起步

2.1 project 元素

一个 ant 文件中一定要有一个 project 元素。name 属性定义工作名字；default 属性代表默认启动 target；description 属性定义说明文本。

```
1 <?xml version="1.0"?>
2
3 <project name="myproject" default="firstExample">
4
5   <description>
6     Show this description with command:
7     ant -projecthelp
8   </description>
9
10 </project>
```

2.2 target 元素

target 元素一定要有 name 属性。

2.2.1 depends 属性

表示当前 target 执行依赖于其他的 target 成功执行。可以有多个 depends 之间用“,”分隔。

```
1 <?xml version="1.0"?>
2
3 <project name="myproject" default="firstExample">
4
5   <description>
6     Show this description with command:
7     ant -projecthelp
8   </description>
9
10 </project>
```

2.2.2 if 属性

验证执行前必须设定的属性。例如设定 JDK:

```
1 <?xml version="1.0"?>
2
3 <!--
4   ant -f build.03.xml
5   ant -f build.03.xml checkFinished
6 -->
7 <project name="myproject" default="firstExample">
8
```

```
9 <target name="showJava" if="ant.java.version">
10   <echo message="java version: ${ant.java.version}" />
11 </target>
12
13 <!--
14 <property name="isFinished" value="finished" />
15 -->
16
17 <target name="checkFinished" if="isFinished">
18   <echo message="is finished: ${isFinished}" />
19 </target>
20
21 <target name="firstExample" depends="showJava, checkFinished">
22   <echo message="always show this" />
23 </target>
24 </project>
```

2.2.3 unless 属性

没有设置才执行

```
1 <?xml version="1.0"?>
2
3 <!--
4   ant -f build.04.xml checkUnFinished
5 -->
6 <project name="myproject" default="firstExample">
7   <!--
8   <property name="isFinished" value="finished" />
9   -->
10
11   <target name="checkUnFinished" unless="isFinished">
12     <echo message="show this if not set ifFinished" />
```

```
13 </target>
14
15 <target name="firstExample" depends="checkUnFinished">
16     <echo message="always show this" />
17 </target>
18
19 </project>
```

2.3 property 元素

property 可以看作参数或是变量的定义。可以在构建文件中通过 property 元素建立，也可以在构建文件外建立一个 build.properties 文件来存放。

```
1 <?xml version="1.0"?>
2
3 <project name="myproject" default="firstExample">
4
5     <!-- reference properties file -->
6     <property file="build.properties" />
7
8     <property name="src" value="src" />
9
10    <target name="init">
11        <!-- get current time -->
12        <tstamp/>
13    </target>
14
15    <target name="firstExample" depends="init">
16        <echo message="author name:${author.name}" />
17        <echo message="author nick:${author.nickname}" />
18        <echo message="currentTime:${DSTAMP}" />
19    </target>
```

```
20  
21 </project>
```

2.4 task 元素

task 元素是一系列完成指定功能的脚本和程序。比如自带的编译用的 `javac`、打包用的 `jar`、建立目录的 `mkdir` 等。用户也可以编写自己的 task。

2.5 命令行使用 ant

2.5.1 指定文件

`-buildfile` 或 `-file` 或 `-f` 都可以。

2.5.2 显示任务描述

`-projecthelp`。

2.5.3 指定变量的值

`-D<property>=<value>`

Chapter 3

核心类型

3.1 断言类型: Assertions Type

断言类型指定哪些类需要让 ant 工具执行 java 断言。enableSystemAssertions 属性设定是否允许系统断言，默认为 unspecified。

断言类型内还可以定义 enable 类型与 disable 类型，通过这两个类型的 class 属性与 package 属性来设置指定的类或是包是否可以执行断言。

允许所有的用户代码执行断言：

```
1 <assertions>
2   <enable />
3 </assertions>
```

只允许 Test 这个类执行断言：

```
1 <assertions>
2   <enable class="Test"/>
3 </assertions>
```


允许一个包下所有的类都执行断言：

```
1 <assertions>
2   <enable package="org.apache"/>
3 </assertions>
```

更复杂的定义：

```
1 <assertions>
2   <enable package="org.apache"/>
3   <disable package="org.apache.tools.ant"/>
4   <enable class="org.apache.tools.ant.Main"/>
5 </assertions>
```

引用一个已经存在的定义：

```
1 <assertions id="project.assertions">
2   <disable package="org.apache.test"/>
3 </assertions>
4
5 <assertions refid="project.assertions" />
```

3.2 匹配模式：PatternSet

3.2.1 包含与排除

匹配格式通过以下四个属性指定包含与排除：

`include` 属性：可以指定多个模式用逗号或空格分隔，相当于一个列表。

`includesfile` 属性：指定具体包含的文件。可以引用在 `properties` 文件中指定的内容。

`exclude` 属性：和上面的相反。

`excludesfile` 属性：和上面的相反。

3.2.2 过滤的模式

以上四个属性都有三个属性来匹配指定的内容：

name 属性：文件名、路径或格式。

if 属性：指定的属性有定义则生效。

unless 属性：指定的属性没有定义则生效。

3.2.3 例子

类名中不包含 Test 的类：

```
1 <patternset id="non.test.sources">
2   <include name="**/*.java" />
3   <exclude name="**/*Test.java" />
4 </patternset>
```

引用已经定义的模式：

```
1 <patternset refid="non.test.sources">
```

可以包含多个定义：

```
1 <patternset id="sources">
2   <include name="std/**/*.java" />
3   <include name="prof/**/*.java" if="professional" />
4   <exclude name="**/*Test.java" />
5 </patternset>
```

指定文件名中包含了"some-file" 的文件：

```
1 <patternset>
2   <includesfile name="some-file" />
3 </patternset>
```

也可以简写成下面的格式：

```
1 <patternset includesfile="some-file" />
```

配合 if 条件的形式：

```
1 <patternset>
2   <includesfile name="some-file" if="isInclue" />
3 </patternset>
```

3.3 匹配目录：DirSet

DirSet 类似于 PatternSet，用于匹配目录。必填属性 `dir` 用于指定一个目录。

DirSet 可以引用已经存在的 PatternSet，也可以直接使用 `include`、`includesfile`、`exclude`、`excludesfile` 属性。

另外，DirSet 还有 `casesensitive` 属性来设置是否大小写敏感；`followsymlinks` 属性来设置采用操作系统差异（如 linux 与 windows 的路径符号）等。

定义目录集合的例子：

```
1 <dirset dir="${build.dir}">
2   <include name="**/*.java" />
3   <exclude name="**/*Test.java" />
4 </dirset>
5
6 <dirset dir="${build.dir}">
7   <patternset id="non.test.classes">
8     <include name="**/*.java" />
9     <exclude name="**/*Test.java" />
10  </patternset>
11 </dirset>
12
13 <dirset dir="${build.dir}" >
```

```
14 <patternset refid="non.text.classes" />
15 </dirset>
```

3.4 匹配目录：FileSet

3.4.1 设置匹配

FileSet 匹配符合的文件，它不仅包含 DirSet 的属性外，还有一些其他的属性：

file 属性：指定单个文件，fileSet 定义中至少要 dir 属性或是 file 属性。

defaultexcludes：当值为 yes 时默认忽略指定的文件（如版本控制文件），常用的模式有：

```
**/*~, **/#*#, **/.#*, **/%*%, **/._* ,
**/CVS, **/CVS/**, **/.cvsignore,
**/SCCS, **/SCCS/**, **/vssver.scc,
**/.svn, **/.svn/**, **/.DS_Store
```

3.4.2 过滤筛选：selector

selector 可以看作是 FileSet 中的一个元素。有以下几种常见的过滤工具：

3.4.3 内容过滤：contains

只选择包含 text 属性定义字符串的文件。

text 属性：文件包含的字符串，不能为空。

casesensitive 属性：大小写敏感。

ignorewhitespace 属性：忽略空白字符。

```
1 <fileset dir="${doc.path}" includes="**/*.html">
2   <contains text="script" casesensitive="no" />
3 </fileset>
```

3.4.4 时间过滤：date

datetime 属性：指定时间格式为：MM/DD/YYYY HH:MM AM。例如：09/09/2006 10:10 am

milis 属性：指定毫秒时间。datetime 与 milis 两个必选一个。

when 属性：文件修改时间的比较方式，默认为 equals。其他值还有：before 与 after。

granularity 属性：允许的时间误差毫秒数。

pattern 属性：指定 datetime 是否兼容 Java 中的 SimpleDateFormat 格式。

checkdirs 属性：是否检查文件目录创建时间，默认为 false。

```
1 <fileset dir="${doc.path}" includes="**/*.jar">
2   <date datetime="01/01/2001 12:00 AM" when="before" />
3 </fileset>
```

3.4.5 比较过滤：depend

比较两个目录下同名的文件，选择最后修改的那个。

targetdir 属性：指定进行比较的目录。

granularity 属性：允许的时间误差毫秒数。

比较 1.4 版与 1.5 版中有修改过的源文件。

```
1 <fileset dir="${ant.1.5}/src/main" includes="**/*.java">
2   <depend targetdir="${ant.1.4}/src/main" />
3 </fileset>
```

3.4.6 深度过滤：depth

min 最小与 max 最大。

```
1 <fileset dir="${doc.path}" includes="**/*">
2   <depend max="1" />
3 </fileset>
```

3.4.7 差异过滤：different

比较目录下的文件中是否有不同（当前目录和一层子目录）。比较的内容有：

- 1) 一个地方有另一个地方没有。
- 2) 文件大小不同。
- 3) ignoreFileTimes 不为 off，且文件修改时间不同。
- 4) ignoreContents 为 true，内容不同。

属性有：targetdir、granularity（允许的时间误差范围）、ignoreFileTimes、ignoreContents。

```
1 <fileset dir="${ant.1.5}/src/main" includes="**/*.java">
2   <different targetdir="${ant.1.4}/src/main" />
3 </fileset>
```

3.4.8 文件名过滤：filename

主要属性：name (文件名匹配)、casesensitive、negate (反选，选中不匹配的)。

例如，选择所有的 css 样式文件：

```
1 <fileset dir="${doc.path}" includes="**/*">
2   <filename name="**/*.css" />
3 </fileset>
```

3.4.9 目录过滤：present

选择一个与当前目录对应的目录 (targetdir)，找出指定目录中不存在或是两个目录中都存在的文件。主要属性有：

present 属性：值为 srconly 本目录有而 targetDir 没有的文件；值为 both 选中两个目录都有的文件。

```
1 <fileset dir="${ant.1.5}/src/main" includes="**/*.java">
2   <present present="only" targetdir="${ant.1.4}/src/main" />
3 </fileset>
```

3.4.10 正则过滤：containsregexp

```
1 <fileset dir="${doc.path}" includes="**/*">
2   <containsregexp expression="[4-6]\.[0-9]" />
3 </fileset>
```

3.4.11 大小过滤：size

value 属性：值。

units 属性：单位。以 1000 为进制的 K, M, G。或是以 1024 为单位的 Ki, Mi, Gi。默认为 bytes。

when 属性：more, less, equal。

```
1 <fileset dir="${jar.path}">
2   <patternset>
3     <include name="**/*.jar" />
4   </patternset>
5   <size value="4" unites="Ki" when="more" />
6 </fileset>
```

3.4.12 类型过滤：type

指定是文件（file）还是目录（dir）。

```
1 <fileset dir="${src}">
2   <type type="dir" />
3 </fileset>
```

3.5 文件列表：FileList

```
1 <filelist id="docfiles" dir="${doc.src}"
2   files="foo.xml,bar.xml" />
3
4 <filelist id="docfiles2" refid="docfiles"/>
5
6 <filelist id="docfiles3" dir="${doc.src}" >
7   <file name="foo.xml" />
8   <file name="bar.xml" />
9 </filelist>
```


3.6 文件过滤器：FilterSet

过滤器可以在对文件进行复制或移动时对文件内容进行替换操作。

3.6.1 主要属性

- 1) begintoken 属性：用一个字符（默认为 @）指定过滤字符串的开始。如：@DATE@
- 2) endtoken 属性：定义结束。
- 3) recurse 属性：是否查找多个替换标记，默认为 true。

3.6.2 替换版权与日期的例子

```
1 <target name="ex01" >
2   <tstamp>
3     <format property="now" pattern="yyyy/MM/dd HH:mm:ss" />
4   </tstamp>
5   <copy todir="build" filtering="true">
6     <fileset dir="src">
7       <include name="**/*.java" />
8     </fileset>
9     <!-- find @COPYRIGHT@ and @BUILD_DATE@ -->
10    <filterset>
11      <filter token="BUILD_DATE" value="${now}" />
12      <filter token="COPYRIGHT" value="Jade Shan" />
13    </filterset>
14  </copy>
15 </target>
```

可引用的版本：

```
1 <!-- get current time -->
```

```
2 <tstamp>
3   <format property="now" pattern="yyyy/MM/dd HH:mm:ss" />
4 </tstamp>
5
6 <!-- find @COPYRIGHT@ and @BUILD_DATE@ -->
7 <filterset id="date.copyright.filterset">
8   <filter token="BUILD_DATE" value="${now}" />
9   <filter token="COPYRIGHT" value="Jade Shan" />
10 </filterset>
11
12 <target name="firstExample" >
13   <copy todir="build" filtering="true">
14     <fileset dir="src">
15       <include name="**/*.java" />
16     </fileset>
17     <filterset refid="date.copyright.filterset" />
18   </copy>
19 </target>
```

3.6.3 把替换的内容放在属性文件中

要替换的文本:

```
1 This is the test sample.
2 The message will be replaced: @MESSAGE@
```

属性文件:

```
1 MESSAGE=HELLO
```

构建文件:

```
1 <filterset id="prop.message">
2   <filtersfile file="abc.properties" />
```

```
3 </filterset>
4
5 <target name="filterProp" >
6   <copy todir="build" filtering="true">
7     <fileset dir="src">
8       <include name="src.txt" />
9     </fileset>
10    <filterset refid="prop.message" />
11  </copy>
12 </target>
```

3.7 属性集合：PropertySet

定义一套可以组其他标签引用的属性集合。

3.7.1 属性与功能

dynamic 属性：是否动态地加载属性。

negate 属性：取反默认为 false，如果为 true 代表返回除 PropertySet 外的属性。

3.7.2 引用已经存在的属性

Propertyref 类型可以引用一个已经定义的 property。主要属性有：

name 属性：引用的名字。

prefix 属性：引用指定开头的多个属性。

regex 属性：用正则匹配。

builtin 属性：引用 ant 内建的属性，值为 all 时表示所有内建属性。

3.7.3 使用属性集合的例子

组建一个属性集合：

```
1 <property name="lib1" value="lib1path" />
2 <property name="lib2" value="lib2path" />
3
4 <propertyset id="projectset1">
5   <projectref name="lib1" />
6   <projectref name="lib2" />
7 </propertyset>
```

projectset 间相互引用的例子：

```
1 <propertyset id="properties-starting-with-foo">
2   <propertyref prefix="foo" />
3 </propertyset>
4
5 <propertyset id="properties-starting-with-bar">
6   <propertyref prefix="bar" />
7 </propertyset>
8
9 <propertyset id="my-set">
10   <propertyset refid="properties-starting-with-foo" />
11   <propertyset refid="properties-starting-with-bar" />
12 </propertyset>
```

3.8 文件映射：mapper

用来定义文件之间的对应关系的，主要属性有：

type 属性：定义一个实现的类型，可以用现成的也可以自己实现一个。

classname 属性：通过类名指定一个实现类型。type 和 classname 一定要选一个。

classpath 属性：查找类的路径。

classpathref 属性：引用已经有的 path 作为 classpath。

from 属性：源文件的位置。

to 属性：目标文件的位置。

对于文件映射，不同的实现类提供了不同的映射实现方法，以下各小节是现有的实现。每个类的调用都可以通过类名调用与 type 属性调用两种方法来写：

3.8.1 identity

源文件与目标文件同名。只取文件名，忽略路径：

```
1 <!--
2     A.java                      -> A.java
3     foo/bar/B.java             -> B.java
4     C.properties               -> C.properties
5     Classes/dir/dri2/A.properties -> A.properties
6 -->
7 <mapper type="identity" />
8 <identitymapper />
```

3.8.2 flatten

忽略目录把文件放到一个压缩文件中：

```
1 <!--
2     A.java                      -> archive.tar
3     foo/bar/B.java             -> archive.tar
4     C.properties               -> archive.tar
5     Classes/dir/dri2/A.properties -> archive.tar
6 -->
7 <mapper type="flatten" />
```

```
8 <flattenmapper />
```

3.8.3 glob

匹配路径与文件名：

```
1 <!--  
2   A.java                      -> A.java.bak  
3   foo/bar/B.java             -> foo/bar/B.java.bak  
4   C.properties               -> Ignored  
5   Classes/dir/dri2/A.properties -> Ignored  
6 -->  
7 <mapper type="glob" from="*.java" to="*.java.bak"/>  
8 <globmapper      from="*.java" to="*.java.bak" />
```

3.8.4 merge

把源文件打包到压缩文件中：

```
1 <!--  
2   A.java                      -> archive.tar  
3   foo/bar/B.java             -> archive.tar  
4   C.properties               -> archive.tar  
5   Classes/dir/dri2/A.properties -> archive.tar  
6 -->  
7 <mapper type="merge" to="archive.tar" />  
8 <mergemapper      to="archive.tar" />
```

3.8.5 regexp

通过正则来映射：

```
1 <!--
2   A.java                -> A.java.bak
3   foo/bar/B.java        -> foo/bar/B.java.bak
4   C.properties          -> Ignored
5   Classes/dir/dri2/A.properties -> Ignored
6 -->
7 <mapper type="regexp" from="^(.*)\.java$$" to="\1.java.bak" />
8 <regexpmapper           from="^(.*)\.java$$" to="\1.java.bak" />
```

3.8.6 package

替换目录名称。

```
1 <!--
2   org/apache/tools/ant/util/PackageMapperTest.java
3   -> Test-org.apache.tools.ant.util.PackageMapperTest.xml
4   org/apache/tools/ant/util/Helper.java
5   -> ignored
6 -->
7 <mapper type="package" from="*Test.java" to="TEST-*Test.xml" />
8 <packagemapper         from="*Test.java" to="TEST-*Test.xml" />
```

3.8.7 composite

多个 mapper 都对源文件进行操作。Composite Mapper 不能通过 Mapper 类型的 type 属性指定:

```
1 <!--
2   foo/bar/A.java -> foo/bar/A.java
3   foo/bar/A.java -> foo.bar.A
4 -->
```

```
5 <compositemapper>
6   <identitymapper />
7   <packagemapper from="*.java" to="" />
8 </compositemapper>
```

3.8.8 chained

包含多个 mapper, 源文件依次经过每个 mapper 操作。

```
1 <!--
2   foo/bar/A.java -> new/path/A.java1
3   foo/bar/A.java -> new/path/A.java2
4   foo/bar/B.java -> new/path/B.java1
5   foo/bar/B.java -> new/path/B.java2
6 -->
7 <chainedmapper>
8   <flattenmapper/>
9   <globmapper from="*" to="new/path/*" />
10  <mapper>
11    <globmapper from="*" to="*1" />
12    <globmapper from="*" to="*2" />
13  </mapper>
14 </chainedmapper>
```

3.8.9 filtermapper

对文件名进行过滤:

```
1 <!--
2   foo\bar\A.java -> new/path/A.java1
3 -->
4 <filtermapper>
```



```
5 <replacestring from="\\" to="/" />
6 </filtermapper>
```

3.9 压缩文件：zip

有两种压缩文件：

- 1) 当使用 `src` 属性时，目录下的文件会以.zip 文件格式进行组织。
- 2) 当使用 `dir` 属性时，目录下的文件会以文件系统形式进行组织。

3.9.1 属性与功能

`prefix` 属性：文件路径前缀，符合的会被选中。

`fullpath` 属性：包含全路径。

`src` 属性：用于替代当前的目录位置。

`filemode` 属性与 `dirmode` 属性：文件的权限，如 linux 下的权限 777。

3.9.2 例子

以 zip 格式压缩 `htdocs/manual` 目录下的所有文件。存放到 `docs/user-guide` 目录下。

同时添加 `ChangeLog27.txt` 文件到 zip 文件的 `docs` 目录下。

这个 zip 文件还包含 `example.zip` 文件。`example.zip` 文件包含 `docs/examples` 目录及其子目录下的所有 html 文件。

```
1 <zip desfile="${dist}/manual.zip" >
2   <zipfileset dir="htdocs/manual"
3     prefix="docs/user-guide" />
```

```
4 <zipfileset dir="." includes="ChangeLog27.txt"
5   fullpath="docs/ChangeLog.txt" />
6 <zipfileset prefix="docs/examples" includes="**/*.html"
7   src="example.zip" />
8 </zip>
```

3.10 过滤链与过滤读取器：FilterChains and Filter-Reader

一组有序的 FilterReader 组成 FilterChains，用户可以实现自己的 FilterReader。

FilterReader 通过 classname 属性指定实现类。

在 ant 任务 concat、copy、loadFile、loadProperties、move 中都可以直接使用 FilterChain 进行过滤操作。

```
1 <copy file="${src.file}" tofile="${dest.file}" >
2   <!-- define the chains -->
3   <filterchain>
4     <filterreader classname="aa.FilterReader">
5       <param name="foo" value="bar" />
6     </filterreader>
7     <!-- define your own reader -->
8     <filterreader classname="bb.FilterReader">
9       <!-- class path -->
10      <classpath>
11        <pathelement path="${classpath}" />
12      </classpath>
13      <param name="bar" value="blee" />
14      <param name="abc" value="cadd" />
15    </filterreader>
16  </filterchain>
17 </copy>
```

3.11 定制与扩展

用户可以定制的有：conditions、selectors 和 filters 类型。

3.11.1 定制条件判断：Condition

实现一个判断字符串是大写的条件判断：

```
1 package exp.ant;
2
3 import org.apache.tools.ant.BuildException;
4 import org.apache.tools.ant.taskdefs.condition.Condition;
5
6 public class AllUpperCaseCondition implements Condition{
7     private String value;
8
9     public boolean eval() {
10         if(value == null){
11             throw new BuildException("value attribute is not set");
12         }
13         return value.toUpperCase().equals(value);
14     }
15
16     public void setValue(String value) {
17         this.value = value;
18     }
19 }
```

构建文件中通过 typedef 导入，然后使用定义的类。

```
1 <typedef name="alluppercase"  
2   classname="exp.ant.AllUpperCaseCondition"  
3   classpath="${src.class}" />  
4  
5 <condition property="allupper">  
6   <alluppercase value="THIS IS ALL UPPER CASE" />  
7 </condition>
```

3.11.2 定制选择器: Selector

```
1 package exp.ant;  
2  
3 import java.io.File;  
4 import org.apache.tools.ant.types.selectors.FileSelector;  
5  
6 public class JavaSelector implements FileSelector {  
7  
8   public boolean isSelected(File b, String filename, File f) {  
9     return filename.toLowerCase().endsWith(".java");  
10  }  
11 }
```

```
1 <typedef name="javaselector"  
2   classname="exp.ant.JavaSelector"  
3   classpath="${src.class}" />  
4  
5 <copy todir="to">  
6   <fileset dir="src">  
7     <javaselector/>  
8   </fileset>  
9 </copy>
```

ant 已经提供了一个 BaseSelector 做了一些预处理功能：setError(String errMsg) 可提供出错信息；validate() 方法会在 isSelected() 前进行验证。

3.11.3 定制过滤器：Filter

```
1 package exp.ant;
2
3 // ... other packages
4 import org.apache.tools.ant.types.filters.ChainableReader;
5
6 public class RevomeOddCharacters implements ChainableReader{
7
8     public Reader chain(Reader reader) {
9         return new BaseFilterReader(reader) {
10             int count = 0;
11             public int read() throws IOException{
12                 // other
13             }
14         }
15     }
16
17 }
```

直接通过类名调用：

```
1 <copy file="${src.file}" todir="to">
2     <filterchain>
3         <filterreader classname="exp.ant.RemoveOddCharacters" />
4     </filterchain>
5 </copy>
```

Part II

工具整合

Part III

实践