Apache Maven and Android

Manfred Moser

simpligility technologies inc.
http://www.simpligility.com

About Manfred Moser

- Long time Linux user and Java developer currently working as Android application developer

- Author of the chapter “Android Application Development with Apache Maven” in the book Maven: The Complete Reference

- Committer on Maven Android Plugin and author of Maven Android SDK Deployer

- VIJUG

Now that you know a bit about me..

What about you?

Android development?
Other mobile/embedded development?
Java (SE, EE, ME...)?
Linux?
Apache Maven?
Other

simpligility technologies inc

Agenda

Motivation

Maven Introduction

Maven Android Tool Chain with examples

Discussion, Feedback

Oh .. and feel free to interrupt with questions.

Building Android Apps Currently

Eclipse ADT

Or

Default Apache Ant based build

Whats wrong with that?

You don't like Eclipse or don't want to depend on it for build.

You need additional features in the build.

You want command line and continuous integration server usage.

You need to work with multiple dependencies.

You want to reuse code from another project.

Example Dependencies

Make lib folder and copy jars files into it.

Check them into svn.

Upgrade means replace jar file and transitive dependencies.

Example Dependencies

Known to be a hazzle from years of Apache Ant usage on Java projects:

- Unknown dependencies
- Transitive dependencies
- No documentation
- No collision detection

Led to Maven, Ivy ...

Introduction to Apache Maven

"Software project management and comprehension tool"

Builds your software and much more

De-facto standard for Java software builds

Convention over Configuration

Java based but used for more than just Java

Couple of Things to Know About

pom.xml

Default Build Life Cycle

Plugins

Repository (local and remote)

Maven Invocation

mvn [options] [<goal(s)>] [<phase(s)>]

- Options – get list with mvn -h
- Goals – with syntax Plugin:PluginGoal
  - e.g. mvn android:deploy
- Phases – e.g. clean compile test install package
  - e.g. mvn clean install

Maven Options

-h, --help – Display help information
-D,--define <arg> - Define a system property
-e,--errors - Produce execution error messages
-f,--file - Force the use of an alternate POM file.
-N,--non-recursive -Do not recurse into sub-projects
-P,--activate-profiles <arg> Comma-delimited list of profiles to activate
-X,--debug - Produce execution debug output

Goals

different per plugin
behaviour can be defined in pom
e.g. mvn install:install

but does not have to be

mvn archetype:create ... mvn install:file ....
pass parameters in with -Dparameter=value

Phases

- predefined order of things that need to be done
- what happens is defined in pom and default settings (super-pom)
- additional plugins can be bound to lifecycle phases

pre-clean, clean, post-clean
validate, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test- sources, process-test-sources, generate-test-resources, process- test-resources, test-compile, test, prepare-package, package, pre- integration-test, integration-test, post-integration-test, verify, install, deploy

Plugins, Plugins, Plugins

Maven itself does nearly nothing
– just a container (M2 - Plexus, M3 - Guice)

Super POM defines standard configuration
which does a LOT

Lots of plugins available at apache, codehaus and beyond

Maven and Android

Maven Android SDK Deployer

Maven Android Plugin

M2Eclipse

M2Eclipse Android Integration

Maven Android SDK Deployer

Dependencies from Android SDK

into local or remote repository

Hello Flash Example

Packaging apk

Android dependency

Java source folder

Maven Android Plugin configuration

Maven Compiler Plugin

Using the plugin

Build
mvn clean install

Start emulator
mvn android:emulator-start

Deploy application
mvn android:deploy

Other plugin goals

- android:deploy
- android:undeploy
- android:emulator-start
- android:emulator-stop
- android:apk
- android:dex
- android:pull
- android:push

Using external dependencies

Add dependency

Everthing else happens automagically

Roboguice – Astroboy example

Unit tests

With testng or junit, also e.g. Cobertura, Emma,

MorseFlash example

Maps Extension

```
<dependency>
  <groupId>com.google.android.maps</groupId>
  <artifactId>maps</artifactId>
  <version>7_r1</version>
  <scope>provided</scope>
</dependency>
```

Instrumentation tests

Separate module with instrumentation test

Maven-android-plugin-samples apidemos

Advanced Android Usage

Reuse Android projects – dependency type “apksource”

Sign apk with jarsigner plugin

Zipalign release apk

Other Things You Can Do

Reuse e.g. Pojo's from server side application

Testing code coverage

Produce JavaDoc and more on website

Continuous integration builds

Ensure License Header in all files

Static analysis, site build and more ...

What would you like to do?

Might be possible already..

Looking to improve plugin and tools ..

Give us feedback, give it all a spin, create issues and more

Maven Resources

Apache Maven
http://maven.apache.org

Maven: The Complete Reference
and other books and resources
http://www.sonatype.com/book

M2 Eclipse
http://m2eclipse.sonatype.org/

Maven Android Resources

Maven Android SDK Deployer
http://github.com/mosabua/maven-android-sdk-deployer/

Maven Android Plugin
http://code.google.com/p/maven-android-plugin/

M2 Eclipse
http://m2eclipse.sonatype.org/

M2 Eclipse Android Integration link
http://code.google.com/p/m2eclipse-android-integration/

Android Jars into Maven Central
http://code.google.com/p/android/issues/detail?id=4577

Summary

Proven, ready alternative

More flexibility

More power

Lets get Google and others to wake up!

The End

Questions?

Ideas?

Any feedback welcome!