Ant的概念  
Make命令是一个项目管理工具，而Ant所实现功能与此类似。像make，gnumake和nmake这些编译工具都有一定的缺陷，但是Ant却克服了这些工具的缺陷。最初Ant开发者在开发跨平台的应用时，用样也是基于这些缺陷对Ant做了更好的设计。  
  
Ant 与 makefile  
Makefile有一些不足之处，比如很多人都会碰到的烦人的Tab问题。最初的Ant开发者多次强调”只是我在Tab前面加了一个空格，所以我的命令就不能执行”。有一些工具在一定程度上解决了这个问题，但还是有很多其他的问题。Ant则与一般基于命令的工具有所不同，它是Java类的扩展。Ant运行需要的XML格式的文件不是Shell命令文件。它是由一个Project组成的，而一个Project又可分成可多target，target再细分又分成很多task，每一个task都是通过一个实现特定接口的java类来完成的。  
  
Ant的优点  
  
Ant是Apache软件基金会JAKARTA目录中的一个子项目，它有以下的优点。  
   跨平台性。Ant是存Java语言编写的，所示具有很好的跨平台性。  
   操作简单。Ant是由一个内置任务和可选任务组成的。Ant运行时需要一个XML文件(构建文件)。  
  
Ant通过调用target树，就可以执行各种task。每个task实现了特定接口对象。由于Ant构建文件时XML格式的文件，所以和容易维护和书写，而且结构很清晰。  
Ant可以集成到开发环境中。由于Ant的跨平台性和操作简单的特点，它很容易集成到一些开发环境中去。  
  
Ant 开发  
  
Ant的构建文件  
   当开始一个新的项目时，首先应该编写Ant构建文件。构建文件定义了构建过程，并被团队开发中每个人使用。Ant构建文件默认命名为 build.xml，也可以取其他的名字。只不过在运行的时候把这个命名当作参数传给Ant。构建文件可以放在任何的位置。一般做法是放在项目顶层目录中，这样可以保持项目的简洁和清晰。下面是一个典型的项目层次结构。  
(1) src存放文件。  
(2) class存放编译后的文件。  
(3) lib存放第三方JAR包。  
(4) dist存放打包，发布以后的代码。  
Ant构建文件是XML文件。每个构建文件定义一个唯一的项目(Project元素)。每个项目下可以定义很多目标(target元素)，这些目标之间可以有依赖关系。当执行这类目标时，需要执行他们所依赖的目标。每个目标中可以定义多个任务，目标中还定义了所要执行的任务序列。Ant在构建目标时必须调用所定义的任务。任务定义了Ant实际执行的命令。Ant中的任务可以为3类。  
（1） 核心任务。核心任务是Ant自带的任务。  
（2） 可选任务。可选任务实来自第三方的任务，因此需要一个附加的JAR文件。  
（3） 用户自定义的任务。用户自定义的任务实用户自己开发的任务。  
1.<project>标签  
  每个构建文件对应一个项目。<project>标签时构建文件的根标签。它可以有多个内在属性，就如代码中所示，其各个属性的含义分别如下。  
(1) default表示默认的运行目标，这个属性是必须的。  
(2) basedir表示项目的基准目录。  
(3) name表示项目名。  
(4) description表示项目的描述。  
每个构建文件都对应于一个项目，但是大型项目经常包含大量的子项目，每一个子项目都可以有自己的构建文件。  
  
2.<target>标签   
一个项目标签驴梢杂幸桓龌蚨喔?/span>target标签。一个target标签可以依赖其他的target标签。例如，有一个target用于编译程序，另一个target用于声称可执行文件。在生成可执行文件之前必须先编译该文件，因策可执行文件的target依赖于编译程序的 target。Target的所有属性如下。  
(1).name表示标明，这个属性是必须的。  
(2).depends表示依赖的目标。  
(3)if表示仅当属性设置时才执行。  
(4)unless表示当属性没有设置时才执行。  
(5)description表示项目的描述。  
Ant的depends属性指定了target的执行顺序。Ant会依照depends属性中target出现顺序依次执行每个target。在执行之前，首先需要执行它所依赖的target。程序中的名为run的target的depends属性compile，而名为compile的target的 depends属性是prepare，所以这几个target执行的顺序是prepare->compile->run。一个target只能被执行一次，即使有多个target依赖于它。如果没有if或unless属性，target总会被执行。  
  
3.<mkdir>标签  
该标签用于创建一个目录，它有一个属性dir用来指定所创建的目录名，其代码如下：  
<mkdir dir=”＄{class.root}”/>  
通过以上代码就创建了一个目录，这个目录已经被前面的property标签所指定。  
  
4<jar>标签  
该标签用来生成一个JAR文件，其属性如下。  
(1) destfile表示JAR文件名。  
(2) basedir表示被归档的文件名。  
(3) includes表示别归档的文件模式。  
(4) exchudes表示被排除的文件模式。  
  
5．<javac标签>  
该标签用于编译一个或一组java文件，其属性如下。  
(1).srcdir表示源程序的目录。  
(2).destdir表示class文件的输出目录。  
(3).include表示被编译的文件的模式。  
(4).excludes表示被排除的文件的模式。  
(5).classpath表示所使用的类路径。  
(6).debug表示包含的调试信息。  
(7).optimize表示是否使用优化。  
(8).verbose 表示提供详细的输出信息。  
(9).fileonerror表示当碰到错误就自动停止。  
  
6．<java>标签  
该标签用来执行编译生成的.class文件，其属性如下。  
(1).classname 表示将执行的类名。  
(2).jar表示包含该类的JAR文件名。  
(3).classpath所表示用到的类路径。  
(4).fork表示在一个新的虚拟机中运行该类。  
(5).failonerror表示当出现错误时自动停止。  
(6).output 表示输出文件。  
(7).append表示追加或者覆盖默认文件。  
  
7.<delete>标签  
该标签用于删除一个文件或一组文件，去属性如下。  
(1)/file表示要删除的文件。  
(2).dir表示要删除的目录。  
(3).includeEmptyDirs 表示指定是否要删除空目录，默认值是删除。  
(4).failonerror 表示指定当碰到错误是否停止，默认值是自动停止。  
(5).verbose表示指定是否列出所删除的文件，默认值为不列出。  
  
8.<copy>标签  
该标签用于文件或文件集的拷贝，其属性如下。  
(1).file 表示源文件。  
(2).tofile 表示目标文件。  
(3).todir 表示目标目录。  
(4).overwrite 表示指定是否覆盖目标文件，默认值是不覆盖。  
(5).includeEmptyDirs 表示制定是否拷贝空目录，默认值为拷贝。  
(6).failonerror 表示指定如目标没有发现是否自动停止，默认值是停止。  
(7).verbose 表示制定是否显示详细信息，默认值不显示。  
  
Ant的数据类型  
在构建文件中为了标识文件或文件组，经常需要使用数据类型。数据类型包含在org.apache.tool.ant.types包中。下面简单介绍构建文件中常用的数据类型。  
  
1. argument 类型  
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。  
(1).values 是一个命令参数。如果参数种有空格，但又想将它作为单独一个值，则使用此属性。  
(2).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。  
(3).line表示用空格分隔的多个参数列表。  
(4).path表示路径。  
  
2.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。  
(1).file表示环境变量值得文件名。此文件名要被转换位一个绝对路径。  
(2).path表示环境变量的路径。Ant会将它转换为一个本地约定。  
(3).value 表示环境变量的一个直接变量。  
(4).key 表示环境变量名。  
注意     file path 或 value只能取一个。  
  
3.filelist类型  
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。  
(1).dir是用于计算绝对文件名的目录。  
(2).files 是用逗号分隔的文件名列表。  
(3).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)。  
  
4.fileset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。  
(1).dir表示fileset 的基目录。  
(2).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.  
(3).defaultexcludes 用来确定是否使用默认的排除模式，默认为true。  
(4).excludes 是用逗号分隔的需要派出的文件模式列表。  
(5).excludesfile 表示每行包含一个排除模式的文件的文件名。  
(6).includes 是用逗号分隔的，需要包含的文件模式列表。  
(7).includesfile 表示每行包括一个包含模式的文件名。  
  
5.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 和 excludesfile,与fileset相同。Patternset 还允许以下嵌套元素：include,exclude,includefile 和 excludesfile.  
  
6.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：  
(1).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。  
(2).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。  
(3).id是过滤器的唯一标志符。  
(4).refid是对构建文件中某处定义一个过滤器的引用。  
  
7.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。在用作揖个属性时，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。  
(1).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。  
(2).refid 是对当前构建文件中某处定义的一个path的引用。  
(3).path表示一个文件或路径名列表。  
  
8.mapper类型  
Mapper类型定义了一组输入文件和一组输出文件间的关系，其属性如下。  
(1).classname 表示实现mapper类的类名。当内置mapper不满足要求时，用于创建定制mapper.  
(2).classpath表示查找一个定制mapper时所用的类型路径。  
(3).classpathref是对某处定义的一个类路径的引用。  
(4).from属性的含义取决于所用的mapper.  
(5).to属性的含义取决于所用的mapper.  
(6).type属性的取值为identity，flatten glob merge     regexp  其中之一，它定义了要是用的内置mapper的类型。  
  
Ant 的运行  
安装好Ant并且配置好路径之后，在命令行中切换到构建文件的目录，输入Ant命令就可以运行Ant.若没有指定任何参数，Ant会在当前目录下查询 build.xml文件。如果找到了就用该文件作为构建文件。如果使用了 –find 选项，Ant 就会在上级目录中找构建文件，直至到达文件系统得跟目录。如果构建文件的名字不是build.xml ，则Ant运行的时候就可以使用 –buildfile file ,这里file 指定了要使用的构建文件的名称，示例如下：Ant  
如下说明了表示当前目录的构建文件为build.xml 运行 ant 执行默认的目标。Ant –buildfile     test.xml  
使用当前目录下的test.xml 文件运行Ant ,执行默认的目标.

Xml代码

1. <?xml version="1.0" encoding="GB2312" ?>
2. <!--
3. =======================================================================
4. hello-ant 项目 ,学习ant工具的build file.
6. 参照ant的jakarta-ant-1.6alpha的build.xml
8. Copyright (c) 2002 The Neusoft Software Foundation. All rights
9. reserved.
11. =======================================================================
12. -->
13. <!--
14. 文档结构为:
15. <project>
16. <property/> 全局变量的定义
17. <property/>...
19. <target name="1"> 任务组(tasks)
20. <javac></javac> 一项javac任务
21. ...
22. <oneTask></ontTask> 一项其它任务
23. </target>
25. <target name="2">
26. <javac></javac>
27. ...
28. <oneTask></ontTask>
29. </target>
30. </project>
32. project代表一个项目，
33. default:运行到名称为"dist"的target(任务组)
34. basedir:基准路径。
35. -->
36. <project default="dist" basedir=".">
38. <!--
39. ===================================================================
40. 定义属性（property tasks）
41. 最好把用到的路径呀，名称呀都在这里定义成全局变量
42. 例：定义
43. <property name="a" value="hello"/>
44. 以后就可以这样用它：
45. <property name="b" value="${a}/b"/>
46. 现在:b=="hello/b"
47. ===================================================================
48. -->
50. <!--主要的系统环境属性-->
51. <property environment="env"/><!--取window,unix...的环境变量-->
52. <property name="java.home" value="${env.JAVA\_HOME}"/>
53. <property name="ant.home" value="${env.ANT\_HOME}"/>
55. <!--主要的app环境属性-->
56. <property name="app.name" value="hello-ant"/>
57. <property name="app.jar" value="${app.name}.jar"/>
58. <property name="app.copyright" value=" Copyright (c) 2002 The Neusoft Software Foundation. All rights reserved."/>
60. <!--app中src的属性-->
61. <property name="src.dir" value="src" />
62. <property name="src.main" value="${src.dir}/main"/>
63. <property name="src.script" value="${src.dir}/script"/>
65. <!--app用到的lib-->
66. <property name="lib.dir" value="lib"/>
68. <!--app的build目录中-->
69. <property name="build.dir" value="build" />
70. <property name="build.classes" value="${build.dir}/classes"/>
71. <property name="build.docs" value="${build.dir}/docs"/>
72. <property name="build.docs.api" value="${build.docs}/api"/>
73. <property name="build.lib" value="${build.dir}/lib"/>
75. <!--app的dist (distribution) 目录中-->
76. <property name="dist.dir" value="dist"/>
77. <property name="dist.bin" value="${dist.dir}/bin"/>
78. <property name="dist.docs" value="${dist.dir}/docs"/>
79. <property name="dist.lib" value="${dist.dir}/lib"/>
81. <!--app的docs目录中-->
82. <property name="docs.dir" value="docs"/>
84. <!--
85. 定义一组路径以后可以通过id重用这组路径 ，例：
86. <javac srcdir="src/main" destdir="build/classes">
87. <classpath refid="classpath"/>
88. </javac>
89. -->
90. <path id="classpath">
91. <!--本项目只有一个java，用不上classpath，这里只是做个例子-->
92. <pathelement location="${build.classes}"/>
93. <pathelement path="${java.home}/lib/tools.jar"/>
94. </path>
96. <!--
97. ===================================================================
98. init 准备目录(File Tasks)
99. 主要的目录结构通常是不会变的，一起生成他们
100. ===================================================================
101. -->
102. <target name="init">
103. <!--清除以前目录-->
104. <delete dir="${build.dir}" failonerror="false" />
105. <delete dir="${dist.dir}" failonerror="false"/>
107. <!--准备目录-->
108. <mkdir dir="${build.dir}"/>
109. <mkdir dir="${build.classes}"/>
110. <mkdir dir="${build.docs}"/>
111. <mkdir dir="${build.docs.api}"/>
112. <mkdir dir="${build.lib}"/>
114. <mkdir dir="${dist.dir}"/>
115. <mkdir dir="${dist.bin}"/>
116. <mkdir dir="${dist.lib}"/>
118. </target>
120. <!--
121. ===================================================================
122. Build the code (Compile Tasks,File Tasks)
123. ===================================================================
124. -->
125. <target name="build" depends="init">
126. <!--编译-->
127. <javac srcdir="${src.main}" destdir="${build.classes}">
128. <classpath refid="classpath"/>
129. </javac>
130. </target>
132. <!--
133. ===================================================================
134. 打包文档(Archive Tasks)
135. Create the project jars: xxx1.jar and xxx2.jar
136. ===================================================================
137. -->
138. <target name="jars" depends="build">
139. <jar basedir="${build.classes}" jarfile="${build.lib}/${app.jar}"/>
140. </target>
142. <!--
143. ===================================================================
144. Creates the API documentation
145. ===================================================================
146. -->
147. <target name="javadocs"
148. depends="jars"
149. description="--> creates the API documentation">
150. <!--copy docs 手册... -->
151. <copy todir="${build.docs}">
152. <fileset dir="${docs.dir}"/>
153. </copy>
155. <javadoc packagenames="hello.ant.\*"
156. sourcepath="${src.main}"
157. defaultexcludes="yes"
158. destdir="${build.docs.api}"
159. author="true"
160. version="true"
161. use="true"
162. windowtitle="Docs API">
163. <doctitle><![CDATA[<h1>hello ant Docs API</h1>]]></doctitle>
164. <bottom><![CDATA[<i>${app.copyright}</i>]]></bottom>
165. <tag name="todo" scope="all" description="To do:" />
166. </javadoc>
167. </target>
169. <!--
170. ===================================================================
171. Create the distribution that can run (Archive Tasks)
172. 主要是从各目录中把该copy的copy上
173. ===================================================================
174. -->
175. <target name="dist" depends="javadocs">
176. <!--copy bin 执行文件 -->
177. <copy todir="${dist.bin}">
178. <fileset dir="${src.script}/"/>
179. </copy>
180. <copy todir="${dist.docs}">
181. <fileset dir="${build.docs}/"/>
182. </copy>
183. <!-- copy lib 文件 -->
184. <copy todir="${dist.lib}">
185. <fileset dir="${build.lib}/"/>
186. </copy>
188. </target>
189. <!--
190. ===================================================================
191. Cleans everything(File Tasks)
192. 例如可以删除build中的文件
193. ===================================================================
194. -->
195. </project>

<?xml version="1.0" encoding="GB2312" ?>

<!--

=======================================================================

hello-ant 项目 ,学习ant工具的build file.

参照ant的jakarta-ant-1.6alpha的build.xml

Copyright (c) 2002 The Neusoft Software Foundation. All rights

reserved.

=======================================================================

-->

<!--

文档结构为:

<project>

<property/> 全局变量的定义

<property/>...

<target name="1"> 任务组(tasks)

<javac></javac> 一项javac任务

...

<oneTask></ontTask> 一项其它任务

</target>

<target name="2">

<javac></javac>

...

<oneTask></ontTask>

</target>

</project>

project代表一个项目，

default:运行到名称为"dist"的target(任务组)

basedir:基准路径。

-->

<project default="dist" basedir=".">

<!--

===================================================================

定义属性（property tasks）

最好把用到的路径呀，名称呀都在这里定义成全局变量

例：定义

<property name="a" value="hello"/>

以后就可以这样用它：

<property name="b" value="${a}/b"/>

现在:b=="hello/b"

===================================================================

-->

<!--主要的系统环境属性-->

<property environment="env"/><!--取window,unix...的环境变量-->

<property name="java.home" value="${env.JAVA\_HOME}"/>

<property name="ant.home" value="${env.ANT\_HOME}"/>

<!--主要的app环境属性-->

<property name="app.name" value="hello-ant"/>

<property name="app.jar" value="${app.name}.jar"/>

<property name="app.copyright" value=" Copyright (c) 2002 The Neusoft Software Foundation. All rights reserved."/>

<!--app中src的属性-->

<property name="src.dir" value="src" />

<property name="src.main" value="${src.dir}/main"/>

<property name="src.script" value="${src.dir}/script"/>

<!--app用到的lib-->

<property name="lib.dir" value="lib"/>

<!--app的build目录中-->

<property name="build.dir" value="build" />

<property name="build.classes" value="${build.dir}/classes"/>

<property name="build.docs" value="${build.dir}/docs"/>

<property name="build.docs.api" value="${build.docs}/api"/>

<property name="build.lib" value="${build.dir}/lib"/>

<!--app的dist (distribution) 目录中-->

<property name="dist.dir" value="dist"/>

<property name="dist.bin" value="${dist.dir}/bin"/>

<property name="dist.docs" value="${dist.dir}/docs"/>

<property name="dist.lib" value="${dist.dir}/lib"/>

<!--app的docs目录中-->

<property name="docs.dir" value="docs"/>

<!--

定义一组路径以后可以通过id重用这组路径 ，例：

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

<classpath refid="classpath"/>

</javac>

-->

<path id="classpath">

<!--本项目只有一个java，用不上classpath，这里只是做个例子-->

<pathelement location="${build.classes}"/>

<pathelement path="${java.home}/lib/tools.jar"/>

</path>

<!--

===================================================================

init 准备目录(File Tasks)

主要的目录结构通常是不会变的，一起生成他们

===================================================================

-->

<target name="init">

<!--清除以前目录-->

<delete dir="${build.dir}" failonerror="false" />

<delete dir="${dist.dir}" failonerror="false"/>

<!--准备目录-->

<mkdir dir="${build.dir}"/>

<mkdir dir="${build.classes}"/>

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

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

<mkdir dir="${build.lib}"/>

<mkdir dir="${dist.dir}"/>

<mkdir dir="${dist.bin}"/>

<mkdir dir="${dist.lib}"/>

</target>

<!--

===================================================================

Build the code (Compile Tasks,File Tasks)

===================================================================

-->

<target name="build" depends="init">

<!--编译-->

<javac srcdir="${src.main}" destdir="${build.classes}">

<classpath refid="classpath"/>

</javac>

</target>

<!--

===================================================================

打包文档(Archive Tasks)

Create the project jars: xxx1.jar and xxx2.jar

===================================================================

-->

<target name="jars" depends="build">

<jar basedir="${build.classes}" jarfile="${build.lib}/${app.jar}"/>

</target>

<!--

===================================================================

Creates the API documentation

===================================================================

-->

<target name="javadocs"

depends="jars"

description="--> creates the API documentation">

<!--copy docs 手册... -->

<copy todir="${build.docs}">

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

</copy>

<javadoc packagenames="hello.ant.\*"

sourcepath="${src.main}"

defaultexcludes="yes"

destdir="${build.docs.api}"

author="true"

version="true"

use="true"

windowtitle="Docs API">

<doctitle><![CDATA[<h1>hello ant Docs API</h1>]]></doctitle>

<bottom><![CDATA[<i>${app.copyright}</i>]]></bottom>

<tag name="todo" scope="all" description="To do:" />

</javadoc>

</target>

<!--

===================================================================

Create the distribution that can run (Archive Tasks)

主要是从各目录中把该copy的copy上

===================================================================

-->

<target name="dist" depends="javadocs">

<!--copy bin 执行文件 -->

<copy todir="${dist.bin}">

<fileset dir="${src.script}/"/>

</copy>

<copy todir="${dist.docs}">

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

</copy>

<!-- copy lib 文件 -->

<copy todir="${dist.lib}">

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

</copy>

</target>

<!--

===================================================================

Cleans everything(File Tasks)

例如可以删除build中的文件

===================================================================

-->

</project>

趁热打铁，接着上面的再发一个实例：

Xml代码

1. <?xml version="1.0"?>
2. <project name="ssh" basedir="." default="usage">
3. <property name="name" value="ssh"/>
4. <property name="war.dir" value="war"/>
5. <property name="src.dir" value="src"/>
6. <property name="client.dir" value="client"/>
7. <property name="build.dir" value=".classes"/>
8. <property name="webcontent.dir" value="WebContent"/>
9. <property name="prjlib.dir" value="lib"/>
10. <property name="webcontentlib.dir" value="${webcontent.dir}/WEB-INF/lib"/>
11. <property name="weblib.dir" value="${war.dir}/WEB-INF/lib"/>
12. <property name="dist.dir" value="dist"/>
13. <property environment="env"/>
14. <property name="tomcat.home" value="${env.CATALINA\_HOME}"/>
15. <property name="webapp.dist" value="${dist.dir}/webapps"/>
16. <path id="master-classpath">
17. <fileset dir="${webcontentlib.dir}">
18. <include name="hibernate3.jar"/>
19. <include name="spring.jar"/>
20. <include name="struts.jar"/>
21. <include name="struts-el.jar"/>
22. <include name="struts-menu-2.4.2.jar"/>
23. <include name="acegi-security-1.0.2.jar"/>
24. <include name="activation.jar"/>
25. <include name="antlr.jar"/>
26. <include name="antlr-2.7.6.jar"/>
27. <include name="asm.jar"/>
28. <include name="aspectjweaver-1.5.2.jar"/>
29. <include name="cglib-2.1.3.jar"/>
30. <include name="commons-beanutils.jar"/>
31. <include name="commons-codec-1.3.jar"/>
32. <include name="commons-collections.jar"/>
33. <include name="commons-dbcp.jar"/>
34. <include name="commons-digester.jar"/>
35. <include name="commons-fileupload.jar"/>
36. <include name="commons-io.jar"/>
37. <include name="commons-lang.jar"/>
38. <include name="commons-logging-1.1.jar"/>
39. <include name="commons-pool.jar"/>
40. <include name="commons-validator.jar"/>
41. <include name="displaytag-1.1.jar"/>
42. <include name="dom4j-1.6.1.jar"/>
43. <include name="dwr.jar"/>
44. <include name="ehcache-1.2.3.jar"/>
45. <include name="itext-1.4.jar"/>
46. <include name="jakarta-oro.jar"/>
47. <include name="jstl.jar"/>
48. <include name="jta.jar"/>
49. <include name="log4j-1.2.11.jar"/>
50. <include name="mail.jar"/>
51. <include name="oscache-2.3.2.jar"/>
52. <include name="mysql-connector-java-5.0.3-bin.jar"/>
53. <include name="sitemesh-2.2.1.jar"/>
54. <include name="standard.jar"/>
55. <include name="urlrewrite-3.0-beta.jar"/>
56. <include name="velocity-1.4.jar"/>
57. <include name="velocity-tools-view-1.1.jar"/>
58. </fileset>
59. <fileset dir="${prjlib.dir}/servletapi-2.3">
60. <include name="servletapi-2.3.jar"/>
61. </fileset>
62. </path>
63. <target name="usage">
64. <echo message=""/>
65. <echo message="Spring JPetStore build file"/>
66. <echo message="------------------------------------------------------"/>
67. <echo message=""/>
68. <echo message="Available targets are:"/>
69. <echo message=""/>
70. <echo message="clean      --> Clean output dirs"/>
71. <echo message="build      --> Compile main Java sources and copy libraries"/>
72. <echo message="warfile --> Build the web application archive"/>
73. <echo message="all        --> Clean, build, warfile"/>
74. <echo message=""/>
75. </target>
76. <target name="clean" description="Clean output dirs (build, weblib, dist)">
77. <delete dir="${build.dir}"/>
78. <delete dir="${weblib.dir}"/>
79. <delete dir="${war.dir}"/>
80. <delete dir="${dist.dir}"/>
81. <delete file="client/${name}.jar"/>
82. </target>
83. <target name="build" description="Compile main source tree java files into class files, generate jar files">
84. <mkdir dir="${build.dir}"/>
85. <mkdir dir="${war.dir}"/>
86. <javac destdir="${build.dir}" source="1.3" target="1.3" debug="true"
87. deprecation="false" optimize="false" failonerror="true">
88. <src path="${src.dir}"/>
89. <classpath refid="master-classpath"/>
90. </javac>
91. <mkdir dir="${weblib.dir}"/>
92. <mkdir dir="${war.dir}/WEB-INF/classes"/>
93. <jar jarfile="${weblib.dir}/${name}.jar" compress="true" basedir="${build.dir}"/>
94. <copy todir="${war.dir}" preservelastmodified="true">
95. <fileset dir="${webcontent.dir}">
96. <include name="\*\*/\*\*"/>
97. <include name="\*\*.\*\*"/>
98. </fileset>
99. </copy>
100. <copy todir="${war.dir}/WEB-INF/classes" preservelastmodified="true">
101. <fileset dir="${src.dir}">
102. <include name="\*.xml"/>
103. <include name="\*\*/\*.properties"/>
104. <include name="\*\*/\*.vm"/>
105. <exclude name="\*\*/.\*"/>
106. </fileset>
107. </copy>
109. <copy file="${weblib.dir}/${name}.jar" tofile="${client.dir}/${name}.jar"/>
110. </target>
111. <target name="dist" depends="warfile">
112. <!--
113. Delegate to warfile target by depending on it. dist is just to offer
114. a generic target name across all Spring sample apps that may be used
115. for autobuilds testing.
116. -->
117. </target>
118. <target name="warfile" depends="build" description="Build the web application archive">
119. <mkdir dir="${dist.dir}"/>
120. <war warfile="${dist.dir}/${name}.war" basedir="${war.dir}" webxml="${webcontent.dir}/WEB-INF/web.xml">
121. <include name="\*"/>
122. <include name="images/\*\*"/>
123. <include name="common/\*\*"/>
124. <include name="decorators/\*\*"/>
125. <include name="scripts/\*\*"/>
126. <include name="styles/\*\*"/>
127. <include name="WEB-INF/\*.\*"/>
128. <include name="WEB-INF/lib/\*\*"/>
129. <include name="WEB-INF/pages/\*\*"/>
130. <include name="WEB-INF/classes/\*\*"/>
131. <exclude name="WEB-INF/web.xml"/>
132. </war>
133. </target>
134. <target name="all" depends="clean,build,warfile" description="Clean,build,warfile"/>
135. </project>

<?xml version="1.0"?>

<project name="ssh" basedir="." default="usage">

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

      <property name="war.dir" value="war"/>

      <property name="src.dir" value="src"/>

      <property name="client.dir" value="client"/>

      <property name="build.dir" value=".classes"/>

      <property name="webcontent.dir" value="WebContent"/>

      <property name="prjlib.dir" value="lib"/>

      <property name="webcontentlib.dir" value="${webcontent.dir}/WEB-INF/lib"/>

      <property name="weblib.dir" value="${war.dir}/WEB-INF/lib"/>

      <property name="dist.dir" value="dist"/>

      <property environment="env"/>

      <property name="tomcat.home" value="${env.CATALINA\_HOME}"/>

      <property name="webapp.dist" value="${dist.dir}/webapps"/>

      <path id="master-classpath">

          <fileset dir="${webcontentlib.dir}">

              <include name="hibernate3.jar"/>

              <include name="spring.jar"/>

              <include name="struts.jar"/>

              <include name="struts-el.jar"/>

              <include name="struts-menu-2.4.2.jar"/>

              <include name="acegi-security-1.0.2.jar"/>

              <include name="activation.jar"/>

              <include name="antlr.jar"/>

              <include name="antlr-2.7.6.jar"/>

              <include name="asm.jar"/>

              <include name="aspectjweaver-1.5.2.jar"/>

              <include name="cglib-2.1.3.jar"/>

              <include name="commons-beanutils.jar"/>

              <include name="commons-codec-1.3.jar"/>

              <include name="commons-collections.jar"/>

              <include name="commons-dbcp.jar"/>

              <include name="commons-digester.jar"/>

              <include name="commons-fileupload.jar"/>

              <include name="commons-io.jar"/>

              <include name="commons-lang.jar"/>

              <include name="commons-logging-1.1.jar"/>

              <include name="commons-pool.jar"/>

              <include name="commons-validator.jar"/>

              <include name="displaytag-1.1.jar"/>

              <include name="dom4j-1.6.1.jar"/>

              <include name="dwr.jar"/>

              <include name="ehcache-1.2.3.jar"/>

              <include name="itext-1.4.jar"/>

              <include name="jakarta-oro.jar"/>

              <include name="jstl.jar"/>

              <include name="jta.jar"/>

              <include name="log4j-1.2.11.jar"/>

              <include name="mail.jar"/>

              <include name="oscache-2.3.2.jar"/>

              <include name="mysql-connector-java-5.0.3-bin.jar"/>

              <include name="sitemesh-2.2.1.jar"/>

              <include name="standard.jar"/>

              <include name="urlrewrite-3.0-beta.jar"/>

              <include name="velocity-1.4.jar"/>

              <include name="velocity-tools-view-1.1.jar"/>

          </fileset>

          <fileset dir="${prjlib.dir}/servletapi-2.3">

              <include name="servletapi-2.3.jar"/>

          </fileset>

      </path>

      <target name="usage">

          <echo message=""/>

          <echo message="Spring JPetStore build file"/>

          <echo message="------------------------------------------------------"/>

          <echo message=""/>

          <echo message="Available targets are:"/>

          <echo message=""/>

          <echo message="clean      --> Clean output dirs"/>

          <echo message="build      --> Compile main Java sources and copy libraries"/>

          <echo message="warfile --> Build the web application archive"/>

          <echo message="all        --> Clean, build, warfile"/>

          <echo message=""/>

      </target>

      <target name="clean" description="Clean output dirs (build, weblib, dist)">

          <delete dir="${build.dir}"/>

          <delete dir="${weblib.dir}"/>

          <delete dir="${war.dir}"/>

          <delete dir="${dist.dir}"/>

          <delete file="client/${name}.jar"/>

      </target>

      <target name="build" description="Compile main source tree java files into class files, generate jar files">

          <mkdir dir="${build.dir}"/>

          <mkdir dir="${war.dir}"/>

          <javac destdir="${build.dir}" source="1.3" target="1.3" debug="true"

                  deprecation="false" optimize="false" failonerror="true">

              <src path="${src.dir}"/>

              <classpath refid="master-classpath"/>

          </javac>

          <mkdir dir="${weblib.dir}"/>

          <mkdir dir="${war.dir}/WEB-INF/classes"/>

          <jar jarfile="${weblib.dir}/${name}.jar" compress="true" basedir="${build.dir}"/>

          <copy todir="${war.dir}" preservelastmodified="true">

              <fileset dir="${webcontent.dir}">

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

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

              </fileset>

          </copy>

          <copy todir="${war.dir}/WEB-INF/classes" preservelastmodified="true">

              <fileset dir="${src.dir}">

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

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

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

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

              </fileset>

          </copy>

          <copy file="${weblib.dir}/${name}.jar" tofile="${client.dir}/${name}.jar"/>

      </target>

      <target name="dist" depends="warfile">

          <!--

              Delegate to warfile target by depending on it. dist is just to offer

              a generic target name across all Spring sample apps that may be used

              for autobuilds testing.

          -->

      </target>

      <target name="warfile" depends="build" description="Build the web application archive">

          <mkdir dir="${dist.dir}"/>

          <war warfile="${dist.dir}/${name}.war" basedir="${war.dir}" webxml="${webcontent.dir}/WEB-INF/web.xml">

              <include name="\*"/>

              <include name="images/\*\*"/>

              <include name="common/\*\*"/>

              <include name="decorators/\*\*"/>

              <include name="scripts/\*\*"/>

              <include name="styles/\*\*"/>

              <include name="WEB-INF/\*.\*"/>

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

              <include name="WEB-INF/pages/\*\*"/>

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

              <exclude name="WEB-INF/web.xml"/>

          </war>

      </target>

      <target name="all" depends="clean,build,warfile" description="Clean,build,warfile"/>

</project>

在Eclipse中使用Ant Ant是Java平台下非常棒的批处理命令执行程序，能非常方便地自动完成编译，测试，打包，部署等等一系列任务，大大提高开发效率。如果你现在还没有开始使用Ant，那就要赶快开始学习使用，使自己的开发水平上一个新台阶。  
  
Eclipse中已经集成了Ant，我们可以直接在Eclipse中运行Ant。  
  
以前面建立的Hello工程为例，创建以下目录结构：  
  
  
  
新建一个build.xml，放在工程根目录下。build.xml定义了Ant要执行的批处理命令。虽然Ant也可以使用其它文件名，但是遵循标准能更使开发更规范，同时易于与别人交流。  
通常，src存放Java源文件，classes存放编译后的class文件，lib存放编译和运行用到的所有jar文件，web存放JSP等web文件，dist存放打包后的jar文件，doc存放API文档。  
然后在根目录下创建build.xml文件，输入以下内容：

Xml代码

1. <?xml version="1.0"?>
2. <project name="Hello world" default="doc">
3. <!-- properies -->
4. <property name="src.dir" value="src" />
5. <property name="report.dir" value="report" />
6. <property name="classes.dir" value="classes" />
7. <property name="lib.dir" value="lib" />
8. <property name="dist.dir" value="dist" />
9. <property name="doc.dir" value="doc"/>
10. <!-- 定义classpath -->
11. <path id="master-classpath">
12. <fileset file="${lib.dir}/\*.jar" />
13. <pathelement path="${classes.dir}"/>
14. </path>
15. <!-- 初始化任务 -->
16. <target name="init">
17. </target>
18. <!-- 编译 -->
19. <target name="compile" depends="init" description="compile the source files">
20. <mkdir dir="${classes.dir}"/>
21. <javac srcdir="${src.dir}" destdir="${classes.dir}" target="1.4">
22. <classpath refid="master-classpath"/>
23. </javac>
24. </target>
25. <!-- 测试 -->
26. <target name="test" depends="compile" description="run junit test">
27. <mkdir dir="${report.dir}"/>
28. <junit printsummary="on"
29. haltonfailure="false"
30. failureproperty="tests.failed"
31. showoutput="true">
32. <classpath refid="master-classpath" />
33. <formatter type="plain"/>
34. <batchtest todir="${report.dir}">
35. <fileset dir="${classes.dir}">
36. <include name="\*\*/\*Test.\*"/>
37. </fileset>
38. </batchtest>
39. </junit>
40. <fail if="tests.failed">
41. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
42. \*\*\*\*   One or more tests failed!   Check the output ...   \*\*\*\*
43. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
44. </fail>
45. </target>
46. <!-- 打包成jar -->
47. <target name="pack" depends="test" description="make .jar file">
48. <mkdir dir="${dist.dir}" />
49. <jar destfile="${dist.dir}/hello.jar" basedir="${classes.dir}">
50. <exclude name="\*\*/\*Test.\*" />
51. <exclude name="\*\*/Test\*.\*" />
52. </jar>
53. </target>
54. <!-- 输出api文档 -->
55. <target name="doc" depends="pack" description="create api doc">
56. <mkdir dir="${doc.dir}" />
57. <javadoc destdir="${doc.dir}"
58. author="true"
59. version="true"
60. use="true"
61. windowtitle="Test API">
62. <packageset dir="${src.dir}" defaultexcludes="yes">
63. <include name="example/\*\*" />
64. </packageset>
65. <doctitle><![CDATA[<h1>Hello, test</h1>]]></doctitle>
66. <bottom><![CDATA[<i>All Rights Reserved.</i>]]></bottom>
67. <tag name="todo" scope="all" description="To do:" />
68. </javadoc>
69. </target>
70. </project>

<?xml version="1.0"?>

<project name="Hello world" default="doc">

<!-- properies -->

     <property name="src.dir" value="src" />

     <property name="report.dir" value="report" />

     <property name="classes.dir" value="classes" />

     <property name="lib.dir" value="lib" />

     <property name="dist.dir" value="dist" />

<property name="doc.dir" value="doc"/>

     <!-- 定义classpath -->

     <path id="master-classpath">

         <fileset file="${lib.dir}/\*.jar" />

         <pathelement path="${classes.dir}"/>

     </path>

     <!-- 初始化任务 -->

     <target name="init">

     </target>

     <!-- 编译 -->

     <target name="compile" depends="init" description="compile the source files">

         <mkdir dir="${classes.dir}"/>

         <javac srcdir="${src.dir}" destdir="${classes.dir}" target="1.4">

             <classpath refid="master-classpath"/>

         </javac>

     </target>

     <!-- 测试 -->

     <target name="test" depends="compile" description="run junit test">

         <mkdir dir="${report.dir}"/>

         <junit printsummary="on"

                 haltonfailure="false"

                 failureproperty="tests.failed"

                 showoutput="true">

             <classpath refid="master-classpath" />

             <formatter type="plain"/>

             <batchtest todir="${report.dir}">

                 <fileset dir="${classes.dir}">

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

                 </fileset>

             </batchtest>

         </junit>

         <fail if="tests.failed">

         \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

         \*\*\*\*   One or more tests failed!   Check the output ...   \*\*\*\*

         \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

         </fail>

     </target>

     <!-- 打包成jar -->

     <target name="pack" depends="test" description="make .jar file">

      <mkdir dir="${dist.dir}" />

         <jar destfile="${dist.dir}/hello.jar" basedir="${classes.dir}">

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

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

         </jar>

     </target>

     <!-- 输出api文档 -->

     <target name="doc" depends="pack" description="create api doc">

      <mkdir dir="${doc.dir}" />

      <javadoc destdir="${doc.dir}"

             author="true"

             version="true"

             use="true"

             windowtitle="Test API">

             <packageset dir="${src.dir}" defaultexcludes="yes">

                 <include name="example/\*\*" />

             </packageset>

             <doctitle><![CDATA[<h1>Hello, test</h1>]]></doctitle>

             <bottom><![CDATA[<i>All Rights Reserved.</i>]]></bottom>

             <tag name="todo" scope="all" description="To do:" />

         </javadoc>

     </target>

</project>

以上xml依次定义了init（初始化），compile（编译），test（测试），doc（生成文档），pack（打包）任务，可以作为模板。  
选中Hello工程，然后选择“Project”，“Properties”，“Builders”，“New…”，选择“Ant Build”：  
  
填入Name：Ant\_Builder；Buildfile：build.xml；BaseDirectory：${workspace\_loc: /Hello}（按“BrowseWorkspace”选择工程根目录），由于用到了junit.jar包，搜索Eclipse目录，找到 junit.jar，把它复制到Hello/lib目录下，并添加到Ant的Classpath中：  
  
然后在Builder面板中钩上Ant\_Build，去掉Java Builder：  
  
再次编译，即可在控制台看到Ant的输出：  
Buildfile: F:\eclipse-projects\Hello\build.xml  
init:  
compile:  
        [mkdir] Created dir: F:\eclipse-projects\Hello\classes  
        [javac] Compiling 2 source files to F:\eclipse-projects\Hello\classes  
test:  
        [mkdir] Created dir: F:\eclipse-projects\Hello\report  
        [junit] Running example.HelloTest  
        [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.02 sec  
pack:  
        [mkdir] Created dir: F:\eclipse-projects\Hello\dist  
          [jar] Building jar: F:\eclipse-projects\Hello\dist\hello.jar  
doc:  
        [mkdir] Created dir: F:\eclipse-projects\Hello\doc  
      [javadoc] Generating Javadoc  
      [javadoc] Javadoc execution  
      [javadoc] Loading source files for package example...  
      [javadoc] Constructing Javadoc information...  
      [javadoc] Standard Doclet version 1.4.2\_04  
      [javadoc] Building tree for all the packages and classes...  
      [javadoc] Building index for all the packages and classes...  
      [javadoc] Building index for all classes...  
      [javadoc] Generating F:\eclipse-projects\Hello\doc\stylesheet.css...  
      [javadoc] Note: Custom tags that could override future standardtags:   @todo. To avoid potential overrides, use at least one periodcharacter (.) in custom tag names.  
      [javadoc] Note: Custom tags that were not seen:   @todo  
BUILD SUCCESSFUL  
Total time: 11 seconds  
Ant依次执行初始化，编译，测试，打包，生成API文档一系列任务，极大地提高了开发效率。将来开发J2EE项目时，还可加入部署等任务。并且，即使脱离了Eclipse环境，只要正确安装了Ant，配置好环境变量ANT\_HOME=<Ant解压目录&gt;， Path=…;%ANT\_HOME%\bin，在命令行提示符下切换到Hello目录，简单地键入ant即可。