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 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   
一个项目标签下可以有一个或多个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。下面镜简单介绍构建文件中一些常用的数据类型。

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

**ant的build.xml文件详解（二）**

以build\_for\_ejb\_templet.xml为示例，讲解Ant中常用的元素和任务。   
  
约定： "…"表示这里有很多代码，未列出   
  
build\_for\_ejb\_templet.xml任务的分为以下几大部分   
i. 开始   
ii. 初始化   
iii. 定义classpath   
iv. 为编译作准备   
v. 编译EJB部分   
vi. 编译WEB部分   
vii. 编译J2EE Application   
viii. 部署Application   
ix. 创建组件的API   
x. 确定build的目标   
  
build\_for\_ejb\_templet.xml的讲解   
  
**开始**  
<?xml version="1.0" encoding="UTF-8"?>   
讲解：encoding="UTF-8"表示XML文件采用UTF-8编码格式，如果要使用GBK编码，需定义为encodeing="GBK"。  
  
<!--    
   Build file for 'componentName'    
   Creation date : $Date: yyyy-m-d $    
   Updated date : $Date: yyyy-m-d $    
   Author:   developerName  
   Copyright 2002 CompanyName, Inc. All rights reserved.    
-->    
讲解：此部分为文件内容的简介，包括文件名称、创建日期、最后修改日期、创建文件的作者、版权。  
componentName 表示 文件名称  
yyyy-m-dd 表示 创建日期、最后修改日期的格式，如2002-5-1  
developerName 表示 创建文件的作者  
CompanyName 表示 公司名称或URL  
  
<project name="componentName" default="core" basedir=".">  
…  
</project>  
讲解：此部分定义了一个名称为componentName的项目元素，缺省的任务为"core"任务，根目录为当前目录。componentName表示组件的名称，这里指EJB的名称。  
  
  
  
**初始化**  
<target name="init">  
…  
</target>  
讲解：此部分用于初始化所有的变量  
  
<property file="../../build.properties" />  
讲解：导入项目根目录下build.properties中的全局变量，开发人员也可以在此文件中重新定义全局变量。  
  
<property name="jboss.lib" value="${jboss.home}/lib" />    
<property name="jboss.client" value="${jboss.home}/client" />    
<property name="jboss.deploy" value="${jboss.home}/server/${jboss.configuration}/deploy" />    
…  
<property name="jboss.dir"       value="${jboss.home}" />    
…  
<property name="deploy.ormi"     value=""/>    
<property name="deploy.username" value=""/>    
<property name="deploy.password" value=""/>  
讲解：定义和Jboss Application Server有关的变量，包括lib目录、client目录、  
deploy目录和J2EE Application部署要用到的一些变量。  
  
<property name="name"   value="componentName"/>    
讲解：定义组件的名称  
  
<property name="src.dir" value="${basedir}/src"/>    
讲解：定义源代码目录路径  
  
<property name="etc.dir" value="${basedir}/etc"/>    
讲解：定义资源目录路径  
  
<property name="lib.dir" value="${basedir}/lib"/>    
讲解：定义库目录路径  
  
<property name="build.dir" value="${basedir}/build"/>    
讲解：定义build目录路径  
  
<property name="src.main.dir" value="${src.dir}/main"/>  
讲解：定义源代码的主目录路径  
  
<property name="src.ejb.dir" value="${src.main.dir}/ejb"/>    
讲解：定义存放EJB的源代码目录路径  
  
<property name="src.javabean.dir" value="${src.main.dir}/javabean"/>  
讲解：定义存放JavaBean的源代码目录路径  
  
<property name="src.servlet.dir" value="${src.main.dir}/servlet"/>  
讲解：定义存放Servlet的源代码目录路径  
  
<property name="src.web.dir" value="${src.main.dir}/web"/>    
讲解：定义存放WEB部分文件（包括JSP程序、HTML文件、图片、CSS文件、JS脚本等）的目录路径  
  
<property name="javadoc.dir" value="${lib.dir}/docs/api"/>   
讲解：定义存放组件API的开发文件目录路径  
  
<property name="ejb-classes.dest" value="${lib.dir}/ejb"/>    
讲解：定义存放EJB的编译代码目录路径  
  
<property name="javabean-classes.dest" value="${lib.dir}/javabean"/>    
讲解：定义存放JavaBean的编译代码目录路径  
  
<property name="web-classes.dest" value="${lib.dir}/web/WEB-INF/classes" />  
讲解：定义WEB目录的类目录路径  
  
<property name="web-lib.dest" value="${lib.dir}/web/WEB-INF/lib" />    
讲解：定义WEB目录的库目录名称  
  
<property name="pkg-dist.name" value="${name}-pkg"/>    
讲解：定义压缩文档的名称  
  
<property name="ProjectName.components.home" value="../../components" />    
讲解：定义项目的组件目录路径  
  
<!-- Define componentName Component -->  
讲解：这段为注释，说明以下是定义当前组件的变量  
  
<property name="ProjectName.componentName.name" value="componentName"/>  
讲解：定义当前组件的名称  
  
<property name="ProjectName.componentName.home" value="${ProjectName.components.home}/componentName"/>  
讲解：定义当前组件的目录路径  
  
<property name="ProjectName.componentName.classbindir"   
value="${ProjectName.componentName.home}/lib/ejb"/>  
讲解：定义当前组件的EJB编译目录路径  
<property name="ProjectName.componentName.ejbjar"   
value="${ProjectName.componentName.home}/build/componentName-ejb.jar"/>  
讲解：定义当前组件内的EJB包的路径  
  
<property name="ProjectName.componentName.ejbjar.client"   
value="${ProjectName.componentName.home}/build/componentName-ejb-client.jar"/>  
讲解：定义当前组件内的EJB客户端包的路径  
  
<!-- Define referencesComponentName Component -->  
讲解：这段为注释，说明以下是定义引入其它组件的变量  
  
<property name="ProjectName.referencesComponentName.name"   
value="referencesComponentName"/>  
讲解：定义指定组件的名称  
  
<property name="ProjectName.referencesComponentName.home"   
value="${ProjectName.components.home}/referencesComponentName"/>  
讲解：定义指定组件的目录路径  
  
<property name="ProjectName.referencesComponentName.classbindir"   
value="${ProjectName.referencesComponentName.home}/lib/ejb"/>  
讲解：定义指定组件的EJB编译目录路径  
  
<property name="ProjectName.referencesComponentName.ejbjar"   
value="${ProjectName.referencesComponentName.home}  
/build/referencesComponentName-ejb.jar"/>  
讲解：定义指定组件内的EJB包的路径  
  
<property name="ProjectName.referencesComponentName.ejbjar.client"   
value="${ProjectName.referencesComponentName.home}  
/build/referencesComponentName-ejb-client.jar"/>  
讲解：定义指定组件内的EJB客户端包的路径  
  
<property name="build.classpath"   
value="${jboss.client}/jboss-j2ee.jar:${jboss.client}  
/jnp-client.jar:${jboss.client}/jnp-client.jar:${jboss.client}  
/jbossmq-client.jar:${jboss.client}/jbosssx-client.jar:${jboss.client}  
/concurrent.jar:${jboss.client}/jaas.jar:${jboss.lib}  
/jboss-jmx.jar:${jboss.home}/server/${jboss.configuration}  
/lib/jbosssx.jar:${jboss.home}/server/${jboss.configuration}  
/lib/mail.jar:${servlet-lib.path}:${ejb-classes.dest}:  
${web-classes.dest}:${ProjectName.componentName.classbindir}:  
${ProjectName.componentName.ejbjar.client}:  
${ProjectName.referencesComponentName.classbindir}:  
${ProjectName.referencesComponentName.ejbjar.client}" />  
讲解：定义classpath，编译bean时要用到。这是定义classpath的一种方法，下面还有另一种方法。  
  
**定义classpath**  
<!-- ================================================================== -->    
   <!-- Define the classpath for compile the component                     -->  
   <!-- ================================================================== -->    
   <path id="base.path">    
       <pathelement location="${jboss.client}/jboss-j2ee.jar" />    
       <pathelement location="${jboss.client}/jnp-client.jar" />    
       <pathelement location="${jboss.client}/jbossmq-client.jar" />    
       <pathelement location="${jboss.client}/jbosssx-client.jar" />    
       <pathelement location="${jboss.client}/concurrent.jar" />   
       <pathelement location="${jboss.client}/jaas.jar" />    
       <pathelement location="${jboss.lib}/jboss-jmx.jar" />    
       <pathelement   
location="${jboss.home}/server/${jboss.configuration}/lib/jbosssx.jar" />    
       <pathelement   
location="${jboss.home}/server/${jboss.configuration}/lib/mail.jar" />    
</path>    
讲解：此段定义应用服务器中包文件，如支持ejb的jboss-j2ee.jar、支持客户端程序的jnp-client.jar、jbossmq-client.jar、jbosssx-client.jar、支持JavaMail的mail.jar等。  
  
   <path id="project.path">    
       <path refid="base.path"/>  
       <pathelement location="${ProjectName.componentName.classbindir}"/>  
       <pathelement location="${ProjectName.componentName.ejbjar.client}"/>  
       <pathelement  
location="${ProjectName.referencesComponentName.classbindir}"/>  
       <pathelement location="${ProjectName.referencesComponentName.ejbjar.client}"/>  
   </path>    
讲解：此段定义项目中要用到的包文件。  
  
   <path id="web.path">  
         <path refid="project.path"/>  
         <pathelement location="${servlet-lib.path}"/>  
         <pathelement location="${ejb-classes.dest}"/>  
   </path>  
讲解：此段定义在编译servlet时的classpath，${ servlet-lib.path }是系统指定的Servlet引擎包。  
 **为编译作准备工作**  
<!--=============================================================== -->    
   <!-- Removes all created files and directories                       -->    
   <!-- ============================================================== -->    
   <target name="clean" depends="init">    
       <delete dir="${lib.dir}" />    
       <delete dir="${build.dir}" />    
   </target>  
讲解：清除build、lib目录下的所有文件和目录。  
  
   <!-- ================================================================ -->    
   <!-- Makes sure the needed directory structure is in place               -->    
   <!-- ================================================================ -->    
   <target name="prepare" depends="init,clean">    
       <mkdir dir="${lib.dir}" />    
       <mkdir dir="${lib.dir}/ejb" />    
       <mkdir dir="${lib.dir}/ejb/META-INF" />    
       <mkdir dir="${lib.dir}/web" />    
       <mkdir dir="${lib.dir}/web/WEB-INF" />    
       <mkdir dir="${lib.dir}/web/WEB-INF/classes" />    
       <mkdir dir="${lib.dir}/web/WEB-INF/lib" />    
       <mkdir dir="${lib.dir}/j2ee" />    
       <mkdir dir="${lib.dir}/META-INF" />    
       <mkdir dir="${lib.dir}/docs/api" />  
       <mkdir dir="${build.dir}" />    
   </target>    
讲解：创建build中要用到的所有目录，根据需要可以加入自定义的目录如：  
       <mkdir dir="${lib.dir}/javabean/ " />  
  
**编译EJB部分**  
<!-- ================================================================ -->    
   <!-- Compilation of the EJB part of the application                     -->    
   <!-- ================================================================ -->    
   <target name="ejb-classes" depends="prepare">    
       <javac srcdir="${src.ejb.dir}"    
             destdir="${ejb-classes.dest}"    
             includes="com/\*\*"    
             classpathref="base.path" />  
   </target>    
讲解：此小段用来完成编译ejb和其它help classes。根据需要可以加入一个非常有用的元素：  
             encoding="${javac.encoding}"  
  
   <target name="ejb-meta-inf" depends="prepare">    
       <copy file="${etc.dir}/ejb-jar.xml"    
             tofile="${lib.dir}/ejb/META-INF/ejb-jar.xml" />    
       <copy file="${etc.dir}/jaws.xml"    
             tofile="${lib.dir}/ejb/META-INF/jaws.xml" />    
       <copy file="${etc.dir}/jboss.xml"    
             tofile="${lib.dir}/ejb/META-INF/jboss.xml" />    
       <copy file="${etc.dir}/jbosscmp-jdbc.xml"    
             tofile="${lib.dir}/ejb/META-INF/jbosscmp-jdbc.xml" />    
   </target>  
讲解：此小段用来拷贝EJB部署文件，在JAS中EJB部署文件有jaws.xml、jboss.xml、jbosscmp-jdbc.xml。  
  
   <target name="ejb-jar" depends="ejb-classes,ejb-meta-inf">    
       <jar jarfile="${build.dir}/${name}-ejb.jar"    
           basedir="${lib.dir}/ejb" />    
   </target>    
讲解：此小段用来把class和部署文件压缩成包文件，这个包文件就是一个EJB组件。  
  
   <target name="ejbclientjar" depends="ejb-jar,web-classes">  
       <copy   
file="${ProjectName.referencesComponentName.home}/build/  
${ProjectName.referencesComponentName.name}-ejb-client.jar"    
           tofile="${build.dir}/${ProjectName.referencesComponentName.name}  
           -ejb-client.jar" />    
       <jar jarfile="${build.dir}/${name}-ejb-client.jar"    
           basedir="${lib.dir}/ejb"  
           excludes="com/ProjectName/componentName/ejb/ComponentNameEJB.class" />    
   </target>  
讲解：此小段用来把class和部署文件压缩成包文件，以支持客户端运行。  
  
**编译WEB部分**  
<!-- ================================================================== -->    
   <!-- Compilation of the web part of the application                     -->    
   <!-- ================================================================== -->    
   <target name="web-classes" depends="prepare,ejb-jar">    
       <javac srcdir="${src.servlet.dir}"    
             destdir="${lib.dir}/web/WEB-INF/classes"    
             includes="com/\*\*"    
             classpath="${build.classpath}" />    
   </target>    
讲解：此小段用来完成编译servlet。  
  
   <target name="web-web-inf" depends="prepare">  
       <copy file="${etc.dir}/jboss-web.xml"    
             tofile="${lib.dir}/web/WEB-INF/jboss-web.xml" />    
       <copy file="${etc.dir}/web.xml"    
             tofile="${lib.dir}/web/WEB-INF/web.xml" />    
   </target>  
讲解：此小段用来拷贝WEB部署文件，在JAS中WEB部署文件有jboss-web.xml。  
  
   <target name="war" depends="web-classes,web-web-inf">    
       <copy todir="${lib.dir}/web" >  
       <fileset dir="${src.web.dir}"/>  
       </copy>  
       <copy  
file="${build.dir}/${ProjectName.referencesComponentName.name}-ejb-client.jar"    
tofile="${lib.dir}/web/WEB-INF/lib/${ProjectName.referencesComponentName.name}  
-ejb-client.jar" />      
     <jar jarfile="${build.dir}/${name}-web.war"    
           basedir="${lib.dir}/web" />    
   </target>  
讲解：此小段用来把所有的JSP程序、Html、Css、图片和部署文件等压缩成WAR文件。  
  
**编译J2EE Application**  
<!-- ================================================================== -->    
   <!-- Compilation of the complete J2EE application (both web and EJB)     -->    
   <!-- ================================================================== -->    
   <target name="j2ee-meta-inf" depends="prepare">    
       <copy file="${etc.dir}/application.xml"    
             tofile="${lib.dir}/j2ee/META-INF/application.xml" />    
   </target>    
讲解：此小段用来拷贝Application的部署文件。  
  
   <target name="ear" depends="ejb-jar,war,j2ee-meta-inf">    
       <copy file="${build.dir}/${name}-ejb.jar"    
             tofile="${lib.dir}/j2ee/${name}-ejb.jar" />    
       <copy file="${build.dir}/${name}-web.war"    
             tofile="${lib.dir}/j2ee/${name}-web.war" />    
       <jar jarfile="${build.dir}/${name}.ear"    
           basedir="${lib.dir}/j2ee" />    
   </target>    
讲解：此小段用来把EJB组件、支持客户端运行的包和部署文件压缩成EAR文件，它就是一个J2EE Application。这里要说明，在进行build时，根据需要可以不生成EAR文件。  
  
**部署Application**  
<!-- ================================================================ -->  
   <!-- Deploy EAR file                                                   -->  
   <!-- ================================================================ -->  
   <target name="deploy-server" depends="ear,war">    
       <copy todir="${jboss.deploy}">    
         <fileset dir="${build.dir}" includes="\*.ear">    
         </fileset>    
       </copy>    
   </target>    
讲解：此小段用来部署Application，在JAS3.0中${jboss.deploy}是JAS的热部署目录。  
  
**创建组件的API**  
<!-- =================================================================== -->  
   <!-- Create class and package usage pages                                 -->  
   <!-- =================================================================== -->  
   <target name="docs" depends="init">  
     <javadoc locale="${javadoc.locale}" packagenames="${package.names}.${name}.\*"    
             destdir="${javadoc.dir}"    
             classpath="${build.classpath}"  
             encoding="${javadoc.encoding}"  
             author="${javadoc.author}"  
version="${javadoc.version}"   
use="${javadoc.usage}"    
             windowtitle="${project.name} ${name} Component API"  
             doctitle="${project.name} ${name} Component"    
             bottom="Copyright ${sign.copyright} ${project.date}   
${company.signature}. All Rights Reserved.">    
<classpath >  
         <pathelement path="${lib.dir}/ejb"/>    
       </classpath>  
       <sourcepath>  
         <pathelement path="${src.main.dir}/ejb"/>  
       </sourcepath>  
     </javadoc>  
   </target>  
讲解：此小段用来创建组件的API。这里强烈要求类设计人员和编码人员按照Javadoc定义的标签对源代码进行注释。  
  
  
  
**确定build的目标**  
<target name="startbanner">  
       <echo>+---------------------------------------+</echo>  
       <echo>+     Building ${name} Component         +</echo>  
       <echo>+---------------------------------------+</echo>  
   </target>  
   <target name="endbanner" depends="main" >  
       <echo>+---------------------------------------+</echo>  
       <echo>+     Finished ${name} Component         +</echo>  
       <echo>+---------------------------------------+</echo>  
   </target>  
   <target name="main" depends="startbanner, ejb-jar, ejbclientjar" />  
   <target name="main2" depends="startbanner, ejb-jar,  
ejbclientjar,war" />  
   <target name="main3" depends="startbanner, ejb-jar,   
ejbclientjar,war,ear,deploy-server" />  
   <target name="core" depends="ejb-war" />  
   <target name="ejb-war" depends="main2,endbanner" />  
   <target name="deploy" depends="main3,endbanner" />  
   <target name="all" depends="core, docs" />  
讲解：此小段用来确定build的目标。缺省目录为core，所以在build不加参数时，系统将只生成jar文件和war文件。如果 build时加上参数，系统将根据需要来生成文件，例如：在命令行输入ant deploy，系统将生成jar文件、war文件、ear文件，同时将ear文件进行部署。