

## Introdução ao Apache Ant

## Tópicos abordados

- *Este módulo tem como finalidade introduzir a principal ferramenta utilizada no ambiente de desenvolvimento usado neste curso*
- *Ant é uma ferramenta de código aberto e é o padrão de fato do mercado para gerenciamento de projetos independentes de IDEs proprietárias*
- *Por que Ant?*
  - *Os melhores projetos open-source usam Ant*
  - *Várias aplicações J2EE dependem do Ant*
  - *Ant pode ser integrada e usada como principal ferramenta de construção nos principais ambientes de desenvolvimento comerciais*

# O que é Ant?

- Uma ferramenta para **construção** de aplicações
  - Implementada em Java
  - Baseada em roteiros XML
  - Extensível (via scripts ou classes)
  - 'padrão' do mercado
  - Open Source (Grupo Apache, Projeto Jakarta)
- Semelhante a **make**, porém
  - Mais simples e estruturada (XML)
  - Mais adequada a tarefas comuns em projetos Java
  - Independente de plataforma
- Onde encontrar: <http://ant.apache.org>

## Para que serve?

- Para montar praticamente **qualquer** aplicação Java que consista de mais que meia dúzia de classes;

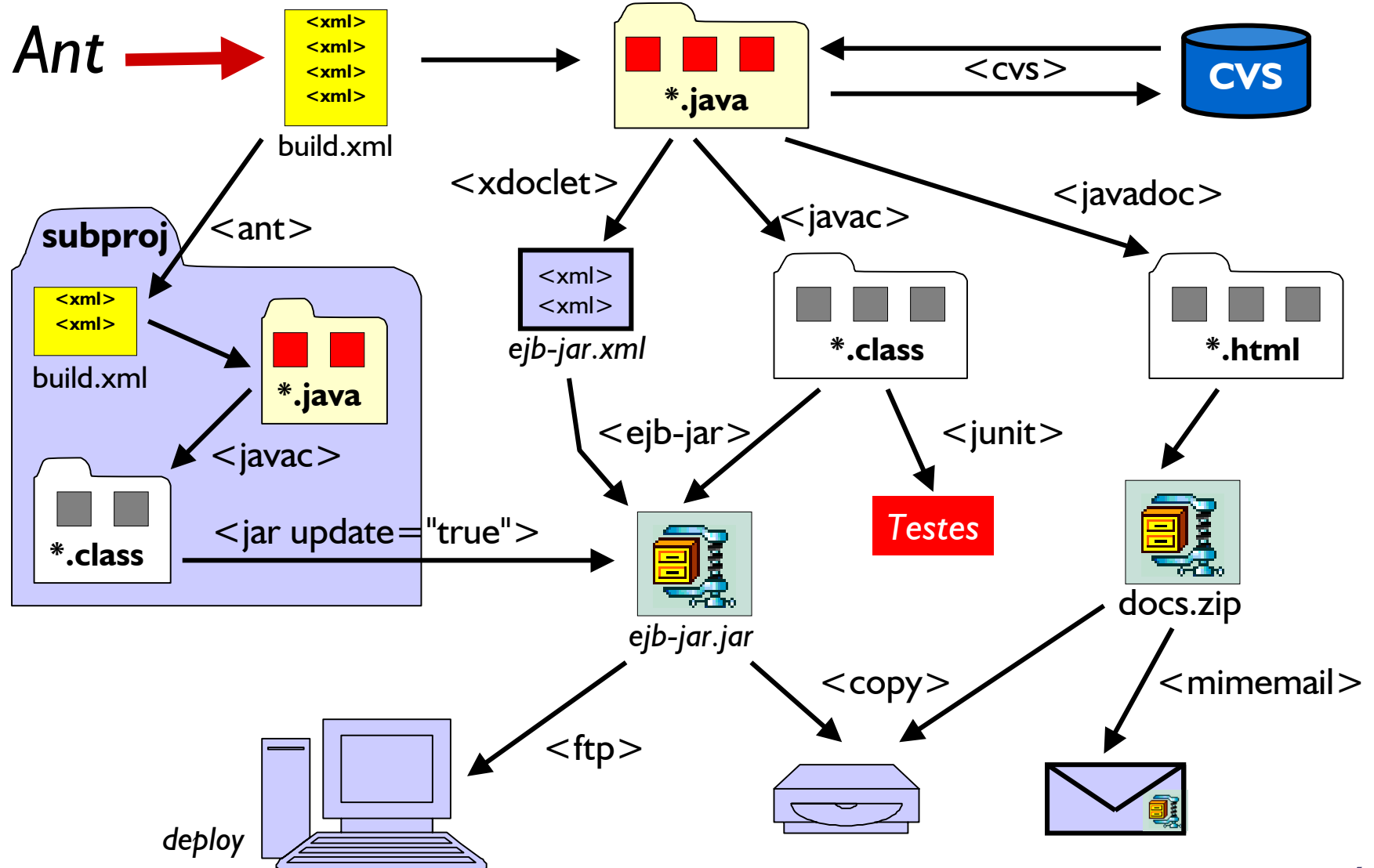
### Aplicações

- Distribuídas em **pacotes**
  - Que requerem a definição de **classpath** locais, e precisam vincular código a bibliotecas (JARs)
  - Cujas criação/instalação depende de mais que uma simples chamada ao javac. Ex: RMI, CORBA, EJB, servlets, JSP,...
- Para **automatizar** processos frequentes
    - Javadoc, XSLT, implantação de serviços Web e J2EE (deployment), CVS, criação de JARs, testes, FTP, email

## Como funciona?

- *Ant executa roteiros escritos em XML: 'buildfiles'*
- *Cada projeto do Ant possui um buildfile*
  - *Subprojetos podem ter, opcionalmente, buildfiles adicionais chamados durante a execução do primeiro*
- *Cada projeto possui uma coleção de alvos*
- *Cada alvo consiste de uma seqüência de tarefas*
- *Exemplos de execução*
  - ▶ `ant`
    - *Procura build.xml no diretório atual e roda alvo default*
  - ▶ `ant -buildfile outro.xml`
    - *Executa alvo default de arquivo outro.xml*
  - ▶ `ant compilar`
    - *Roda alvo 'compilar' e possíveis dependências em build.xml*

# Como funciona (2)



- O buildfile é um arquivo XML: **build.xml** (default)
- Principais elementos
  - `<project default="alvo_default">`
    - Elemento raiz (obrigatório): define o projeto.
  - `<target name="nome_do_alvo">`
    - Coleção de tarefas a serem executadas em seqüência
    - Pode-se estabelecer dependências entre alvos
    - Deve haver **por pelo menos um** `<target>`
  - `<property name="nome" value="valor">`
    - Pares nome/valor usados em atributos dos elementos do build.xml da forma `${nome}`
    - Propriedades também podem ser definidas em linha de comando (`-Dnome=valor`) ou lidas de arquivos externos (atributo `file`)
- **Tarefas** (mais de 130) - usadas dentro dos alvos.  
`<javac>`, `<jar>`, `<java>`, `<copy>`, `<mkdir>`, ...

# Buildfile (2)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-- Compila diversos arquivos .java --> Propriedades

<project default="compile" basedir=".">
  <property name="src.dir" value="src" />
  <property name="build.dir" value="classes" />
  <target name="init">
    <mkdir dir="${build.dir}" />
  </target>
  <target name="clean">
    <delete dir="${build.dir}" />
  </target>
  <target name="compile" depends="init"
    description="Compila os arquivos-fonte">
    <javac srcdir="${src.dir}" destdir="${build.dir}">
      <classpath>
        <pathelement location="${build.dir}" />
      </classpath>
    </javac>
  </target>
</project>
```

**Tarefas**

**Alvos**

**Elementos embutidos nas tarefas**



# Exemplo

## ■ Executando buildfile da página anterior

```
C:\usr\palestra\antdemo> ant
Buildfile: build.xml

init:
  [mkdir] Created dir:
  C:\usr\palestra\antdemo\classes

compile:
  [javac] Compiling 2 source files to
  C:\usr\palestra\antdemo\classes

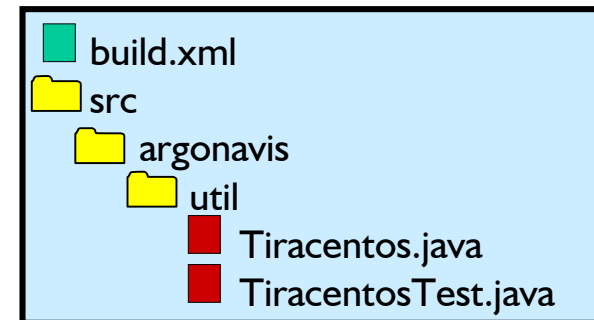
BUILD SUCCESSFUL
Total time: 4 seconds
C:\usr\palestra\antdemo> ant clean
Buildfile: build.xml

clean:
  [delete] Deleting dir:
  C:\usr\palestra\antdemo\classes

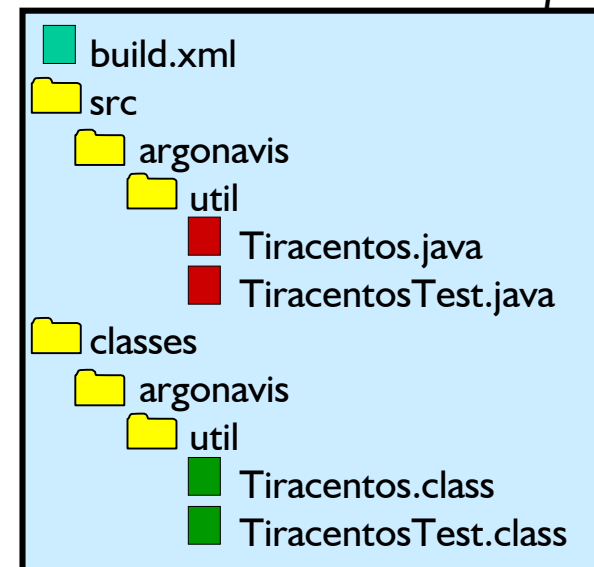
BUILD SUCCESSFUL
Total time: 2 seconds
C:\usr\palestra\antdemo>
```

ANTES de 'ant'

DEPOIS de 'ant clean'



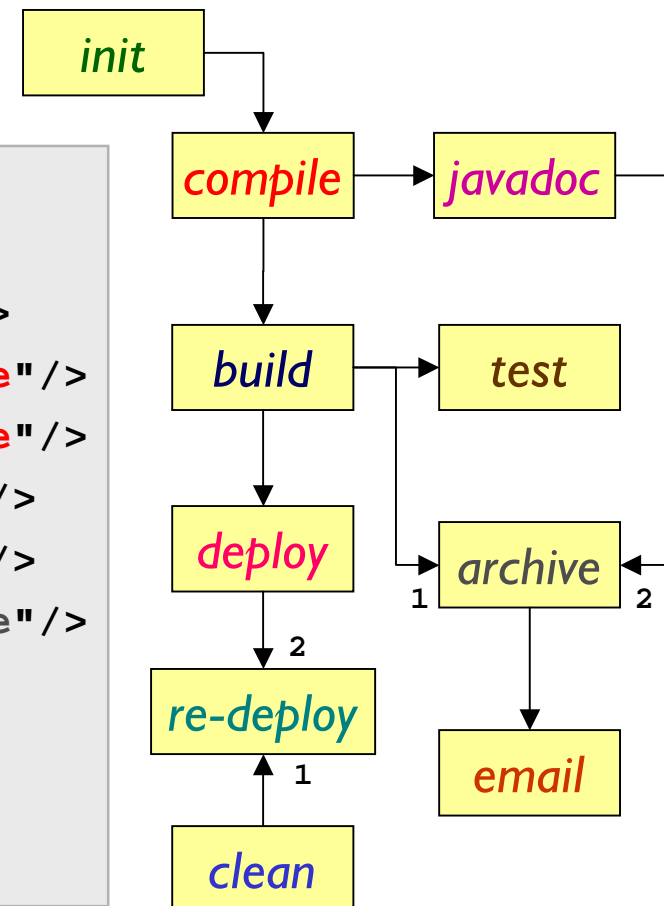
DEPOIS de 'ant' ou 'ant compile'



# Dependências

- Fazem com que a chamada de um alvo cause a chamada de outros alvos, em determinada ordem
  - Promovem reuso de código

```
<target name="init" />
<target name="clean" />
<target name="compile" depends="init"/>
<target name="javadoc" depends="compile"/>
<target name="build" depends="compile"/>
<target name="test" depends="build"/>
<target name="deploy" depends="build"/>
<target name="email" depends="archive"/>
<target name="archive"
  depends="build, javadoc"/>
<target name="re-deploy"
  depends="clean, deploy"/>
```



# Tarefas condicionadas

- *Algumas tarefas só são executadas dentro de determinadas condições*
  - *<mkdir> só cria o diretório se este não existir*
  - *<delete> só apaga o que existe (não faz nada se arquivo ou diretório não existir)*
  - *<javac> compila apenas os arquivos \*.java que foram modificados desde a última compilação*
- *Comportamento condicional do <javac> depende da estrutura de pacotes*
  - *É preciso que a estrutura de diretórios dos fontes (diretório src/) reflita a estrutura de pacotes*
  - *Ex: se Conta.java declara pertencer a pacote banco, deve estar em diretório banco dentro de src/*

# O que se pode fazer com Ant?

- **Compilar.**  
<javac>, <csc>
- **Gerar documentação**  
<javadoc>, <junitreport>, <style>, <stylebook>
- **Gerar código (XDoclet)**  
<ejbdoclet>, <webdoclet>
- **Executar programas**  
<java>, <apply>, <exec>  
<ant>, <sql>
- **Empacotar e comprimir**  
<jar>, <zip>, <tar>, <war>, <ear>, <cab>
- **Expandir, copiar, instalar**  
<copy>, <delete>, <mkdir>, <unjar>, <unwar>, <unzip>
- **Acesso remoto**  
<ftp>, <telnet>, <cvs>, <mail>, <mimemail>
- **Montar componentes**  
<ejbc>, <ejb-jar>, <rmic>
- **Testar unidades de código**  
<junit>
- **Executar roteiros e sons**  
<script>, <sound>
- **Criar novas tarefas**  
<taskdef>

# Tarefas úteis (I)

- **<javac>**: Chama o compilador Java

```
<javac srcdir="dirfontes" destdir="dirbuild" >  
  <classpath>  
    <pathelement path="arquivo.jar" />  
    <pathelement path="/arquivos" />  
  </classpath>  
  <classpath idref="extra" />  
</javac>
```

- **<jar>**: Monta um JAR

```
<jar destfile="bin/executavel.jar">  
  <manifest>  
    <attribute name="Main-class"  
              value="exemplo.main.Exec">  
  </manifest>  
  <fileset dir="${build.dir}"/>  
</jar>
```

## Tarefas úteis (2)

- **<mkdir>**: *cria diretórios*

```
<mkdir dir="diretorio" />
```

- **<copy>**: *copia arquivos*

```
<copy todir="dir" file="arquivo" />
```

```
<copy todir="dir">
```

```
  <fileset dir="fonte" includes="*.txt" />
```

```
</copy>
```

- **<delete>**: *apaga arquivos*

```
<delete file="arquivo" />
```

```
<delete dir="diretorio"/>
```

# Tipos de dados (I)

- **<fileset>**: árvore de arquivos e diretórios
  - Conteúdo do conjunto pode ser reduzido utilizando elementos `<include>` e `<exclude>`
  - Usando dentro de tarefas que manipulam com arquivos e diretórios como `<copy>`, `<zip>`, etc.

```
<copy todir="${build.dir}/META-INF">
  <fileset dir="${xml.dir}" includes="ejb-jar.xml"/>
  <fileset dir="${xml.dir}/jboss">
    <include name="*.xml" />
    <exclude name="*-orig.xml" />
  </fileset>
</copy>
```

Árvore a ser copiada para `${build.dir}/META-INF` consiste de

- O arquivo `ejb-jar.xml` localizado em `${xml.dir}`
- Todos os arquivos `.xml` de `${xml.dir}/jboss` com exceção dos arquivos terminados em `-orig.xml`

## Tarefas úteis (3)

- **<javadoc>**: Gera documentação do código-fonte.
  - Exemplo: alvo *generate-docs* abaixo gera documentação excluindo classes que terminam em 'Test.java'

```
<target name="generate-docs">
  <mkdir dir="docs/api"/>
  <copy todir="tmp">
    <fileset dir="${src.dir}">
      <include name="**/*.java" />
      <exclude name="**/**Test.java" />
    </fileset>
  </copy>

  <javadoc destdir="docs/api"
    packagenames="argonavis.*"
    sourcepath="tmp" />

  <delete dir="tmp" />
</target>
```

Copiar de `${src.dir}`

Procurar em todos os subdiretórios

Onde achar as fontes



# Propriedades

- Podem ser definidas com **<property>**

```
<property name="app.nome" value="jmovie" />
```

- Podem ser carregadas de um arquivo

```
<property file="c:/conf/arquivo.properties" />
```

```
app.ver=1.0  
docs.dir=c:\docs\  
codigo=15323
```

arquivo.properties

- Podem ser passadas na linha de comando

```
c:\> ant -Dautor=Wilde
```

- Para recuperar o valor, usa-se **\${nome}**

```
<jar destfile="${app.nome}-${app.ver}.jar" />
```

```
<echo message="O autor é ${autor}" />
```

```
<mkdir dir="build${codigo}" />
```

# Propriedades especiais

- **<tstamp>**: Grava um instante
  - A hora e data podem ser recuperados como propriedades
    - `#{TSTAMP}`      *hhmm*      *1345*
    - `#{DSTAMP}`      *aaaammdd*      *20020525*
    - `#{TODAY}`      *dia mes ano*      *25 May 2002*
  - Novas propriedades podem ser definidas, locale, etc.
  - Uso típico: `<tstamp/>`
- **<property environment="env">**: Propriedade de onde se pode ler variáveis de ambiente do sistema
  - Dependente de plataforma

```
<target name="init">
  <property environment="env" />
  <property name="j2ee.home"
            value="env.J2EE_HOME" />
</target>
```

## Tipos de dados (2)

- **<patternset>**: coleção de padrões de busca

```
<patternset id="project.jars" >  
  <include name="**/*.jar" />  
  <exclude name="**/*-test.jar" />  
</patternset>
```

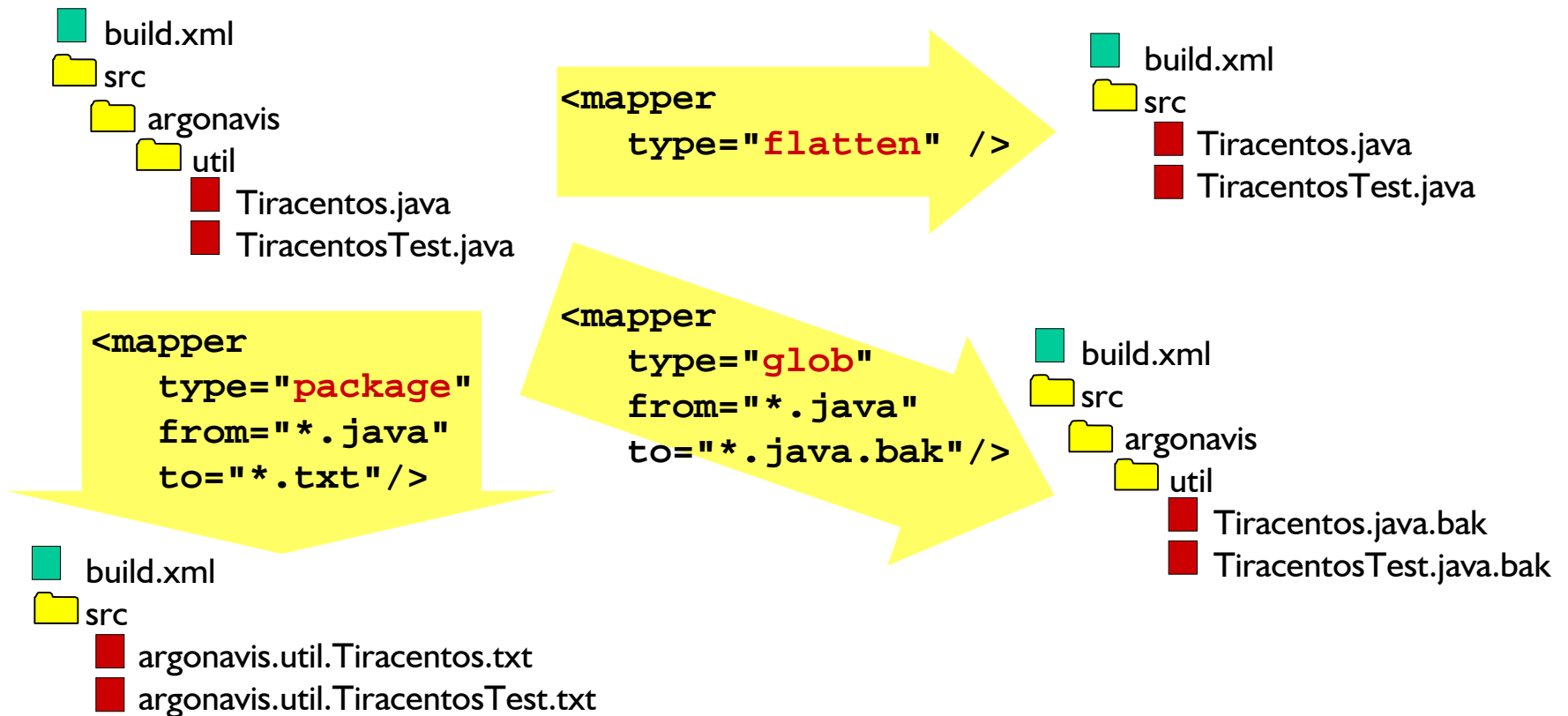
Padrões podem ser reusados e são identificados pelo ID

- **<path>**: coleção de caminhos
  - Associa um ID a grupo de arquivos ou caminhos

```
<path id="server.path">  
  <pathelement path="${j2ee.home}/lib/locale" />  
  <fileset dir="${j2ee.home}/lib">  
    <patternset refid="project.jars" />  
  </fileset>  
</path>  
  
<target name="compile" depends="init">  
  <javac destdir="${build.dir}" srcdir="${src.dir}">  
    <classpath refid="server.path" />  
  </javac>  
</target>
```

## Tipos de dados (3)

- **<mapper>**: altera nomes de arquivos durante cópias ou transformações (use dentro de <copy>, por exemplo)
  - Seis tipos: *identity*, *flatten*, *merge*, *regexp*, *glob*, *package*



## Tipos de dados (4): filtros

- **<filter>** e **<filterset>**: Permite a substituição de padrões em arquivos durante a execução de uma tarefa
  - Caractere default: @
  - Exemplo: a cópia abaixo irá substituir todas as ocorrências de @javahome@ por c:\j2sdk1.4 nos arquivos copiados

```
<copy todir="${dest.dir}">
  <fileset dir="${src.dir}" />
  <filterset>
    <filter token="javahome" value="c:\j2sdk1.4" />
  </filterset>
</copy>
```

- Pares **token=valor** podem ser carregados de arquivo:

```
<filterset>
  <filtersfile file="build.properties" />
</filterset>
```

## Tarefas úteis (4): J2EE

```
<ear destfile="app.ear" appxml="application.xml">
  <fileset dir="${build}" includes="*.jar,*.war"/>
</ear>
```

```
<ejbjar srcdir="${build}" descriptor="${xml.dir}" ... >
  <jboss destdir="${deployjars.dir}" />
</ejbjar>
```

*Há suporte aos principais servidores de aplicação*

```
<war destfile="bookstore.war" webxml="meta/metainf.xml">
  <fileset dir="${build}/${bookstore2}" >
    <include name="*.jsp" />
    <exclude name="*.txt" />
  </fileset>
  <classes dir="${build}" >
    <include name="database/*.class" />
  </classes>
  <lib dir="bibliotecas" />
  <webinf dir="etc" />
</war>
```

*WEB-INF/web.xml*

*Fileset para raiz do WAR*

*Fileset para  
WEB-INF/classes*

*Fileset para  
WEB-INF/lib*

*Fileset para WEB-INF/*

## Tarefas úteis (5): extensão



**<ejbdoclet>** e **<webdoclet>\***: Geram código

- Requer JAR de [xdoclet.sourceforge.net](http://xdoclet.sourceforge.net)
- Ideal para **geração automática de arquivos de configuração** (web.xml, ejb-jar.xml, application.xml, taglibs, struts-config, etc.) e **código-fonte** (beans, value-objects)

```
<ejbdoclet sourcepath="src" destdir="${build.dir}"
           classpathref="xdoclet.path" ejbspec="2.0">
  <fileset dir="src">
    <include name="**/*Bean.java" />
  </fileset>
  <remoteinterface/>
  <homeinterface/>
  <utilobject/>
  <entitypk/>
  <entitycmp/>
  <deploymentdescriptor destdir="${dd.dir}"/>
  <jboss datasource="java:/OracleDS" />
</ejbdoclet>
```

Detalhes da configuração do componente estão nos comentários de JavaDocs do código-fonte dos arquivos envolvidos e arquivos de template

\* Nomes convencionais criados a partir de tarefa externa

## Tarefas úteis (6): execução

- **<java>**: executa o interpretador Java

```
<target name="runrmiclient">
  <java classname="hello.rmi.HelloClient" fork="true">
    <jvmarg value="-Djava.security.policy=rmi.policy"/>
    <arg name="host" value="{remote.host}" />
    <classpath refid="app.path" />
  </java>
</target>
```

- **<exec>**: executa um comando do sistema

```
<target name="orbd">
  <exec executable="{java.home}\bin\orbd">
    <arg line="-ORBInitialHost {nameserver.host}" />
  </exec>
</target>
```

- **<apply>**: semelhante a **<exec>** mas usado em executáveis que operam sobre outros arquivos



## Tarefas úteis (7): rede

- **<ftp>**: Realiza a comunicação com um servidor FTP remoto para upload ou download de arquivos
  - Tarefa opcional que requer **NetComponents.jar** (<http://www.savarese.org>)

```
<target name="remote.jboss.deploy" depends="dist">
  <ftp server="${ftp.host}" port="${ftp.port}"
    remotedir="/jboss/server/default/deploy"
    userid="admin" password="jboss"
    depends="yes" binary="yes">
    <fileset dir="${basedir}">
      <include name="*.war"/>
      <include name="*.ear"/>
      <include name="*.jar"/>
    </fileset>
  </ftp>
</target>
```

## Tarefas úteis (8): XSLT

- **<style>**: Transforma documentos XML em outros formatos usando folha de estilos XSLT (nativa)
  - Usa TrAX (default), Xalan ou outro transformador XSL

```
<style basedir="xmldocs"
       destdir="htmldocs"
       style="xmltohtml.xsl" />
```

- Elemento `<param>` passa valores para elementos `<xsl:param>` da folha de estilos

```
<style in="cartao.xml"
       out="cartao.html"
       style="cartao2html.xsl">
  <param name="docsdir"
        expression="/cartoes"/>
</style>
```

build.xml

cartao2html.xsl

```
(...)  
<xsl:param name="docsdir"/>  
(...)  
<xsl:valueof select="$docsdir"/>  
(...)
```

## Tarefas úteis (9): JDBC

- **<sql>**: Comunica-se com banco de dados através de um driver JDBC

```
<property name="jdbc.url"
  value="jdbc:cloudscape:rmi://server:1099/Cloud" />
<target name="populate.table">
  <sql driver="COM.cloudscape.core.RmiJdbcDriver"
    url="{jdbc.url}"
    userid="helder"
    password="helder"
    onerror="continue">

    <transaction src="droptable.sql" />
    <transaction src="create.sql" />
    <transaction src="populate.sql" />
    <classpath refid="jdbc.driver.path" />

  </sql>
</target>
```

# Tarefas úteis (10): chamadas

- **<ant>**: chama alvo de subprojeto (buildfile

```
<target name="run-sub">
  <ant dir="subproj" />
</target>
```

Chama *alvo default* de *build.xml*  
localizado no subdiretório *subproj/*

```
<target name="run-sub">
  <ant dir="subproj" >
    <property name="versao"
      value="1.0" />
  </ant>
</target>
```

Define propriedade que  
será lida no outro build.xml

```
<target name="fazer-isto">
  <antcall target="fazer">
    <param name="oque"
      value="isto" />
  </antcall>
</target>
```

```
<target name="fazer-aquilo">
  <antcall target="fazer">
    <param name="oque"
      value="aquilo" />
  </antcall>
</target>
```

```
<target name="fazer" if="oque">
  <tarefa atributo="${oque}" />
</target>
```

← **Template!**

## Efeitos sonoros

- **<sound>**: define um par de arquivos de som para soar no sucesso ou falha de um projeto
  - Tarefa opcional que requer Java Media Framework
- Exemplo:
  - No exemplo abaixo, o som `festa.wav` será tocado quando o build terminar sem erros fatais. `vaia.wav` tocará se houver algum erro que interrompa o processo:

```
<target name="init">
  <sound>
    <success source="C:/Media/festa.wav" />
    <fail source="C:/Media/vaia.wav" />
  </sound>
</target>
```

# Extensão usando XML

- Como o *buildfile* é um arquivo XML, pode-se incluir trechos de XML externos através do uso de *entidades externas*

*sound.xml*

```
<property file="sound.properties" />
<sound>
  <success source="{success.sound}"/>
  <fail source="{fail.sound}"/>
</sound>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE project [
  <!ENTITY sound SYSTEM "sound.xml">
]>
<project default="dtd">
  <description>Gera um DTD para o Ant</description>
  <target name="init">
    &sound;
  </target>
  <target name="dtd" depends="init">
    <antstructure output="ant.dtd" />
  </target>
</project>
```

# Como gerenciar projetos com o Ant

- *Crie um diretório para armazenar seu projeto. Guarde na sua raiz o seu **build.xml***
  - *Use um arquivo **build.properties** para definir propriedades exclusivas do seu projeto (assim você consegue reutilizar o mesmo build.xml em outros projetos). Importe-o com*  

```
<property file="build.properties" />
```
- *Dentro desse diretório, crie alguns subdiretórios*
  - **src/** *Para armazenar o código-fonte*
  - **lib/** *Opcional. Para guardar os JARs de APIs usadas*
  - **doc/** *Opcional. Para guardar a documentação gerada*
  - **etc/** *Opcional. Para arquivos de configuração se houver*
  - **web/** *Em projetos Web, para raiz de documentos do site*
- *O seu Ant script deve ainda criar durante a execução*
  - **build/** *Ou **classes/**. Onde estará o código compilado*
  - **dist/** *Ou **jars/** ou **release/**. Onde estarão JARs criados*

## Alvos básicos do build.xml

- Você também deve **padronizar os nomes dos alvos** dos seus build.xml. Alguns alvos típicos são
  - **init** Para criar diretórios, inicializar o ambiente, etc.
  - **clean** Para fazer a faxina, remover diretórios gerados, etc.
  - **compile** Para compilar
  - **build** Para construir a aplicação, integrar, criar JARs
  - **run** Para executar um cliente da aplicação
  - **test** Para executar os testes da aplicação
  - **deploy** Para implantar componentes Web e EJB
- Você pode usar outros nomes, mas mantenha um padrão
- Também pode criar uma nomenclatura que destaque alvos principais, usando maiúsculas. Ex:
  - **CLEAN**, chamando **clean-isto**, **clean-aquilo**, **undeploy**, etc.
  - **BUILD**, que chama **build-depend**, **build-client**, **build-server**



# Exemplo de projeto

```
<project default="compile" name="MiniEd">
  <property file="build.properties"/>
  <target name="init">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${dist.dir}"/>
  </target>
  <target name="clean"> ... </target>
  <target name="compile">
    depends="init"> ... </target>
  <target name="build">
    depends="compile">...</target>
  <target name="javadoc">
    depends="build"> ... </target>
  <target name="run">
    depends="build"> ... </target>
</project>
```

*build.xml*

```
# Nome da aplicação
app.name=minied
# Nomes dos diretórios
src.dir=src
docs.dir=docs
build.dir=classes
dist.dir=jars
# Nome da classe executável
app.main.class=com.javamagazine.minied.MinEditor
root.package=com
```



*build.properties*

# Buildfile: aplicação gráfica executável

```
<project default="compile" name="MiniEd">

  <property file="build.properties"/>

  <target name="compile" depends="init">
    <javac destdir="classes" srcdir="src">
      <classpath>
        <pathelement location="classes"/>
      </classpath>
    </javac>
  </target>

  <target name="build" depends="compile">
    <jar destfile="release/${app.name}.jar">
      <manifest>
        <attribute name="Main-class" value="${app.main.class}" />
      </manifest>
      <fileset dir="classes"/>
    </jar>
  </target>

  <target name="run" depends="build">
    <java jar="release/${app.name}.jar" fork="true" />
  </target>

</project>
```


Definindo o JAR com  
atributo Main-class para  
torná-lo executável



```
# Nome da aplicação - este nome será usado para criar o JAR
app.name=minied
# Nome da classe executável
app.main.class=com.javamagazine.minied.Minieditor
```

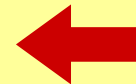
# Buildfile: aplicação RMI-IIOP

```
<project name="Aplicação RMI" default="compile">
  <target name="compile" depends="init"> <!-- Vários <target> omitidos -->
    <javac destdir="classes" srcdir="src" >
      <classpath refid="app.path" />
    </javac>
  </target>
  <target name="buildrmi" depends="compile">
    <rmic idl="true" iiop="true" base="classes">
      <include name="**/rmiop/**Impl.class" />
      <include name="**/portable/**Impl.class" />
    </rmic>
  </target>
  <target name="runserver" depends="buildrmi">
    <java classname="hello.rmiop.HelloServer" fork="true">
      <jvmarg value="-Djava.rmi.server.codebase=${codebase}"/>
      <jvmarg value="-Djava.security.policy=${lib.dir}/rmi.policy"/>
      <jvmarg value="-Djava.naming.factory.initial=..."/>
      <jvmarg value="-Djava.naming.provider.url=iiop://${host}:1900"/>
      <classpath refid="app.path" />
    </java>
  </target>
  <target name="orbd">
    <exec executable="${java.home}\bin\orbd">
      <arg line="-ORBInitialPort 1900 -ORBInitialHost ${host}"/>
    </exec>
  </target>
</project>
```



# Buildfile: aplicação Web

```
<project default="deploy" name="Aplicação Web">                                     build.xml
  <property file="build.properties" /> <!-- init e clean omitidos -->
  <target name="compile" depends="init">
    <javac srcdir="src" destdir="classes">
      <classpath path="${servlet.jar}" />
    </javac>
  </target>
  <target name="war" depends="compile">
    <war warfile="release/${context}.war" webxml="etc/web.xml">
      <fileset dir="web" />
      <classes dir="classes" />
    </war>
  </target>
  <target name="deploy" depends="war">
    <copy todir="${deploy.dir}">
      <fileset dir="release">
        <include name="*.war" />
      </fileset>
    </copy>
  </target>
</project>
```



```
# Localizacao do Servidor                                     build.properties
tomcat.home=/tomcat-4.0

# Altere para informar dir de instalacao
deploy.dir=${tomcat.home}/webapps

# Coloque aqui nome do contexto
context=forum

# JAR com Servlet API
servlet.jar=${tomcat.home}/common/lib/servlet.jar
```

# Buildfile: aplicação EJB

```
<project name="Aplicação EJB" default="deploy">                                     build.xml
  <property file="build.properties" />
  <!-- elementos <path> e <target> init, compile, clean omitidos -->
  <target name="build" depends="compile">
    <copy todir="classes/META-INF">
      <fileset dir="etc" includes="ejb-jar.xml"/>
    </copy>
    <jar jarfile="release/${app.name}.jar">
      <fileset dir="classes" />
    </jar>
  </target>
  <target name="deploy" depends="build">
    <copy todir="${deploy.dir}" file="release/${app.name}.jar" />
  </target>
  <target name="undeploy" depends="build">
    <delete file="${deploy.dir}/${app.name}.jar" />
  </target>
</project>
```



```
# Localizacao do Servidor                                     build.properties
jboss.home=/jboss-3.0.0

# Altere para informar dir de instalacao
deploy.dir=${jboss.home}/server/default/deploy

# Coloque aqui nome da aplicação
app.name=forumejb
```

# Buildfile: transformação XSL

```
<project name="foptask-example" default="pdf">
```

```
  <target name="setup" depends="check">
```

```
    <taskdef name="fop" classname="argonavis.pdf.FopTask">
```

```
      <classpath> ... </classpath>
```

```
    </taskdef>
```

```
  </target>
```

Mescla vários XML em um único XML maior e converte em XSL-FO



```
  <target name="many2fo" depends="init">
```

```
    <style in="template.xml" out="all.xml" style="many2one.xsl">
```

```
      <param name="docsdire" expression="dados"/>
```

```
    </style>
```

```
    <style in="all.xml" out="all.fo"
```

```
      extension=".fo" style="many2fo.xsl"/>
```

```
  </target>
```

Converte XSL-FO em PDF

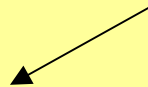


```
  <target name="many2pdf" depends="many2fo">
```

```
    <fop in="all.fo" out="all.pdf" />
```

```
  </target>
```

Converte vários XML em HTML



```
  <target name="html" depends="init">
```

```
    <style basedir="dados" destdir="html"
```

```
      extension=".html" style="toHtml.xsl" />
```

```
  </target>
```

```
</project>
```

# Integração com outras aplicações

- *Ant provoca vários **eventos** que podem ser capturados por outras aplicações*
  - *Útil para implementar integração, enviar notificações por email, gravar logs, etc.*
- **Eventos**
  - *Build iniciou/terminou*
  - *Alvo iniciou/terminou*
  - *Tarefa iniciou/terminou*
  - *Mensagens logadas*
- **Vários listeners e loggers pré-definidos**
  - *Pode-se usar ou estender classe existente.*
  - *Para gravar processo (build) em XML:*
    - > `ant -listener org.apache.tools.ant.XmlLogger`

# Integração com editores e IDEs

- *Produtos que integram com Ant e oferecem interface gráfica e eventos para buildfiles:*
  - **Antidote**: GUI para Ant (do projeto Jakarta)
    - <http://cvs.apache.org/viewcvs/jakarta-ant-antidote/>
  - **JBuilder** (AntRunner plug-in)
    - <http://www.dieter-bogdoll.de/java/AntRunner/>
  - **NetBeans** e **Forté for Java**
    - <http://ant.netbeans.org/>
  - **Eclipse**
    - <http://eclipse.org>
  - **JEdit** (AntFarm plug-in)
    - <http://www.jedit.org>
  - **Jext** (AntWork plug-in)
    - <ftp://jext.sourceforge.net/pub/jext/plugins/AntWork.zip>



# Integração com o JEdit

**Tela do AntFarm mostra alvos do Ant.  
Pode-se clicar sobre o alvo para executá-lo**

**Resultados são mostrados  
no Console do JEdit**

```
151 .
152 /**
153  * Description of the Method.
154  *
155
158
159
161 XMLScanner xs = u.scanner();
162 Validator v = u.valid:
163 xs.takeStart("message)
164 while (xs.atAttribute
165 String an = xs.ta)
166 if (an.equals("ti
```

Ant

Targets:

```
RUN-TESTS message-cleanup deploy CLEAN generate-bean undeploy-all
undeploy-tests deploy-tests clean build-tests test-cleanup undeploy build
[default] init compile JAVADOC DEPLOY
```

# Ant programável


- Há duas formas de estender o Ant com novas funções
  - Implementar roteiros usando *JavaScript*
  - Criar *novas tarefas* reutilizáveis
- A tarefa *<script>* permite embutir JavaScript em um buildfile. Pode-se
  - Realizar operações aritméticas e booleanas
  - Utilizar estruturas como *if/else*, *for*, *foreach* e *while*
  - Manipular com os elementos do buildfile usando DOM
- A tarefa *<taskdef>* permite definir novas tarefas
  - Tarefa deve ser implementada em Java e estender *Task*
  - Método *execute()* contém código de ação da tarefa
  - Cada atributo corresponde a um método *setXXX()*

# Exemplo de script

- Cria 10 diretórios pasta1, pasta2, etc. em pastas/

```
<project name="scriptdemo" default="makedirs" >
  <property name="result.dir" value="pastas" />
  <target name="setup-makedirs">
    <mkdir dir="${result.dir}" />
    <script language="javascript"><![CDATA[
      for (i = 0; i < 10; i++) {
        criadir = scriptdemo.createTask("mkdir");
        // Obter propriedade ${result.dir} deste projeto
        root = scriptdemo.getProperty("result.dir");
        // Definir diretorio a criar
        criadir.setDir(new
          Packages.java.io.File(root+"/pasta"+(i+1)));
        // Executa tarefa mkdir (todo Task tem um metodo execute)
        criadir.execute();
      }
    ]]></script>
  </target>
  <target name="makedirs" depends="setup-makedirs" />
</project>
```

Obtém referência para objeto que implementa tarefa mkdir



# Exemplo de definição de tarefas (I)

```
import org.apache.tools.ant.*;

public class ChangeCaseTask extends Task {

    public static final int UPPERCASE = 1;
    private String message;
    private int strCase = 0;

    public void execute() {
        log( getMessage() );
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void setCase(String strCase) {
        if (strCase.toLowerCase().equals("uppercase"))
            this.strCase = UPPERCASE;
        else if (strCase.toLowerCase().equals("lowercase"))
            this.strCase = -UPPERCASE;
    }

    public String getMessage() {
        switch(strCase) {
            case UPPERCASE: return message.toUpperCase();
            case -UPPERCASE: return message.toLowerCase();
            default: return message;
        }
    }
}
```

```
<target name="define-task" depends="init">
    <taskdef name="changeCase"
        classname="ChangeCaseTask">
        <classpath>
            <pathelement location="tasks.jar" />
        </classpath>
    </taskdef>
</target>

<target name="run-task" depends="define-task">
    <changeCase message="Mensagem simples" />
    <changeCase message="Mensagem em caixa-alta"
        case="uppercase" />
    <changeCase message="Mensagem em caixa-baixa"
        case="lowercase" />
</target>
```

Trecho do build.xml usando tarefa  
`<changeCase>`

Cada atributo é definido em um método `setAtributo(String)`

Método `execute()` chama `log()`, que imprime resultado na saída do Ant

(Exceções foram ignoradas por falta de espaço)

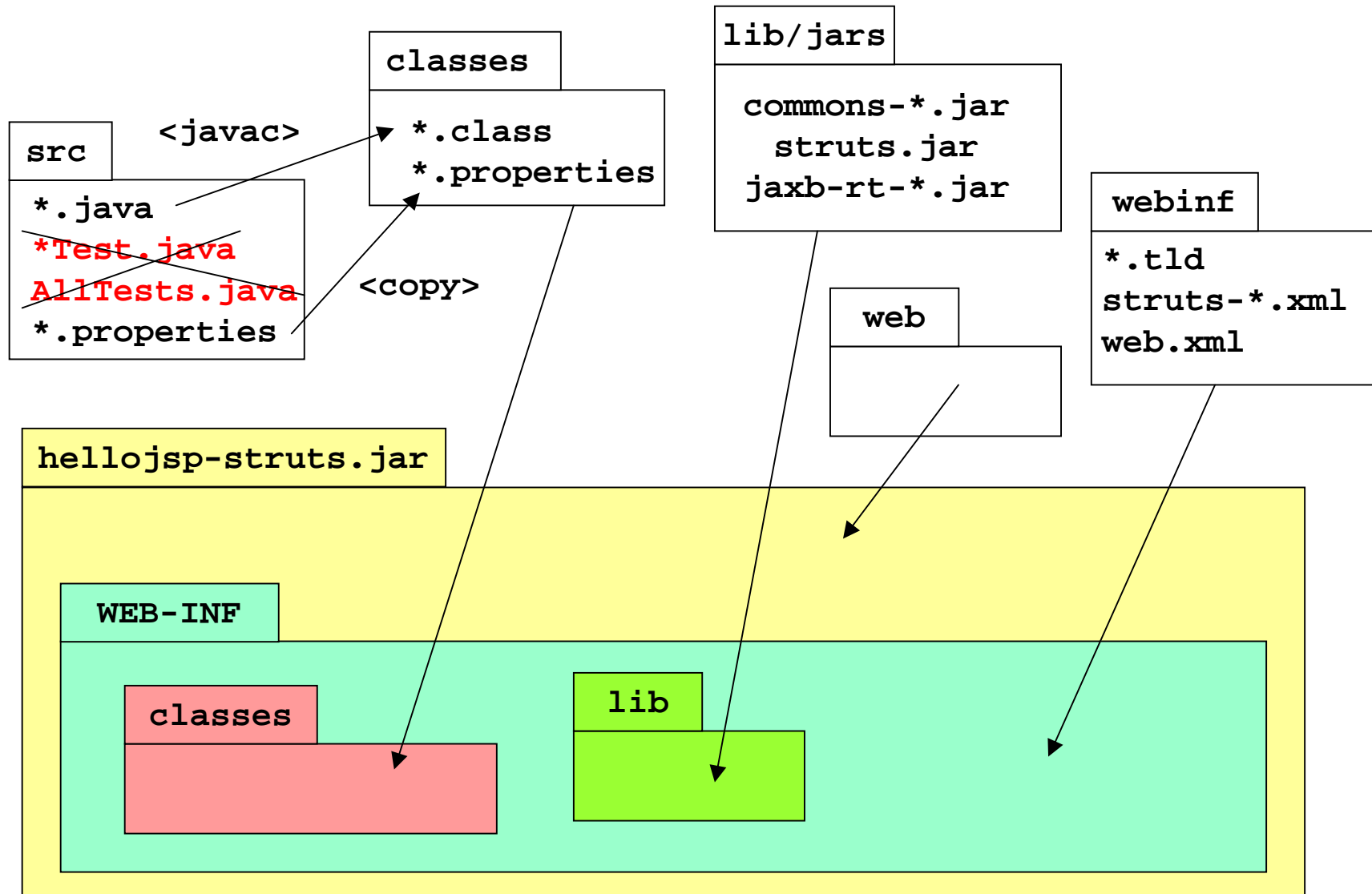
- *Ant é uma ferramenta indispensável em qualquer projeto de desenvolvimento Java*
  - *Permite **automatizar** todo o desenvolvimento*
  - *Facilita a montagem da aplicação por outras pessoas*
  - *Ajuda em diversas tarefas essenciais do desenvolvimento como compilar, rodar, testar, gerar JavaDocs, etc.*
  - *Independente de um IDE comercial (mas pode ser facilmente integrado a um)*
- *Use o Ant em **todos** os seus projetos*
  - *Crie sempre um projeto e um buildfile, por mais simples que seja a sua aplicação*
  - *Escreva buildfiles que possam ser reutilizados*
  - *Desenvolva o hábito de sempre usar o Ant*

- 1. Monte um buildfile simples para a aplicação **MiniEd.jar**. Use o `build.xml` e `build.properties` fornecidos (com comentários). Implemente alvos para
  - *Compilar a aplicação*
  - *Montar a aplicação como um JAR executável*
  - *(opc.) Gerar JavaDocs da aplicação e colocá-los em um ZIP*
  - *(opc.) Executar a aplicação*

*Observações: (a) use `<exclude>` patternsets para não compilar as classes que terminarem em `*Test.java`.*

*(b) Inclua a pasta `icons/` no CLASSPATH (raiz do JAR)*
- 2. (opcional) Monte um **build.xml** para construir um WAR para a aplicação Web fornecida
  - *Veja desenho na próxima página (observe os excludes)*

# Exercicio 2 (diagrama)



- [1] Richard Hightower e Nicholas Lesiecki. *Java Tools for eXtreme Programming*. Wiley, 2002. *Explora Ant e outras ferramentas em ambiente XP.*
- [3] *Apache Ant User's Manual*. *Ótima documentação repleta de exemplos.*
- [3] Steve Lougran. *Ant In Anger - Using Ant in a Production Development System*. (Ant docs) *Ótimo artigo com boas dicas para organizar um projeto mantido com Ant.*
- [4] Martin Fowler, Matthew Foemmel. *Continuous Integration*. <http://www.martinfowler.com/articles/continuousIntegration.html>. *Ótimo artigo sobre integração contínua e o CruiseControl.*
- [5] Erik Hatcher. *Java Development with Ant*. Manning Publications. August 2002. *Explora os recursos básicos e avançados do Ant, sua integração com JUnit e uso com ferramentas de integração contínua como AntHill e CruiseControl.*
- [6] Jesse Tilly e Erik Burke. *Ant: The Definitive Guide*. O'Reilly and Associates. May 2002. *Contém referência completa e ótimo tutorial sobre recursos avançados como controle dos eventos do Ant e criação de novas tarefas.*



# Curso J530: Enterprise JavaBeans

Revisão 2.0 - Junho de 2003

© 2001-2003, Helder da Rocha  
(helder@acm.org)

 [argonavis.com.br](http://argonavis.com.br)