Ant

XML based build scripts

EHSDI

eBuzima

Ant

- Apache Ant is a software tool for automating software build processes
- Similar to MAKE, but:
  - Written in and developed primarily for Java
  - Uses XML scripts

Usage

- Ant can be run from the command line which means projects can be built without a large IDE such as Eclipse
- By default the command line client looks for a build script in the current directory called build.xml, e.g.

```
> ant
```

- If we need to use a different build script, we can specify the path

```
> ant -buildfile other.xml
```

Targets

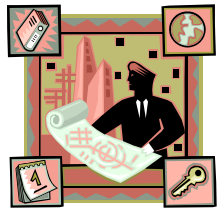
- Build scripts contain targets which are the different jobs that they can perform
- We specify the name of the target to run as an argument, e.g.

```
> ant compile
```

- If we don't specify a target, then the default one will be run, e.g.

```
> ant
```

Basic example: project



```
<?xml version="1.0"?>
<project name="HelloWorld" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="compile" description="compile the Java source files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Everything is contained in a **project** tag

Basic example: targets

```
<?xml version="1.0"?>
<project name=" HelloWorld" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="compile" description="compile the Java source files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Scripts are broken down into **targets** which are the different jobs that this script can perform

Basic example: tasks

```
<?xml version="1.0"?>
<project name="HelloWorld" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="compile" description="compile the Java source files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Targets are composed of **tasks** which are built-in functions

build.xml: project

- Every ant build file has a `project` element as its root element

Used to display it in Eclipse etc

The default target to run if none is specified

```
<?xml version="1.0"?>
<project name="HelloWorld" default="compile">
    <description>
        Ant build script for the HelloWorld project
    </ description >
    ...
</project>
```

Optional description element

build.xml: target

- Targets are like different jobs that the script can perform, e.g.

Used to display it in Eclipse and choose it from command line

Optional description

```
<target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
</target>
```

Targets contains tasks like this **delete** task

Dependencies

- Dependencies are relationships between targets, i.e. we can say that the execution of one target *depends* on the execution of another
- For example, we need our compile target to be executed before we can execute the jar target:

```
<target name="compile" description="compile the Java source files">
    ...
</target>
<target name="jar" depends="compile" description="create a Jar file">
        ...
</target>
```

Properties

- String constants can be defined as properties which is good practice for readability and re-usability, e.g.

```
<property name="classdir" value="classes"/>

<target name="clean" description="remove intermediate files">
        <delete dir="${classdir}"/>
</target>
```

System properties

- We can access various predefined system / environment properties in this way as well, e.g.

```
<property environment="env"/>

<target name="test" description="show path env variable">
        <echo message="path = ${env.PATH}"/>
</target>
```

All environment variables will be imported with this prefix

`echo` is a built-in task

Use prefix from above

Location properties

- If a property is a path then you should use the location attribute as its value will be modified for different file systems, e.g.

```
<property name="datadir" location="../data/files"/>

<target name="clean-data" description="remove data files">
        <delete dir="${datadir}"/>
</target>
```

On Windows `datadir` will become "..\data\files"

On Linux `datadir` will remain as "../data/files"

Time and date properties

- Ant also has three predefined properties for getting the current time:

Property	Format	Example Output
${DSTAMP}	yyyyMMdd	20090725
${TSTAMP}	hhmm	1605
${TODAY}	MMMM dd yyyy	July 25 2009

Tasks: echo

- This simple task writes text, e.g.

```
<echo message="Hello world!" />
```

- We can also write to a file instead of the console, e.g.

```
<echo message="Hello world!" file="msgs.txt" />
```

- A level can be assigned to a message to determine if it should be displayed:

```
<echo message="Hello world!" level="debug" />
```

Will only be displayed if ant is run with `-debug` flag

Tasks: mkdir

- This task creates the specified directory and any non-existent parent directories, e.g.

```
<mkdir dir="project/output/bin/test"/>
```

Creates the *project* folder if it doesn't exist, then creates the *output* folder.. etc

Tasks: delete

- This task deletes the specified directory or file, e.g.

```
<delete file="project/test.java"/>
```

Deletes the filed called *test.java*

```
<delete dir="project/test"/>
```

Deletes the folder called *test* and everything inside it

Tasks: javac

- This task invokes `javac` to compile a directory of Java files, e.g.

```
<javac srcdir="${src.dir}" />
```

- By default class files will be placed in the same directory, however we can specify the output directory, e.g.

```
<javac srcdir="${src.dir}" destdir="${classes.dir}" />
```

- It checks the modified time of each class file and only compile Java files which have been modified since they were last compiled

Tasks: jar

- This task invokes `jar` to create a JAR file from a directory of class files, e.g.

```
<jar jarfile="project.jar" basedir="${classes.dir}" />
```

Tasks: war

- This task builds a WAR file from a Java web application project, e.g.

Web app web.xml

```
<war destfile="myapp.war" webxml="src/metadata/web.xml">
    <fileset dir="src/html/myapp"/>
    <fileset dir="src/jsp/myapp"/>
    <lib dir="libs"/>
    <classes dir="build/main" />
</war>
```

HTML and JSP files

Java classes

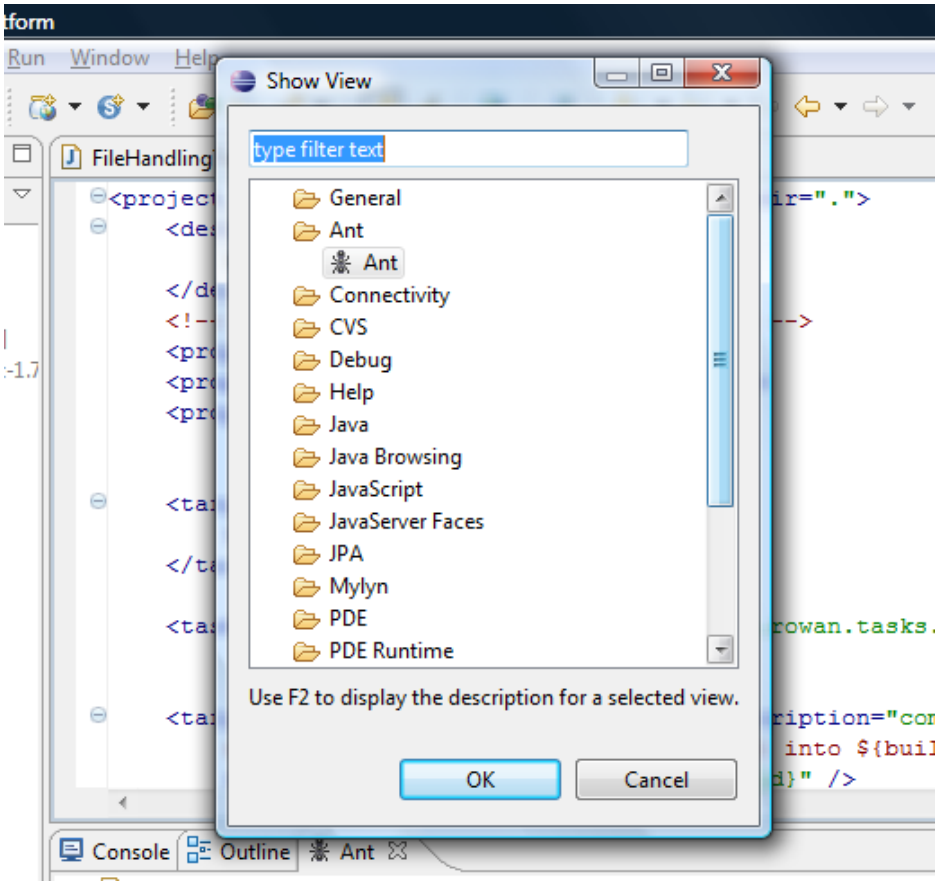
Other tasks

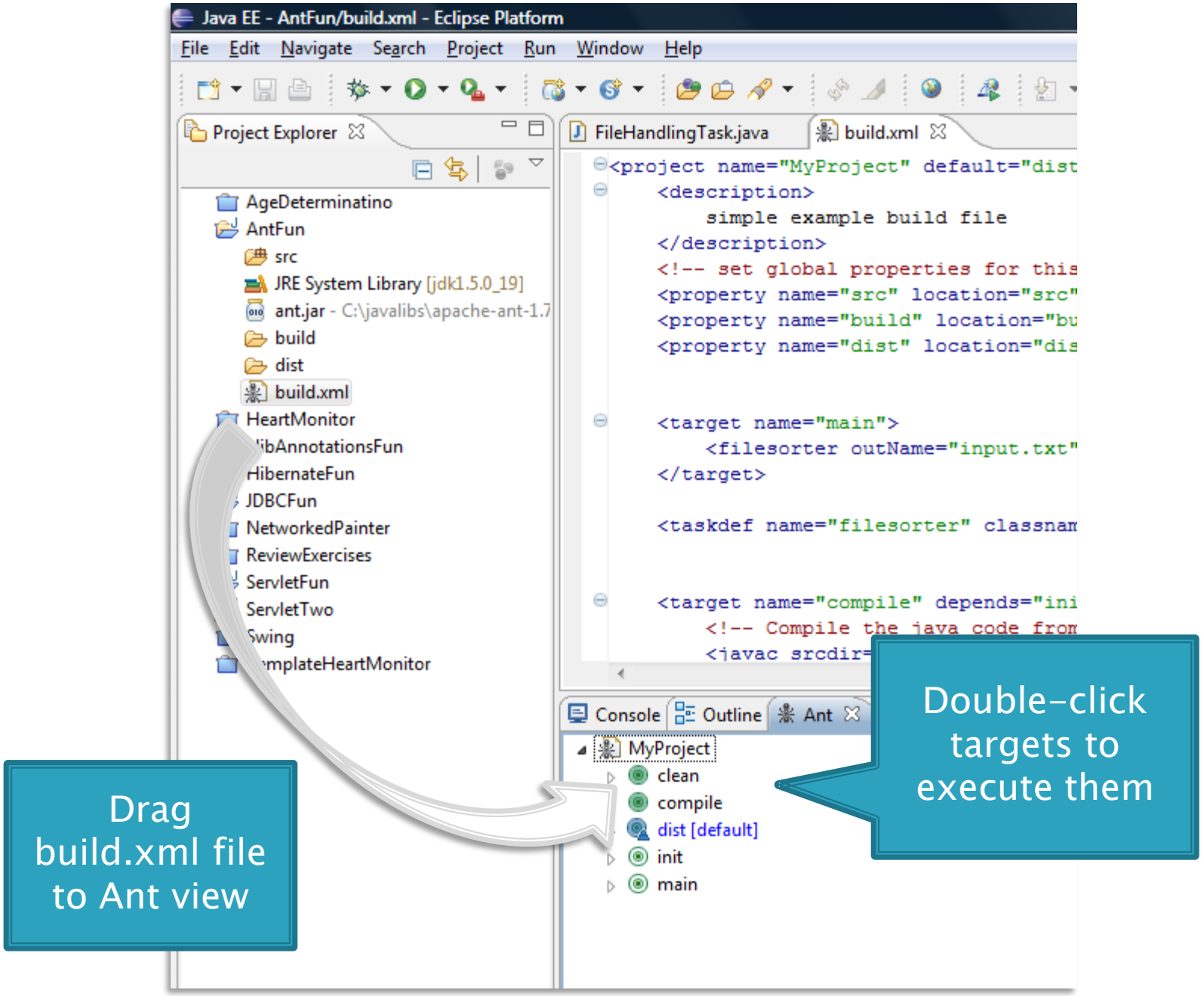
- For a complete list of built-in tasks see http://ant.apache.org/manual/
- You can also write your own tasks
  1. Extend the `org.apache.tools.ant.Task`
  2. Override the execute method
  3. Provide setter methods for all task attributes
  4. Import the custom task into your build script using <taskdef>, e.g.

```
<taskdef name="mytask" classname="com.mydomain.MyTask"/>
```

In Eclipse

- *Window > Show View > Ant*




Java EE - AntFun/build.xml - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Project Explorer
AgeDeterminatino
AntFun
src
JRE System Library [jdk1.5.0_19]
ant.jar - C:\javalibs\apache-ant-1.7
build
dist
build.xml
HeartMonitor
HibernateFun
JDBCFun
NetworkedPainter
ReviewExercises
ServletFun
ServletTwo
Swing
FileHandlingTask.java
build.xml
<project name="MyProject" default="dist
<description>
simple example build file
</description>
<!-- set global properties for this
<property name="src" location="src"
<property name="build" location="bu
<property name="dist" location="dis
<target name="main">
<filesorter outName="input.txt"
</target>
<taskdef name="filesorter" classnam
<target name="compile" depends="ini
<!-- Compile the java code from
Console
Outline
Ant
MyProject
clean
compile
dist [default]
init
main
Drag build.xml file to Ant view
Double-click targets to execute them

Resources

- Official site: http://ant.apache.org
- Manual: http://ant.apache.org/manual/
- Tutorial: http://ideoplex.com/focus/java#ant