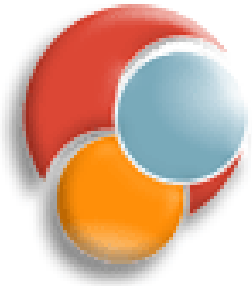


Java y Herramientas de Desarrollo

Sesión 3: Automatización con Ant

Puntos a tratar

- Introducción a la herramienta
- Estructura de un buildfile
- Estructuras de path
- Tareas de Ant
- Integración con Eclipse

Introducción

- El proceso de construcción de una aplicación puede ser complejo:
  - Establecer *classpath*,
  - compilar (`javac`),
  - copiar ficheros necesarios,
  - empaquetar (`jar`),
  - ejecutar (`java`),
  - etc.
- Necesitamos una herramienta que nos permita automatizar este proceso.
- Conservar independencia de la plataforma.

Ant vs Make

- *Make*:
  - Ejecuta comandos del *shell*
    Es dependiente de la plataforma
  - Formato de los `makefile` poco claro (espacios y tabuladores)
- *Ant*:
  - Ejecuta clases Java (tareas)
    Independiente de la plataforma
  - Ficheros de configuración (`build.xml`) en XML
  - Amplio conjunto de tareas disponibles (extensible).

```
<mkdir dir="bin"/>
<javac srcdir="src" destdir="bin"/>
```

Ejecución en el shell

- Configurar variables de entorno

```
JAVA_HOME
ANT_HOME
PATH=${PATH}:${JAVA_HOME}/bin:${ANT_HOME}/bin
```

- Ejecutar *Ant* en el directorio que contenga el fichero `build.xml`

```
ant
```

Proyecto del buildfile

```
<project name="MiProyecto"
         default="run“
         basedir=".">

    <description>
        Ejemplo de fichero build.xml
    </description>

    <!-- Propiedades -->
    ...

    <!-- Declaración de tareas -->
    ...

    <!-- Objetivos -->
    ...

</project>
```

Propiedades

- Nos permiten parametrizar el fichero

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

- Se pueden definir:
  - En el fichero `build.xml`

```
<property name="build.home" value="bin">
```

  - En un fichero de propiedades (p.ej. `build.properties`)

```
build.home=bin
```

  - En `build.xml`

```
<property file="build.properties"/>
```

  - En línea de comando

```
ant -Dbuild.home=bin
```

Declaración de tareas

- Conjunto de tareas extensible.
- Para añadir nuevas tareas debemos declararlas
- Debemos indicar la clase que implementa la tarea y un nombre que la identifique

```
<taskdef name="mitarea"
         classname="es.ua.j2ee.tareas.MiTarea"/>
```

- Podrá utilizarse como cualquier otra tarea

```
<mitarea [atributos]/>
```

Objetivos

- Podemos crear diferentes objetivos

  `compile`, `run`, `clean`, etc

- Para cada objetivo definimos las tareas necesarias para cumplirlo. P.ej:

```
<target name="compile">
  <mkdir dir="${build.home}">
  <javac srcdir="${src.home}" destdir="${build.home}“/>
</target>
```

- Ejecutamos objetivos con

```
   ant [objetivo]
```

Ejemplo

```
<project name="MiProyecto" default="run" basedir=".">
  <description>
    Ejemplo de fichero build.xml
  </description>

  <!-- Propiedades -->
  <property name="src.home" value="src"/>
  <property name="build.home" value="bin"/>

  <!-- Objetivos -->
  <target name="compile">
    <mkdir dir="${build.home}"/>
    <javac srcdir="${src.home}" destdir="${build.home}“/>
  </target>

  <target name="run" depends="compile">
    <java classname="es.ua.j2ee.prueba.Principal“ classpath="${build.home}"/>
  </target>

  <target name="clean">
    <delete dir="${build.home}"/>
  </target>
</project>
```

Definición del classpath

- Para compilar, ejecutar y otras tareas necesitamos establecer el *classpath*
- Definimos el *classpath* como una estructura de *path*

```
<path id="path.compilacion">
  <!-- Elementos del path -->
  ...
</path>
```

- El identificador nos permitirá utilizar este *path* desde las tareas

Elementos individuales del path

- Podemos introducir elementos individuales al *path* (directorios o ficheros)

```
<path id="path.compilacion">
  <pathelement location="classes"/>
  <pathelement location="lib/libreria.jar"/>
</path>
```

- Importar la variable de entorno *classpath*

```
<path id="path.compilacion">
  <pathelement path="${classpath}"/>
</path>
```

Conjuntos de ficheros

- Podemos incluir todo un conjunto de ficheros en el *path*

```
<path id="path.compilacion">
  <fileset dir="lib">
    <include name="**/*.jar"/>
    <include name="**/*.zip"/>
    <exclude name="**/*debug*"/>
  </fileset>
</path>
```

- También podemos incluir directorios

```
<dirset dir="${tomcat.home}">
  <include name="**/classes"/>
</dirset>
```

Compilación (javac)

- Caso básico

```
<javac srcdir="${src.home}"
       destdir="${build.home}">
</javac>
```

- *Classpath*

```
<javac srcdir="${src.home}"
       destdir="${build.home}">
  <classpath refid="path.compilacion"/>
  <extdirs refid="directorios.ext"/>
  <bootclasspath refid="path.nucleo"/>
</javac>
```

Compilación avanzada

- Atributos y directorios de fuentes

```
<javac destdir="${build.home}">
       debug="off"
       optimize="on"
       deprecation="off">
  <src path="${src}"/>
  <src path="${src2}"/>
</javac>
```

- `fileset` implícito

```
<javac srcdir="${src.home}" destdir="${build.home}">
  <include name="es/ua/j2ee/aplic1/**"/>
  <include name="es/ua/j2ee/aplic2/**"/>
  <exclude name="es/ua/j2ee/aplic1/prueba/**"/>
</javac>
```

Documentación (javadoc)

- Genera documentación en formato Javadoc de nuestras clases

```
<javadoc sourcepath="src"
         destdir="${docs.home}"
         packagenames="*">
  <classpath refid="path.compilacion"/>
</javadoc>
```

Empaquetamiento (jar)

- `fileset` implícito

```
<jar destfile="${dist.home}/aplic.jar"
     basedir="${build.home}">
  <include name="build/**/*.class"/>
  <include name="build/**/*.gif"/>
  <exclude name="build/prueba/**"/>
</jar>
```

- Manifest

```
<jar destfile="${dist.home}/aplic.jar"
     basedir="${build.home}">
  <manifest>
    <attribute name="Main-Class"
        value="es.ua.j2ee.prueba.Principal"/>
  </manifest>
</jar>
```

Ejecución (java)

- Paso de argumentos

```
<java classname="es.ua.j2ee.prueba.Principal">
  <arg value="-f"/>
  <arg value="datos.txt"/>
  <classpath>
    <pathelement location="${build.home}"/>
  </classpath>
</java>
```

- Atributos

```
<java classname="es.ua.j2ee.prueba.Principal"
      fork="true"
      failonerror="true"
      maxmemory="128m">
  <classpath refid="path.ejecucion"/>
</java>
```

Ejecución desde JAR ejecutable

- El atributo `fork` debe ser `true`

```
<java jar="${dist.home}/aplic.jar"
      fork="true">
</java>
```

- En este caso no se especifica *classpath*
  - Buscará las clases sólo dentro del JAR

Sistema de ficheros

- Crear directorio

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

- Copiar ficheros (`fileset`)

```
<copy todir="${build.home}">
  <fileset dir="${res.home}"/>
</copy>
```

- Borrar ficheros y directorios (`delete`)

```
<delete dir="${build.home}"/>
<delete file="${dist.home}/aplic.jar"/>
```

- Podemos especificar un `fileset` al borrar

Comandos externos (exec)

- Podemos ejecutar comandos del *shell*

```
<exec executable="genera.exe"
      dir="${tools.home}"
      os="Windows 2000">
  <arg line="-o datos.txt"/>
</exec>
```

- Perdemos portabilidad
  - Definir la etiqueta adecuada para cada SO

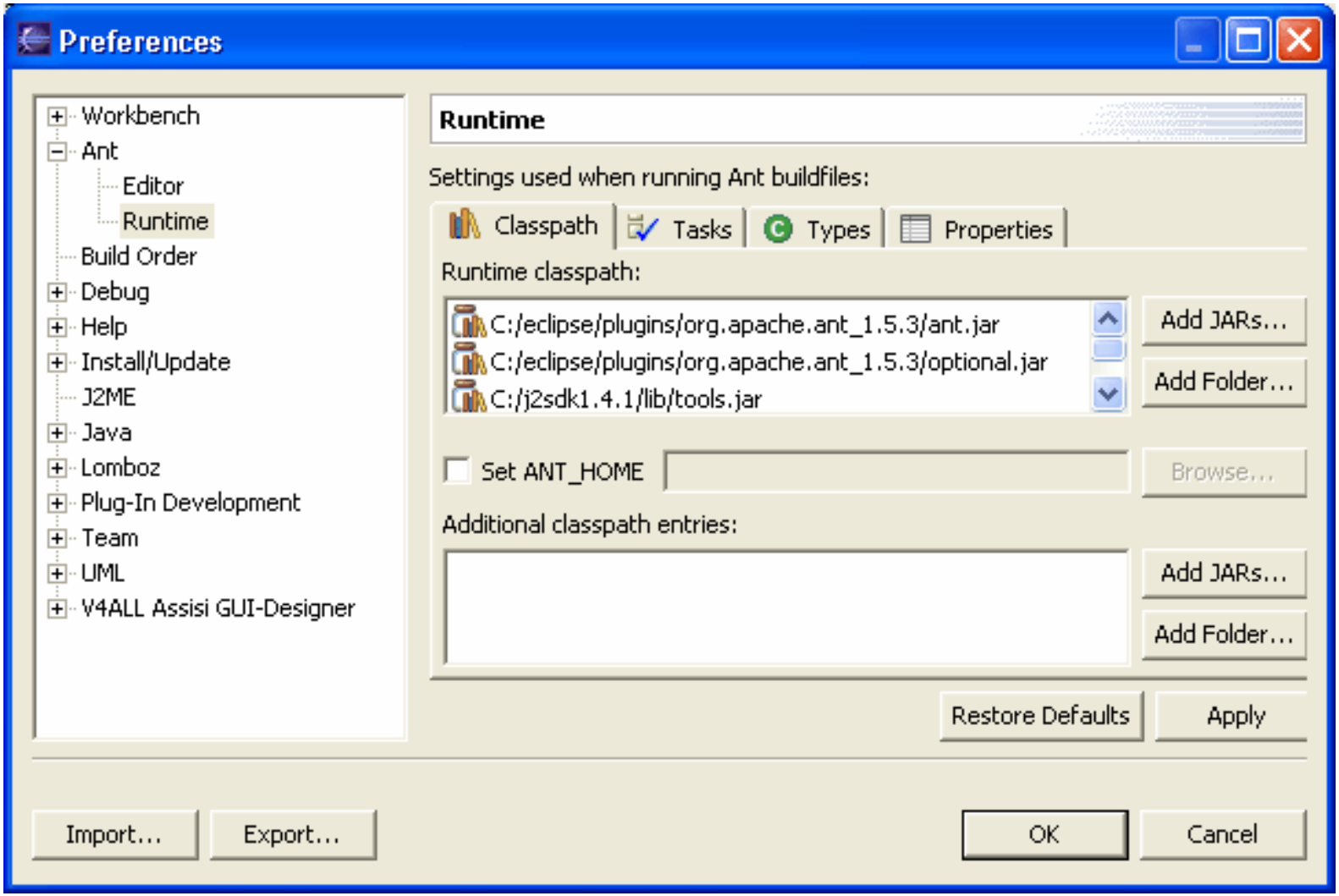
Impresión de texto (echo)

- Nos permite mostrar texto en la consola

```
<echo message="Instrucciones de uso:"
      level="info"/>
```

- Niveles:
  - "error"
  - "warning"
  - "info"
  - "verbose"
  - "debug"

Configuración

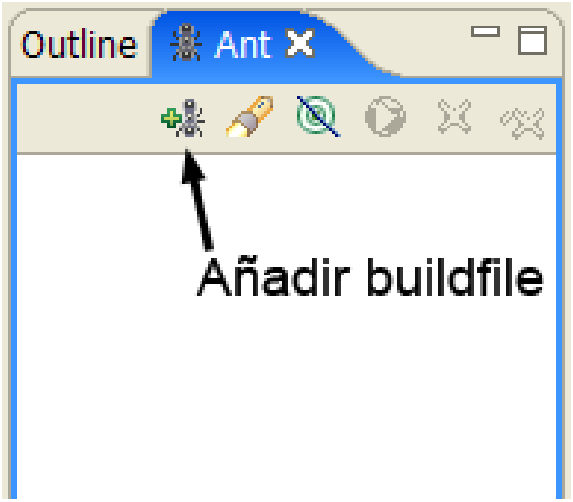


Crear build.xml

- Crear un proyecto
- Crear fichero genérico con *New* → *File*
- Llamar al fichero `build.xml`, en el raíz del proyecto
- Doble *click* sobre el fichero `build.xml` en el explorador de paquetes para abrir el editor de *Ant*
- Introducir el código de *Ant* en el editor y grabar

Ventana de Ant

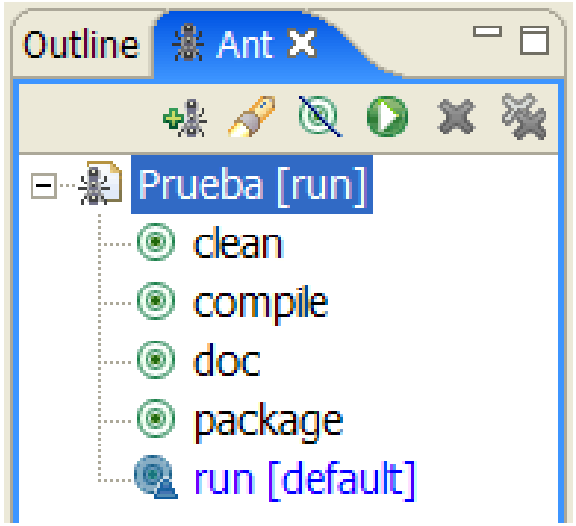
- Abrir ventana de *Ant* (*Window* → *Show View* → *Ant*)



- Añadir fichero de configuración de *Ant* a la ventana
- Seleccionar el fichero que acabamos de crear

Ejecución de objetivos

- Los objetivos definidos en el fichero se muestran en la ventana de *Ant*.



- Pulsar sobre un objetivo para ejecutarlo.