Building Your Ant Colony

Designing an Automated Build System with Apache Ant

PUG Challenge Americas 2017
Build Systems
What is a Build System?

- Creates a “build” (Compile, Link, bundle)
- Automates tasks related to your codebase
- Reduces the amount of time you spend doing boring things
What’s the Business Case?

- Reduces manual boring stuff
- Easier to get new team members on board
- Continuous Integration
- Repeatable builds (Easier bug triage)
How much time do you spend on manual build tasks?
What can a Build System Do?

- Compile files
- Create files and directories
- Copy files and directories
- Delete files directories
- Rename files and directories
- Initialize data
- Run tests
- Run linting/static analysis
- Create docs from docstrings
- Download dependencies
- Zip/Unzip files
- FTP/Telnet/SSH

- Create a DB from a schema file
- Create a schema from a DB
- Index analysis
- Create a Procedure Library
- Create a REST WAR file
If you can script it, you can build it.
Build System Examples

- Make (C, C++)
- Cake (C#)
- Rake (Ruby)
- Grunt (JS)
- Pavement (Python)
- Phing (PHP)
- GB (Go)
- Ant (Java)
- Maven (Java)
- Gradle (Java)
- NAnt (C#)
- MSBuild (C#)

```bash
all: hello

hello: main.o factorial.o hello.o
    g++ main.o factorial.o hello.o -o hello

main.o: main.cpp
    g++ -c main.cpp

factorial.o: factorial.cpp
    g++ -c factorial.cpp

hello.o: hello.cpp
    g++ -c hello.cpp

clean:
    rm *o hello
```
Apache Ant
Intro to Ant

- "Another Neat Tool"
- Java Based
- XML Based
- Integrates with PDSOE / Eclipse
- Extensions written in Java
Getting Started

- Setup Ant Environment Variables
- Download and install Progress Compilation Tools (PCT)
- Create build.xml file
Consists of a `<project>` followed by some number of `<targets>`.

Each target can contain a number of directives called “tasks”.

Targets are run from the commandline via `ant target_name`.

```xml
<?xml version="1.0"?>
<project name="Hello World" default="hello">
    <task name="hello">
        <echo>Hello World!</echo>
    </task>
</project>
```

```bash
> ant hello
Buildfile: /path/to/your/build.xml

hello:
    [echo] Hello World!

BUILD SUCCESSFUL
Total time: 0 seconds
```
Core Concepts

- Properties
- Run-Time Parameters
- Dependencies
- Calling
- Filesets
Properties

- Build Configuration
- Variables that can be used in the build.
- Can be included directly or via an external file.

```xml
<property name="SrcDir" value="src" />
<property name="BuildDir" value="build" />
<property name="DocDir" value="docs" />
<property name="DbDir" value="db" />
<property file="build.properties" />
```

```properties
# build.properties

Dlc=C:\DLC116
Log=build.log
```
Run-Time Parameters

- Ant does accept run-time parameters from the command line, but it is clunky.
- Try to avoid more than one parameter.
- Spaces in the command are seen as different targets.

```xml
<target name="echo">
  <description>Echoes the input parameter.</description>
  <echo>${echo}</echo>
</target>

>ant echo -Decho="HELLO PUG!"
Buildfile: build.xml

echo:
  [echo] HELLO PUG!

BUILD SUCCESSFUL
Total time: 1 second
Dependencies

- Targets can “depend” on other targets.
- Ensures that the “dependee” target is run before the “depender”.
- Useful for separating complex tasks
- Downside: Make tasks harder to reason about

```xml
<!-- Ensures that "init" is always run before "build" -->
<target name="build" depends="init"/>
```
Calling

- Another way of running other tasks
- More like a function call
- Can conditionally execute

<antcall target="build" / >
Filesets

- Allows selection of files that a task acts on.
- Can black-list or white-list.
- Very powerful.

```xml
<copy todir="${BuildDir}"
     fromdir="${SrcDir}"
>
  <fileset dir="${SrcDir}"
           include="**/*.resx"
           exclude="**/exclude.resx"
    />
</copy>
```
Progress Compilation Tools

- Ant Plugin
- Provides a large number of OE related tasks
- Free, Apache Licensed
- [https://github.com/Riverside-Software/pct](https://github.com/Riverside-Software/pct)
Standard OpenEdge Build Targets

- Init
- Clean
- InitDB*
- CleanDB*
- Build
- Test
- Docs
- Package
- Install/Deploy
- Copy Resources
What do Ants and the Moon have in common?
Integrating with Eclipse
Running Tasks from PDSOE

- Can be run from “External Tools” or the “Ant” View
Create, manage, and run configurations

Run an Ant build file.

Name: Init

Buildfile:
${workspace_loc:/AntDemo/build.xml}

Base Directory:

Arguments:
init
Running Targets Automatically

- You can set build targets to run automatically after a PDSOE compile.
Edit launch configuration properties

Create a configuration that will run an Ant build file during a build.

Name: AntBuild

After a "Clean":
<default target selected> [Set Targets...]

Manual Build:
copy_resx [Set Targets...]

Auto Build:
<Builder is not set to run for this build kind> [Set Targets...]

During a "Clean":
<Builder is not set to run for this build kind> [Set Targets...]

Apply Revert OK Cancel
DEMO
Questions?
Advanced Usage

- Continuous Integration
  - Commit to Git -> Push to Gitlab -> Sends to Gitlab CI runner -> Build in new Container/VM
- Include Git Commit # in output
  - Run git via exec task to output SHA1 hash to a BUILD file.
- Deploy Via SSH/FTP
- Automatically Update Release Notes via Commit Logs
- Update Libraries
- Check listing/xref output
Further Reading


PCT Docs: https://github.com/Riverside-Software/pct/wiki

Github: Search for build.xml files or other build files.
Related Talks

- **430: The Future of OpenEdge build system**
  - Wed 9:45
- **201: The Tool-Stack Used by an OpenEdge Tool Vendor**
  - Tues 9:45
John Cleaver
Factivity, Inc.

Email: johnc@factivity.com
Talk: https://speakerdeck.com/jcleaver/intro-to-ant