CSC

Build Automation with Ant

Thanh Vo
Senior Software Engineer

06/15/2016

Client Logo

Introduction

- Your role
- Your background and experience in the subject
- What do you want from this course

Course Objectives

- At the end of the course, you will have acquired sufficient knowledge to:
  - Understand Ant
  - Use Ant to build a project

Agenda

Course Audience and Prerequisite

- The course is for Java Freshers
- The following are prerequisites to the course:
  - Java fundamental

Assessment Disciplines

- Class Participation: 20%
- Final Exam: 80%
- Passing Scores: 70%

Duration and Course Timetable

- Course Duration: 6 hrs

Course Administration

- In order to complete the course you must:
  - Sign in the Class Attendance List
  - Participate in the course
  - Provide your feedback in the End of Course Evaluation

Introduction

What is Ant?

- Its not the insect that will bite you.
- Its not the wireless data transfer protocol like bluetooth.

Then what is ANT?

- Answer:
  - Ant is the acronym for Another Neat Tool.
  - Its a Java-based build tool with the full portability of pure Java code.

- About the author: James Duncan Davidson.(Also the author of Apache Tomcat) http://en.wikipedia.org/wiki/James_Duncan_Davidson

Why ANT?

- Its an open source project, maintained by Apache.
- OS independent: Builds run on both Windows and UNIX/Linux systems.
- You only need the JVM: It runs any where JVM is available.
- Works with anything that can be done from the command line
- Easy to extend (with Java)
- **Moreover its not a programming language.**

What can Ant do?

- Ant can get source code from version control
  - CVS, Subversion, Synergy, Perforce, ClearCase and many more
- Ant can compile source code
- Ant can run unit tests
  - JUnit3, JUnit4, TestNG, or any arbitrary test application
- Ant can package compiled code and resources
  - jars, wars, ears, tars, zips, whatever

Build Automation

- Automated build procedures are a best practice
- Manual procedures are mistake prone
- Automated builds are self documenting
- Automated builds improve productivity
- Automated builds can be triggered by other tools
  - Nightly builds using cron
  - Continous integration using CruiseControl, Jenkins

Using Ant

Install Ant

- Download the binary: http://ant.apache.org/bindownload.cgi

Latest version is 1.9.7 released on Apr 12, 2016.

Install Ant

- Unzip downloaded file into a directory
- Setup Environment Variables
  - Define ANT_HOME to be the location where Ant was unzipped
  - Define JAVA_HOME to be the location where the JDK is installed
  - Add %ANT_HOME%\bin to the PATH

Install Ant

- Lets Test the installation:

  Open the CMD and write: ant

- It should say:

  Buildfile: build.xml does not exist! Build failed

A Simple build file

-
  ```
  <?xml version="1.0"?>
  <project>
          <target name="hello">
                  <echo>Hello, World</echo>
          </target>
  </project>
  ```
- Lets save this as “build1.xml”

A Simple build file

- Goto cmd and move to the dir where you have put the simplebuild.xml
- Now write:

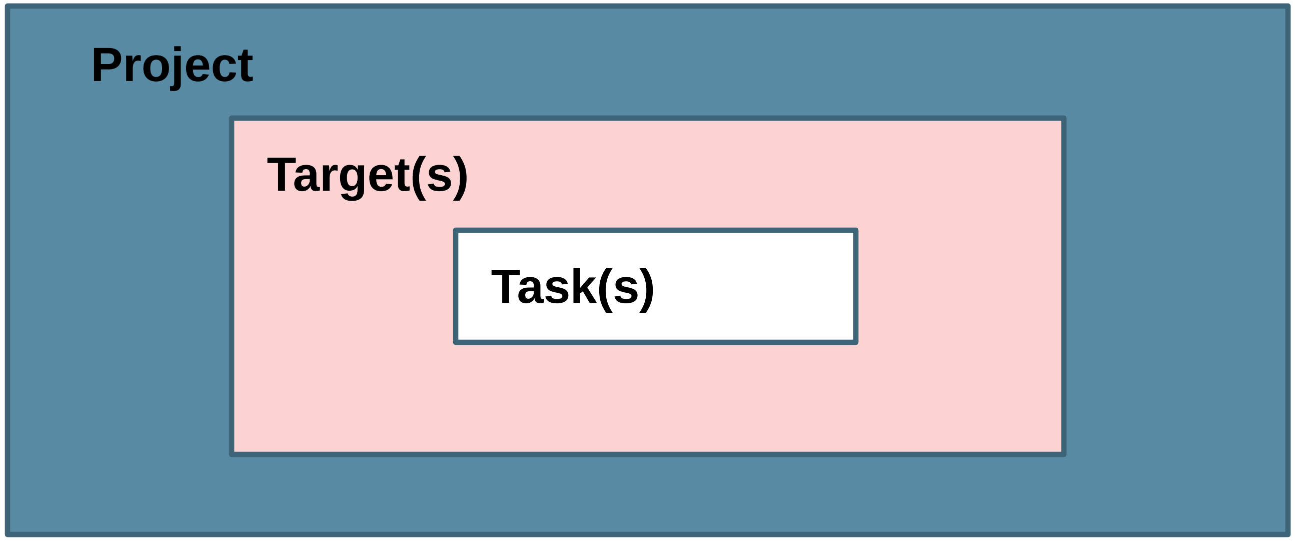
```
ant –file build1.xml hello
```

- This should show us “Hello World”.

Understanding Ant

The Structure

- Every build file will contain this three nodes:
  1. Project
  2. Target
  3. Task



The Project Tag

- Every thing inside the build file in ANT is under a project.
- Attributes: (None is mandatory)

  name > resembles the name of the project

  basedir > This is the directory from where all paths will be calculated. This can be overridden by using a "*basedir*" property. If both of these values are not set then "*basedir"* value is the directory of the build file.

  default > Defines the default target for this project. If no target is provided then it will execute the "*default*" target

The Target tag

- Targets are containers of tasks that is defined to achieve certain states for build process.
- Attributes:

  name > name of the target (Required)

  description > Description for the target (NR)

  depends > Which target this current target depends upon. A comma separated list.

  if > Execute target if value is set for a property

  unless > Execute the target if property value is not set

  extensionOf > *added from 1.8 (check next slide)*

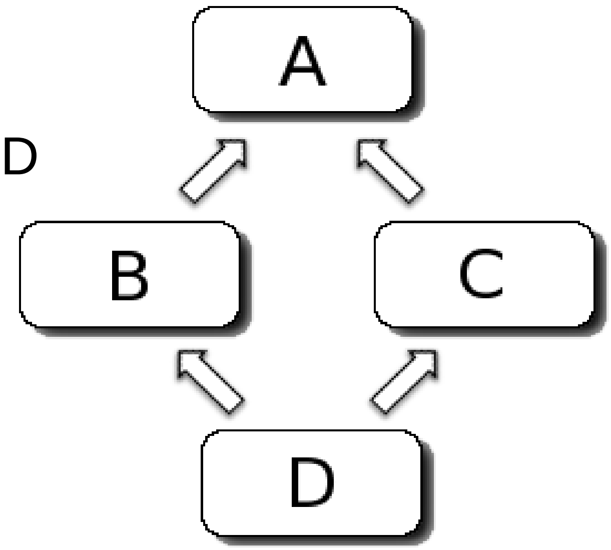
  onMissingExtensionPoint > *added from 1.8 (check next slide)*

The Target tag

- Each project defines zero or more targets
- A target is a set of tasks you want to be executed
- When starting Ant, you can select which target(s) you want to have executed
- When no target is given, the project's default is used
- Targets can be conditionally executed (using if/unless)
- A target can depend on other targets
- Target dependencies are transitive

The Target tag

- <target name="A"/>
  <target name="B" depends="A"/>
  <target name="C" depends="A"/>
  <target name="D" depends="B,C"/>
  Suppose we want to execute target D, which depends upon B and C
- C depends on A
- B depends on A
- so first A is executed, then B, then C, and finally D



The Target tag: Extension-point

- Extension points are like targets except they only use the depends property.

  <target name=“target_1”>…….</target>

  <extension-point name=“extension_point_1" depends=“target_1"/>

  If we want to add a target to this depends list then we will define the “*extensionOf* “attribute for that target.
  <target name=“target_2” extensionOf=“extension_point_1”>…….</target>

  Now if

  <target name=“target_3” depends=“extension_point_1”>…….</target>

  Execution for target_3 : target_1->target_2->target_3

  If the extension_point_1 is missing and “*onMissingExtensionPoint” is set then that target would be executed.*

The Target tag: An example for targets

<target name=“target_1” description=“depends on none”>

<property name=“property_1”></property>

…….

</target>

<target name=“target_2” description=“gets executed if property_1 has a value” if=“property_1”>…..</target>

<target name=“target_3” description=“gets executed if property_1 has no value” unless=“property_1”>…..</target>

The Task tag

- It’s the piece of code that can be executed.
- A task have multiple attributes or argument.
- The generic pattern for tasks:

  <*name attribute1*=“*value1*" *attribute2*="*value2*" ... />

- You can either use the *build in tasks* or you can *build your own task(Not shown here)*.

The Task tag: Build in Tasks

- File Tasks:

  <copy>, <concat>, <delete>, <filter>, <fixcrlf>, <get>

- Compile Tasks:

  <javac>

  – Compiles the specified source file(s) within the running (Ant) VM, or in another VM if the fork attribute is specified.

  <apt>

  – Runs the annotation processor tool (apt), and then optionally compiles the original code, and any generated source code.

  <rmic>

  – Runs the rmic compiler

The Task tag: Build in Tasks

- Archive Tasks:

  <zip>, <unzip>

  – Creates a zipfile.

  <jar>, <unjar>

  – Jars a set of files.

  <war>, <unwar>

  – An extension of the Jar task with special treatment web archive dirs.

  <ear>

  – An extension of the Jar task with special treatment enterprise archive dirs.

The Task tag: Zip and unzip a file with ANT

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="zip-test" default="zip" basedir=".">
  <property name="project-name" value="${ant.project.name}" />
  <property name="folder-to-zip" value="zip-me" />
  <property name="unzip-destination" value="unzipped" />
  <target name="clean">
            <delete file="${project-name}.zip" />
            <delete dir="${unzip-destination}" />
  </target>
  <target name="zip">
            <zip destfile="${project-name}.zip" basedir="${folder-to-zip}" excludes="dont*.*" />
  </target>
  <target name="unzip">
            <unzip src="${project-name}.zip" dest="${unzip-destination}" />
  </target>
</project>
```

The Task tag: Exec Task

- Executes a system command.

```
<target name="target" description="Creating dump file from MSSQL ">
		<property name="dump.filepath" location="./dump.csv" />
		<property name="mssql.user" value=""/>
		<property name="mssql.password" value=""/>
		<property name="mssql.host" value=""/>
		<property name=“bcp" location="C:/Program Files/Microsoft SQL
  Server/90/Tools/Binn/bcp.EXE"/>
		<delete file="${dump.filepath}" />
		<exec executable="${bcp}" >
				<arg line=' “put your query here"' />
				<arg line=' queryout "${dump.filepath}" -c -t, -S${mssql.host} -
  U${mssql.user} -P${mssql.password}' />
		</exec>
</target>
```

The Task tag: Antcall Task

```
<?xml version="1.0" encoding="utf-8"?>
<project name="Test-antcall">
  <target name="root_target">
      <antcall target="target1"></antcall>
      <antcall target="target2"></antcall>
  </target>
  <target name="target1">
    <echo>I am target 1</echo>
    <sleep hours="1" minutes="-59" seconds="-55"/>
    <echo>Waked up from sleep of 5 seconds</echo>
  </target>
  <target name="target2">
    <echo>I am target 2</echo>
  </target>
</project>
```

The Task tag: Mail Task

```
<target name="sendmail">
  <mail tolist=“"
            from=“"
            subject="Email subject"
            mailhost=""
            mailport=""
            ssl=""
            user=““
            password=“">
            <message>Example email sent by Ant Mail task.</message>
  </mail>
</target>
```

Please copy the mail.jar and activation.jar to the lib directory

Properties

- Properties are much like variables you declare.

```
<property name=“property_name” location=“some_value”/>
```

- This property can be used as a argument value of a task. e.g

```
<target name=“target1” description=“test target”>
        <property name=“property1” location=“place a value here”/>
        <mkdir dir=“${property1} “/>
</target>
```

- The property name is case sensitive.
- Properties are immutable: whoever sets a property first freezes it for the rest of the build

Properties: Attributes

- ***name*** > the name of the property to set.
- ***value*** > the value of the property.
- ***location*** > Sets the property to the absolute filename of the given file.
- ***file*** > the location of the properties file to load.
- ***resource*** > the name of the classpath resource containing properties settings in properties file format.
- ***classpath*** > the classpath to use when looking up a resource.
- ***environment*** > environment the prefix to use when retrieving environment variables. An example is given in build2.xml
- There are many more….

Command line options

- ant [options] [target [target2 [target3] ...]]
- Options:

  -help, -h

  -version print the version information and exit

  -diagnostics print information that might be helpful to diagnose or report problems.

  -verbose, -v be extra verbose

  -quiet to be quite (donot show any thing)

  -debug, -d print debugging information

  -lib <path> specifies a path to search for jars and classes

  -logfile <file>, -l <file> use given file for log

Command line options

-buildfile <file>, -file <file>, -f <file> use given buildfile

-D<property>=<value> use value for given property

-propertyfile <name> load all properties from file with -D

Points to Remember

Final Assignment

Final Assignment

- Write a simple web application
- Using Ant to deploy to Tomcat server
- Capture screenshot of each step and explain everything you did in the report
- Submit source code an report to the trainer

Q&A

Thank You

Client Logo

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By
06.14.16	1.0	Initial	Thanh Vo	

CSC

BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING