Software Development Tools

COMP220/COMP285

Seb Coope

Ant and XML: Getting Started

These slides are mainly based on “Java Development with Ant” - E. Hatcher & S.Loughran. Manning Publications, 2003

Getting Started with ANT

First, check that **Ant** is installed:

```
H:\>ant -version
Apache Ant(TM) version 1.8.2 compiled on
December 20 2010
H:\>
```

Getting Started with ANT

- **Let us generate** a **Java *source file***

  `Main.java`

  and a ***build file***

  `build.xml`

  in the same, ***base directory***

  `C:\Antbook\ch02\firstbuild`

  We will use and extend this directory structure in the future considerations.

  Use the same naming of directories.

  In the lab, you will use the drive H: instead of C:

Getting Started with ANT

BUILD with **Ant** the following Java program **`Main.java`**

```
public class Main {

    public static void main(String args[]) {
        for(int i=0;i<args.length;i++) {
            System.out.println(args[i]);
        }
    }
}
```

1. *Compile it*
2. *Archive* (create a **`.jar`** file; later),
3. *Execute* (later).

Writing Ant build file

Create a file **`build.xml`** containing only one target:

```
<?xml version="1.0"?>
<project name="firstbuild" default="compile" >
  <target name="compile">
    <javac srcdir="."
           includeAntRuntime="no"/>
    <echo>compilation complete!</echo>
  </target>
</project>
```

It *compiles* all Java source code *in and below* the ***current directory*** **`"."`** according to **`javac`** **Ant** attribute.

```
srcdir="."
```

It is usually best to set above **`includeAntRuntime`** to **`"false"`** or **`"no"`** so the script's behaviour is not sensitive to the environment in which it is run.

Running you first build

Run Ant at the command prompt from this directory:

```
C:\Antbook\ch02\firstbuild>ant
Buildfile: C:\Antbook\ch02\firstbuild\build.xml

compile:
        [javac] Compiling 1 source file
         [echo] compilation complete!

BUILD SUCCESSFUL
Total time: 2 seconds
```

Look at the compiled file in the directory **`firstbuild`**.

Repeat this run again.

Do you see any difference with the above run?

Can you explain it?

Running you first build

- **Ant** compiled *all Java source* in the current directory (and all its subdirectories, if any) and
- printed a success **`<echo>`** message afterwards.
- ***Now*** you can see the resulting **`Main.class`** in

`C:\Antbook\ch02\firstbuild`

- This looks trivial, but **Ant** can do much more!

XML and ANT

The contents of the above file **build.xml** is an

***XML** document*.

*We will need **only most elementary concepts of** **XML***.

Read more, for example, in

*Annotated **XML** Specification* (Bray 1998)

http://www.xml.com/axml/axml.html

or a simple tutorial in

http://www.w3schools.com/default.asp

XML and **ANT**

Any XML document is a kind of **well-formed** and nested **bracket expression** (with some data in between) like

(((()())()))

where brackets are balanced.

Each left **"("** has corresponding **")"** to the right of it, etc.

No overlapping between any two bracket expressions

(something here) and **(something else here)**

is allowed.

Either one such expression is ***part of another (nesting),*** **or** they ***do not intersect (do not overlap)***.

XML and **ANT**

In **XML** so called *start, or opening tags* like

`<project>, <target>, <javac>`

play **the role of opening bracket "(",**

and corresponding *end, or closing tags*

`</project>, </target>, </javac>`

play **the role of closing bracket ")".**

XML and ANT

XML expression of the form

```
<anytag> anything here </anytag>
```

are called ***elements***

Elements can ***nest*** – can have *sub-elements (children)*, etc.

For example, in **ANT *build file*** the main **`<project>`** *element* can have **`<target>`** *sub-elements* (children) which can have **`<task>`** *sub-sub-elements* (grandchildren).

XML and ANT

Element

```
<anytag></anytag>
```

with ***no*** text or subelements between the tags is called an ***empty element*** , and is abbreviated as

```
<anytag/>
```

Note, that "**/**" is used ***at the end*** of the tag to denote the empty element.

If "/" appears at the beginning of a tag then this tag is considered as ***closing tag***. If "/" appears at the end of a tag then this tag denotes an ***empty element.***

XML and ANT

Any text can be written between tags, like here:

```
<echo>compilation complete!</echo>
```

Such a text is **not** considered as sub-element.

(Sub)elements should always start and end with *matching* ***tags***.

XML and **ANT**

Any XML document should have exactly one ***root*** element

(**ignore** the auxiliary first line

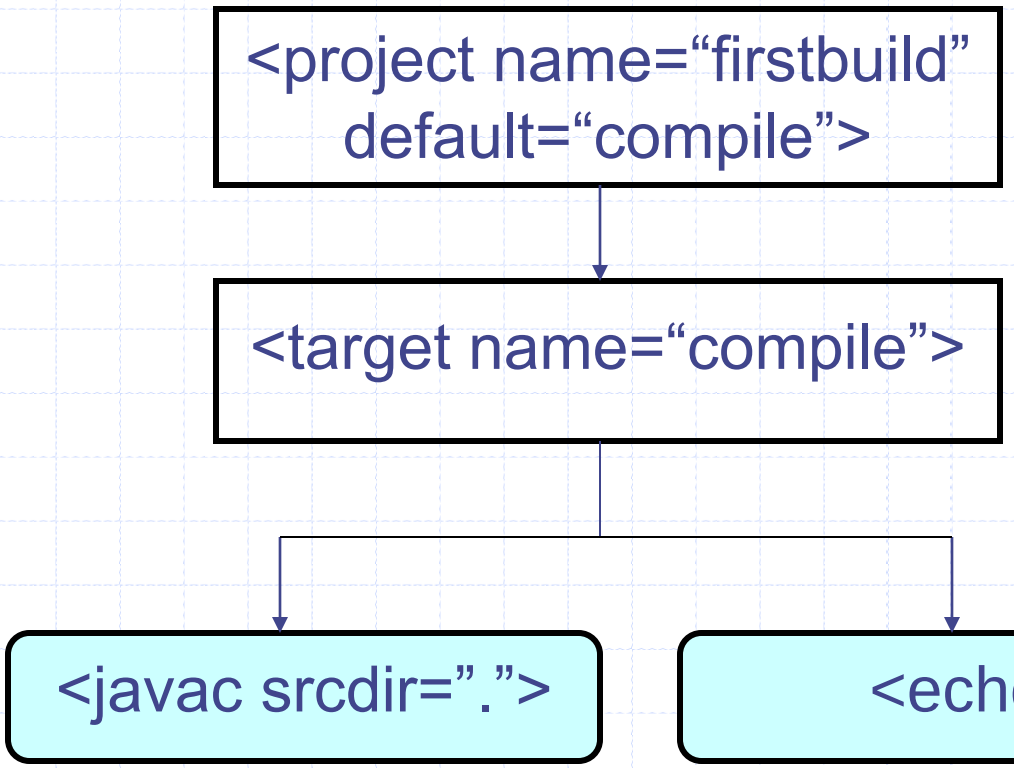
`<?xml version="1.0"?>`).

The root of any **ANT** build file is always a `<project>` element.

The root element contains all other elements as sub-...-sub-elements.

This leads to ***tree representation*** of any XML document.

Ttee representation of the above **build.xml** file



Compilation complete!

Caution!!

Do not mix such a tree representation of a build file with the graph of dependencies between targets considered earlier for another build file. (See Part 6, Slide 13.)

The goals and the meanings of these two different representations are quite different.

The ANT conceptual model

- A ***project*** contains ***targets***,
  - targets contain ***tasks;***
  - there can be further *nesting*
- The XML representation of a build file is a *tree*
- The above root ***project*** element contains only one ***target*** **"compile"**, which contains two ***tasks*** **<javac>** and **<echo>**

XML **attributes** in Ant build files

Start tags can optionally contain ***attributes*** like

```
name, default, srcdir
```

in the above file **build.xml**.

Example: **<target name="compile">**

This is the *start tag* of a target with the *name* **"compile".**

Attributes should have some text ***values*** like

```
name="firstbuild"
default="compile"
name="compile"
srcdir="."
```

Ant vs. HTML

- In general, **XML** *tags*, *attributes* and their *values* may be ***arbitrary***.
- However, in **HTML** and **Ant:**
  - tags, attribute names and values of some attributes have usually a ***predefined meaning***.
- Of course, in **HTML** and **Ant** this ***meaning*** is completely ***different:***
  - ***visualizing*** in **HTML vs**.
  - ***executing*** and ***other actions*** by tasks in **Ant**.
- Say, `srcdir="."` means that the *source directory*, i.e., directory containing source code files (`*.java`), is defined as the ***current directory*** `"."`

  (with respect to the base directory discussed later).

XML elements in **Ant** build files

The *compile* target element

```
<target name="compile">
  <!-- empty element javac: -->
  <javac srcdir="." includeAntRuntime="no"/>
  <echo>compilation complete!</echo>
</target>
```

Comment ignored by Ant

consists of two **task subelements** `<javac>` and `<echo>`

- `<javac>` compiles all source files from the current directory downward
- `<echo>` prints "`compilation complete!`" when the build process reaches that far.
- If the compilation task fails then the build will **fail** and **halt *before*** the echo message gets printed.

Summary

- **Ant** *build file* has one *root* **`<project>`** element containing several **`<target>`** elements (there are some *exceptions* discussed later).
- A ***target*** is a single ***stage*** in the build process.

  It has a unique ***name*** – arbitrary string (avoid spaces!).
- A target consists of several ***task sub-elements***.
- **Ant** is ***extensible*** : new useful ***tasks*** and other ***tags*** may be added to it

See "**Ant Tasks**" in
`C:\JAVA\Ant1.8.2\docs\manual\index.html` or
`http://ant.apache.org/manual/index.html`

for description of currently implemented **Ant** ***tasks***,

their ***attributes*** and ***nested elements*** that configure the task,

as well as handy ***examples***.