J820

Ambiente de Desenvolvimento Java usando Ant

Helder da Rocha (helder@acm.org)

argonavis.com.br

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

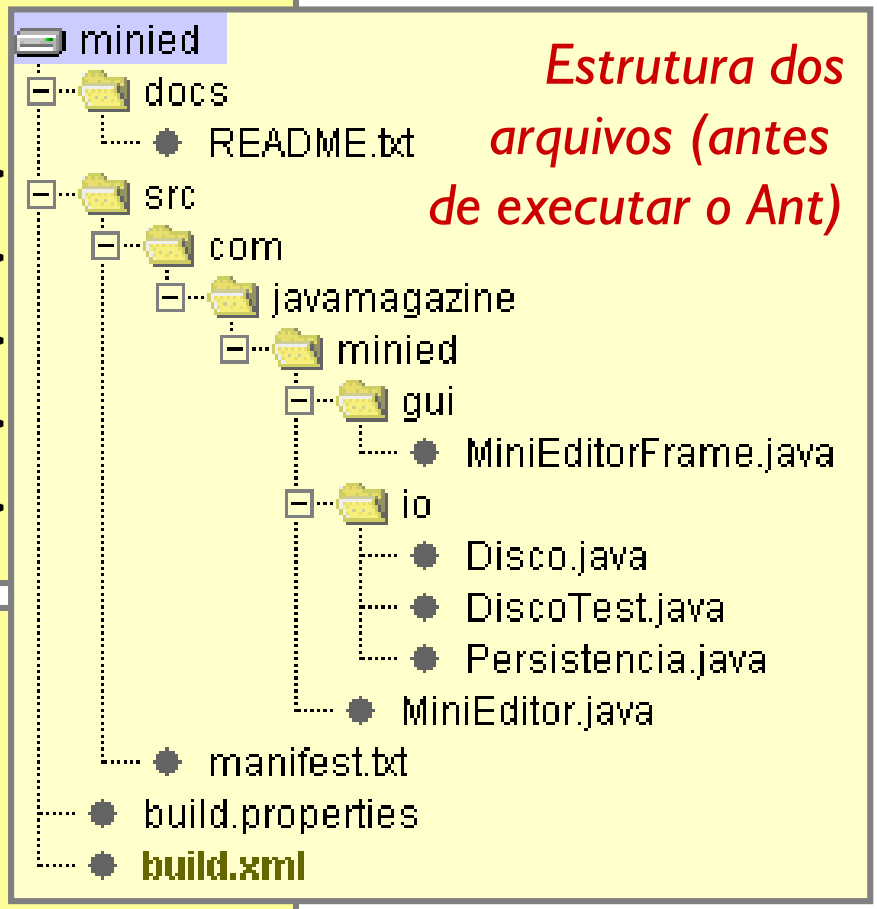
Alvos básicos do build.xml

- Você também deve padronizar os nomes dos alvos dos seus build.xml. Alguns alvos típicos são
  - init — Para criar dietórios, 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



Estrutura dos arquivos (antes de executar o Ant)

```
# 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.MiniEditor
root.package=com
```

build.properties

Buildfile: aplicação gráfica executável

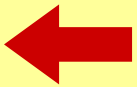
```
<project default="compile" name="MiniEd">

    <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

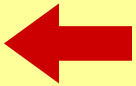
```
<project name="Aplicação RMI" default="compile">
    <target name="compile" depends="init"> <!-- Vários <target> omitidos -->
        <javac destdir="classes" srcdir="src" >
            <classpath refid="app.path" />
        </javac>
    </target>
    <target name="buildrmi" depends="compile">
        <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

build.xml

```
<project default="deploy" name="Aplicação Web">

    <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>
```



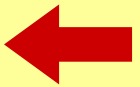
build.properties

```
# Localizacao do Servidor
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

build.xml

```
<project name="Aplicação EJB" default="deploy">

    <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>
```



build.properties

```
# Localizacao do Servidor
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>

    <target name="many2fo" depends="init">
        <style in="template.xml" out="all.xml" style="many2one.xsl">
            <param name="docsdir" expression="dados"/>
        </style>
        <style in="all.xml" out="all.fo"
               extension=".fo" style="many2fo.xsl"/>
    </target>

    <target name="many2pdf" depends="many2fo">
        <fop in="all.fo" out="all.pdf" />
    </target>

    <target name="html" depends="init">
        <style basedir="dados" destdir="html"
               extension=".html" style="toHtml.xsl" />
    </target>

</project>
```

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

Converte XSL-FO em PDF

Converte vários XML em HTML

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 (1)

```
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)

Conclusões

- 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.
  - Independe 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

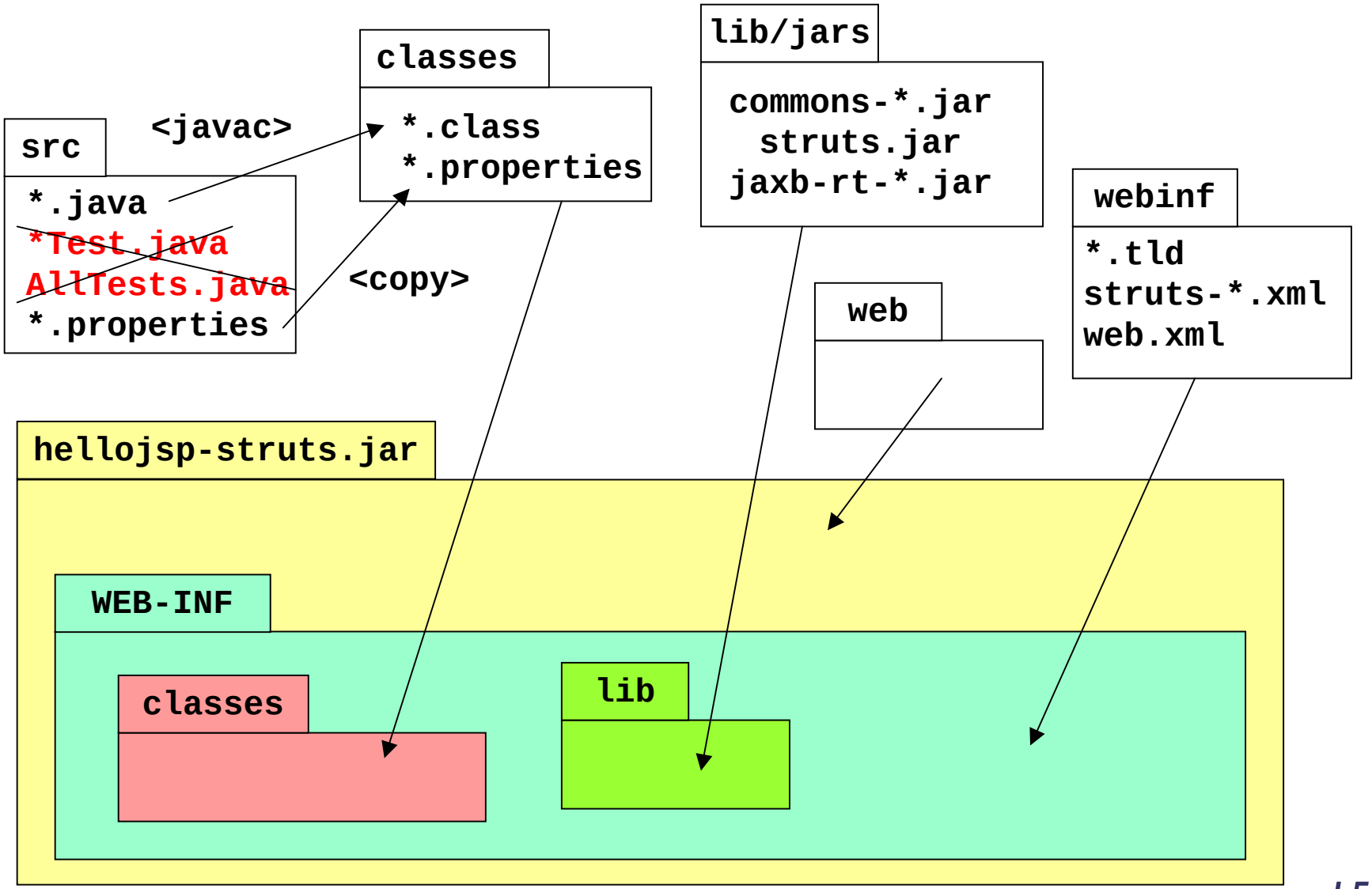
Exercício

- 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
  - Gerar JavaDocs da aplicação e colocá-los em um ZIP
  - 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)



Fontes

[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.

[7] Karl Fogel. *Open Source Development with CVS*. Coriolis Press. http://cvsbook.red-bean.com/.

Curso J820
Produtividade e Qualidade em Java: Ferramentas e Metodologias

Revisão 1.1

(helder@acm.org)

argonavis.com.br