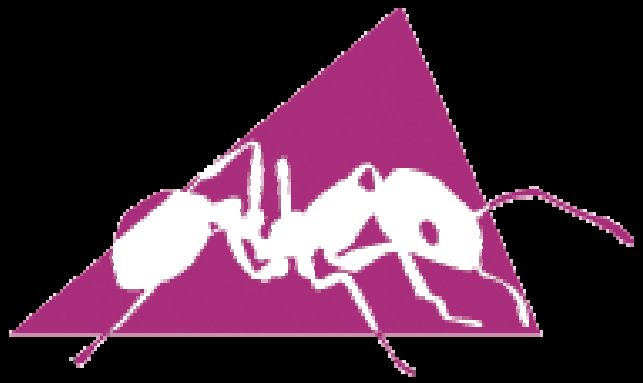
Introduction to ANT
Denver Java Users Group



Jerry Howard

JAXGurus and Associates

The Java And XML experts

Roadmap

- Introduction
- History
- XML Review
- Install and Configure
- Basic Concepts (with Examples)
- Large Project
- Examples
- Review
- Resources

Who am I

Jerry Howard is Senior Principal Consultant with JaxGurus and Associates.

- 13 years experience designing and developing successful software products.
- Covered designing, architecting, and implementing a variety of enterprise software applications and frameworks
- Areas of interest: B2B e-Commerce, Computer Graphics, Healthcare, Pharmaceutical, Travel, and Space Systems industries.

Introduction

- Provide you with a better understanding of Apache Ant.
- Show you how to install and configure Ant
- Cover the basic concepts
- To have fun!!

Some History

- Started as part of Apache Tomcat, mid 1998
- Created as a result of frustration with Make
- Original code was written on a few airplane trips.

  It's amazing what you can accomplish at 35,000 feet over the Atlantic. ;-)
- Ant continues to win development Awards:
  - Software Development magazine's 2002 Productivity Award
  - JavaWorlds Editor of Choice Awards

Why Ant is Special…

- Provides for a simple cross-platform build tool.
- Uses XML
- You can build, and deploy Jar, WAR, and EAR files
- Create daily builds, exec command
- You can do checkouts from source code systems (e.g., cvs, StarTeam, …etc)
- IDE integration (emacs, VisualAge,Jbuilder,…)
- Easily extendable to add your own commands
- It slices and dices….. 8-)

Ant is NOT…

- A Source code control system
- JAR file
- Silver bullet
- Jerry's Definition:

  Ant is a utility tool to assist developers, testers, software configuration, and IT folks in general to get their job done more efficiently

Quick Review of XML

- eXtensible Markup Language (XML)
  - XML is W3C standards based text for interchange of data
- XML describes information, or data
- Allows you to define your own tags. <JERRY>
- Must be well-formed
- ***Well-formed document*** - A document that has one root element within all other elements are neatly nested, but whose logical structure is not validated against the documents DTD/Schema.

  <NAME> <First> … </First> </NAME>

XML Node Types

- Document: Root node
- Element: <NAME> … </NAME>
  - Empty element (Marker): <MiddleName/>
- Attributes: <NAME First="Jerry" Last="Howard">
- Comments: <!-- Your comment -->
- CDATA section: <![CDATA[your text here]]>
  - Commonly used with special characters ("&", "<"), or you can escape them (&)

Installation and Configuration

- Download src or binary version from http://jakarta.apache.org/ant/index.html
- For Windows, you can dowload a "zip" file
- Extract into local directory
- Setup
  - set ANT_HOME environment variable
  - set JAVA_HOME environment variable
  - Add ANT_HOME/bin to your systems PATH environment variable
- Test: "ant -version"

Basic Concepts

- Ant is made up of Project, Target(s), and Task(s)
  - Project (CEO) - is root element of Ant file
  - Targets (middle mgmt)- These are broad goals that you have from your requirements.
  - Tasks (workers)-These are the smallest components of a build file. They perform the actual work

Basic Build File

```
<?xml version="1.0"?>
  <project name="foo" default="help"
  basedir=".">
  <target name="help" description="Usage text for build file">
    <echo message="Help file for build"/>
    <echo message=" ant [ant options] (target1) [(target2)
  ....]"/>
    <echo message="    help - show this message"/>
  </target>
  </project>
```

Project Element

- Root element in the build file
- Attributes:
  - name - Name of project
  - default - refers to default target to execute
  - basedir - defines root directory of the project.

Example:

```
<project name="Project Name" basedir="."
  default="help">
```

Target Element(s)

- Targets are the broad goals of the build. For example, compile and build Jar file
- Targets contain tasks that execute the necessary steps to complete a "target"
- Example:

```
<target name="targetName" description="desc"
   depends="target1,target2,…">
  <echo message="create testdir."/>
  <mkdir dir="testdir"/>
</target>
```

Task Element(s)

- These are the smallest components of a build file.
- These are the "worker ants" in your build file. They perform, or “do”, a specific task
- Common "Core" tasks
  - javac, javadoc
  - jar, tar, war, zip, gzip
  - copy, copydir, copyfile
  - and many others....

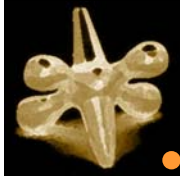
Optional Task(s)

- Optional tasks require additional Jar files (libraries)
  - EJB
  - Junit
  - IDE (specific)
  - Ftp/Telnet
  - .Net
  - Many others
- Let's go to the documentation and take a look

Example 2
Using Target

```
<?xml version="1.0"?>
  <project default="help" basedir=".">
    <target name="help“ description=“Usage Message”>
      <echo message="Help file for build"/>
      <!-- Note: message can't have "<" -->
      <echo message="  ant [ant options] (target1) [(target2) ....]"/>
      <echo message="    help - show this message"/>
      <echo message="    "/>
      <echo message="    "/>
    </target>
<!– Do More than mkdir, copy files, …. →
    <target name="init">
      <mkdir dir="jerry"/>
    </target>
  </project>
```

Traversal



- At the top level (project), Ant does a breadth first traversal, meaning
  - Ant processes all elements below the project element. Ant processes everything but Targets.
- For target(s), Ant does a depth first traversal. Ant processes each element as deep as it can before moving on to the next element.
- “processes element" – the element goes through the complete, full, life cycle on the existing element.

Example 3

```
<?xml version="1.0"?>
<project default="buildall" basedir=".">
  <target name="buildall"
     description="demonstrate order" depends="C, B, A">
  </target>
  <target name="A">
    <echo message="... execute A"/>
  </target>
  <target name="B">
    <echo message="... execute B"/>
  </target>
  <target name="C">
    <echo message="... execute C"/>
  </target>
</project>
```

Properties

- These are basic name/value pairs
- Properties have two characteristics:
  - Global
  - Can't change once processes
- You can also read properties from a file:

  <property file="myproperties.dat"/>
- You reference them by ${propertyName}
- Example:

  <property name="src.dir" value="src"/>

  <property name="java.dir" value="${src.dir}/main"/>

Properties(cont)

- Built in properties:
  - basedir – absolute directory which ant is running
  - ant.file – absolute path of current build file
  - ant.version – Ant version
  - ant.projectname – name of project as defined in project tag
  - ant.java.version – Java VM running

Note: properties are case sensitive

Data Types

- These are the building blocks used by tasks
- Data Types primarily simplify dealing with files
- Example: instead of one long line for property setting classpath

```
<path id="classpath">
  <fileset dir="${lib.dir}">
    <include name="**/*.jar"/>
  </fileset>
```

Pattern Set(s)

- Definition: Groups of patterns that allow you to filter files, or directories, based on a pattern
- Used with: DirSet, FileSet, …etc

- ? – Match single character
- * – Match zero or more characters
- ** – Match zero or more directories recursively

Example 4
Using javac

```
<?xml version="1.0"?>
 <project default="compile" basedir=".">
    <property file="build.properties"/>
    <property file="default.properties"/>
    <path id="classpath">
       <pathelement location="${servlet.jar}"/>
       <pathelement location="${xerces.jar}"/>
       <!-- You may include more -->
</path>
```

Example 4 (cont)

```
<target name="compile" description="compile code base"
    depends="init">
     <javac srcdir="${src.dir}"
          destdir="${build.dir}"
          debug="${debug}"
          optimize="${optimize}"   <!– Similar to –D option -->
          deprecation="${deprecation}">
      <classpath refid="classpath"/>
      <include name="**/*.java" />
    </javac>
</target>
</project>
```

Error Handling

- When it finds a problem it stops. <period>
- Ant does a syntax check before processing elements
- Two types of errors: (Project or Target)
  - reading elements
  - executing elements

Large Projects

- ***Cascading*** **build file- having multiple build files for a project**
- **Large projects may want to have build files for each subproject (e.g., Tomcat, Jboss)**
- **Allows you to easily manage build files (ie., keep them small and simple)**
- **Please include a README for building large projects. 8-)**
- **For small projects, recommend one file**

Large Projects(cont)

- Things to keep in mind
- Properties propagate from parent. If you do NOT want to inherit properties, set inheritAll attribute to false.
- Pre 1.5, Data Types do not propagate outside of the build file.

More Hands On Examples

< Lets go to the computer for some examples>

Example
Print current Ant Dtd

```
<?xml version="1.0"?>
  <project default="createdtd“
           basedir=".">
    <target name="createdtd">
     <antstructure output="project.dtd"/>
    </target>
   <!-- Then use XML Spy or TurboXML to get
   graphical view -->
  </project>
```

Review

- Covered concepts (targets, and tasks)
- Layout of build file
- Looked at several examples
- Discussed how to handle large projects

Home Work

- Review the features that I did not discuss.
- Learn how to write your own Task

Resources

- FAQ:
  - http://www.jguru.com/faq/Ant
  - http://ant.netbeans.org/
- Books
  - “Ant, The Definitive Guide”, Jesse Tilly and Eric Burke
  - “Java Development with Ant”, Erik Hatcher and Steve Loughran
  - “Ant Developers Handbook”, Williamson …et-al

Resources (cont)

- Articles:
  - http://jakarta.apache.org/ant/resources.html
  - http://www.onjava.com/pub/a/onjava/2002/07/24/antauto.html
  - http://www-106.ibm.com/developerworks/library/j-ant/
  - http://javaboutique.internet.com/tutorials/Ant/
  - http://www.javaworld.com/javaworld/jw-10-2000/jw-1020-ant.html

Contact Information

- Contact: jhoward@jaxgurus.com
  - http://www.jaxgurus.com
- JAX Gurus and Associates provides consulting services across multiple industries specializing in web services, technical management, training, and software development for companies, non-profits, and associations to build high quality software products.