Deployment

CSE 2311 - Software Development Project

Wednesday, February 27, 2013

Next week: Midterm evaluation

- A half-hour presentation as usual (except it counts for grades)
- The system must be available to install and try out locally
  - Post an installer or a zip file with instructions online
  - Functionality does not need to be all there, but the system must run
  - You **HAVE TO TEST** that the distribution you provide installs with no errors
- Early versions of documents must be submitted: Requirements, Design, Testing, User Manual

Software deployment

- The system may be running fine under Eclipse, but the customer needs a standalone system
- The code must be delivered to the customer and assembled and configured at their site
- Any dependencies (such as the iText library) must be transparent to the customer
- Deployment also includes maintenance, updating, and uninstalling

Software Deployment Methods

- The foot and hand model: Run around on foot and install software by hand.
  - **Only viable for small client base.**
  - **Expensive.**
- The self-service model: The end users install the software themselves.
  - **Scales well.**
  - **Low cost.**
  - **Becomes difficult as the complexity of installation and configuration increases.**

Things to learn

- How to build a a system and package it for delivery to the customer
  - We will use ant to do this
  - Maven is another tool you might want to look into
- How to deal with dependencies? Two options:
  - Ask the customer to install third party libraries
  - Bundle the library with your code
- To make things easier for the client, create an installer
  - We will use packjacket to do this

What is Ant?

- Java-based build tool from Apache
- De facto standard for building, packaging, and installing Java applications
- Accomplishes same objectives that *make* does on Unix based systems
- Files are written in XML

Based on a slide set by Ali Beyad

Why Ant?

- Unlike makefiles, Ant files work cross platform

  - No need for multiple, complex makefiles depending on the operating system.

  - Tasks declared in platform independent way; Ant engine translates to OS specific commands.

- Easy to create own Ant “tasks”, in addition to core tasks

Running Ant

- Type “ant” at the command line
- Automatically looks for build.xml file in current directory to run
- Type “ant –buildfile *buildfile.xml*” to specify another build file to run
- We will run ant through Eclipse

Ant Overview: Project

- Each build file contains exactly one **project** and at least one **target**
- Project tags specify the basic project attributes and have 3 properties:

  - Name, default target, basedir

- Example:

```
<project name=“MyProject” default=“build”
basedir=“.”>
```

Ant Overview: Targets

- Targets are build modules, e.g. “compile”
- Each target contains task(s) for Ant to do
- One target must match the project default target
- Example:

```
<target name="A"/>
<target name="B" depends="A"/>
<target name="C" depends="B"/>
<target name="D" depends="C,B,A"/>
```

Ant Overview: Tasks

- Each target comprises one or more tasks
- A task is a piece of executable Java code (e.g. javac, jar, etc)
- Tasks do the actual “build” work in Ant
- Ant has core (built in) tasks and the ability to create own tasks

Ant Overview: Task Example

```
<target name="build" depends="copy" >
    <javac srcdir="src" destdir="bin">
        <include name="**/*.java" />
    </javac>
</target>
```

YORK U
UNIVERSITÉ
UNIVERSITY

Ant Overview: Core Tasks

- javac – Runs the Java Compiler
- java – Runs the Java Virtual Machine
- jar (and war) – Create JAR files
- mkdir – Makes a directory
- copy – Copies files to specified location
- delete – Deletes specified files

Ant Overview: Properties

- Special task for setting up build file properties:
- Example:

```
<property name="src" value="/home/src"/>
```

- Can use **${src}** anywhere in build file to denote `/home/src`
- Ant provides access to all system properties as if defined by the <property> task

Ant Overview: Path Structures

- Ant provides means to set various environment variables like PATH and CLASSPATH.
- Example of setting CLASSPATH:

```
<classpath>
   <pathelement path="${classpath}"/>
   <pathelement location="lib/helper.jar"/>
</classpath>
```

Command Line Arguments

- -buildfile *buildfile* – specify build file to use
- *targetname* – specify target to run (instead of running default)
- -verbose, -quiet, -debug – Allows control over the logging information Ant outputs
- -logger *classname* – Allows user to specify their own classes for logging Ant events

IDE Integration

- Eclipse, NetBeans, JBuilder, VisualAge, and almost any other Java IDE has Ant integration built-in to the system
- Refer to each IDE’s documentation for how to use Ant with that IDE
- Let’s see a demo with Eclipse…

Handling dependencies

- To package libraries with the jar file, choose File -> Export… -> Runnable JAR File
- Click Next, and select a Run configuration that launches successfully
- You have the option to create an Ant script that does the same packaging that you can customize if desired
- For manual ways to package libraries, see link on course website

Documentation/References

- Download: http://ant.apache.org/bindownload.cgi
- User Manual: http://ant.apache.org/manual/index.html
- Sun's Web development tutorial (Ant and JSPs):

  http://java.sun.com/webservices/docs/1.2/tutorial/doc/GettingStarted3.html
- Java Development with Ant, by Erik Hatcher and Steve Loughran

Installers

- If a runnable jar is sufficient, it just needs to be posted online
- For more complicated installs, you might need to use an installer creator, such as Packjacket
- Quick demo…