Workshop: Apache Maven for Android Development Pros and Teams

Manfred Moser

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

simpligility technologies inc

Audience Background

Android development

Eclipse

Apache Ant

Apache Maven

Android Maven Plugin

About Manfred Moser

- Android application developer and consultant
- Author
  - Maven: The Complete Reference
  - The Hudson Book
  - Repository Management with Nexus
- Core Committer on Android Maven Plugin
- Author of Maven Android SDK Deployer
- Trainer for Apache Maven and Sonatype Nexus
- Vancouver Island Java User Group Founder/Leader

Agenda

Motivation

Brief Apache Maven introduction

Maven Android tool chain – Setup and Use

Repository Management
and Continuous Integration
– Why and How

Workshop Materials

- Distributed on USB stick in class
- Or from
  - http://goo.gl/FD6pr
  - http://www.simpligility.com/document/android_maven_workshop_simpligility_materials_andevcon3.zip

Building Android Apps Currently

Eclipse ADT

Or

Apache Ant based build

Whats wrong with that?

You don't like Eclipse or Ant 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 tried extending the Ant scripts...

Example Dependencies

Make “libs” folder and copy jars files into it.

Check them into version control.

Upgrade means replace jar file and transitive dependencies.

Different SDK release break differently

Example Dependencies

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

- Unknown dependencies
- Transitive dependencies
- No documentation
- No collision detection
- Large project checkout

Led to Apache Maven, Apache 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 (like RoR)

Declarative rather than procedural

Maven and Android

Android Maven Plugin – core tool

M2Eclipse, m2e-android – for Eclipse users

Maven Android SDK Deployer – for SDK libraries use

Android4Maven – all optional for setup

Maven Android Archetype – project blueprints

Maven Installation

Unzip, create M2_HOME, add to PATH

and then just run

mvn -version

Needs to be 3.0.3+

Demo with launchd, exports..

Getting started

Convert your existing app – add pom.xml file

Create new project – use command line or IDE wizard to create normal Android project, then add pom.xml

Use Maven Android Archetype – on command line or in IDE wizard

HelloFlashLight Example – pom.xml

Packaging apk

Android dependency

Java source folder

Android Maven Plugin configuration

Doing a build

mvn clean install

Invokes the **clean** and the **install**

build life-cycle phase

Life-cycle Phases

- predefined order of tasks for build
- what happens is defined in pom (+ super-pom)
  - show super pom with help:effective-pom
- additional plugin goals can added
- run mvn to see full list of lifecycle-phases
  - demo that ..

Deploy apk

mvn android:deploy

Invokes the
deploy **goal**
of the
android maven plugin

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

Plugins

Maven itself does nearly nothing
– just an IOC container (using Guice)

Super POM defines standard configuration
which can do a LOT

Lots of plugins available at apache, codehaus and beyond

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. compile test install
  - e.g. mvn install

Plugin Execution

- Manually
  - call plugin goal on command line
- Automatically
  - bind plugin goal execution to lifecycle-phase
  - create execution in configuration section of plugin
  - multiple executions possible
  - with separate configuration

Downloading the internet? Not!

Core install of Apache Maven very small (<5mb)

What is needed gets downloaded into repository..

Repository

Storage for plugins and dependencies -> artifacts

Local user repository ~/.m2/repository

Central Repository – http://search.maven.org

Other public repositories

Use your own repository server!

Recap

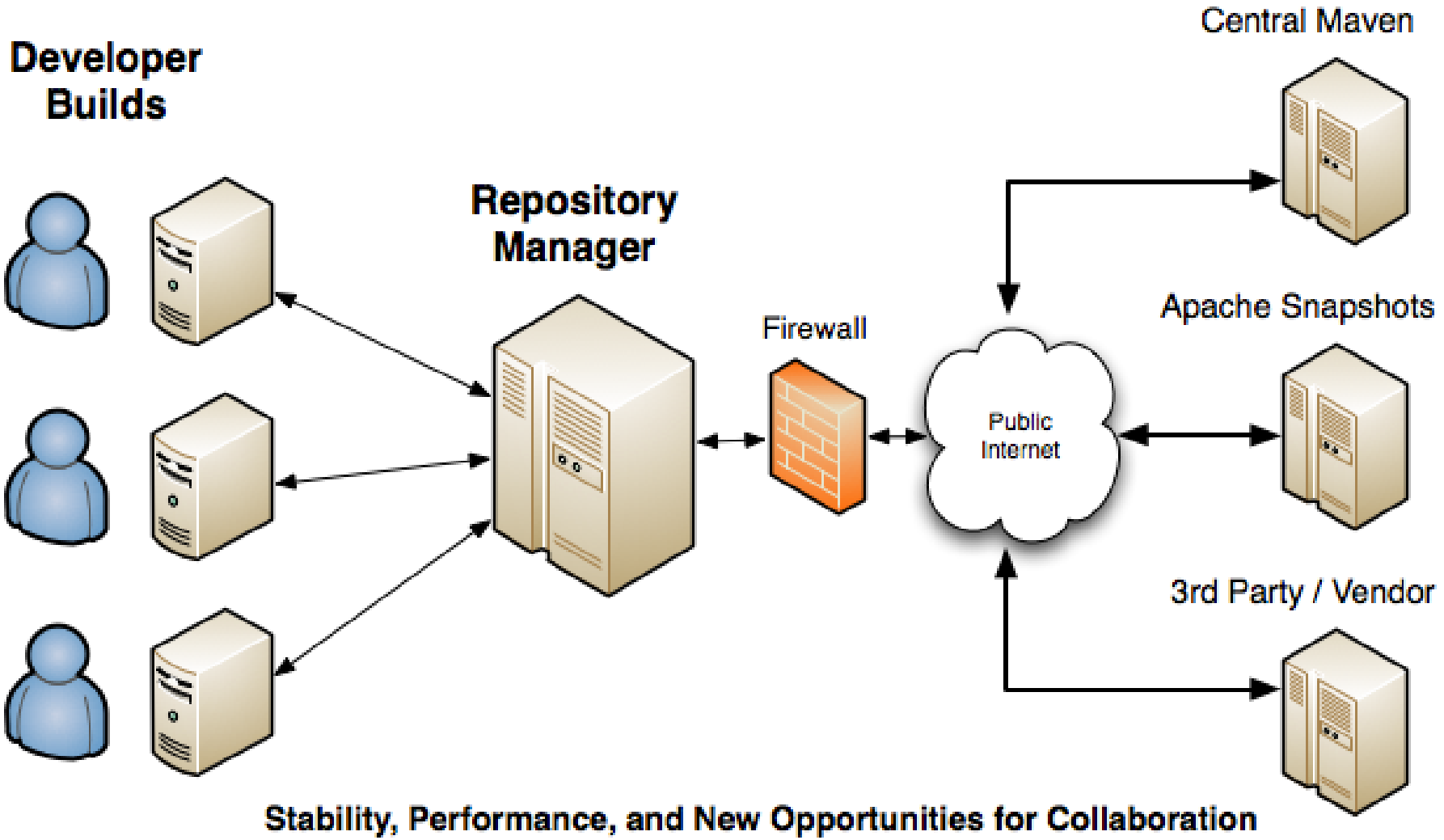
pom.xml

Default build life-cycle with phases

Plugins with goals

Repositories

Repository Manager



Repository Manager Advantages

- Proxy/cache artifacts from public repositories
- Improve speed and reliability
- Saves bandwidth
- Allows better control over what is downloaded
- Store and share internal libraries
- Store and distribute proprietary third-party libraries
- Need for different user privileges

Collaboration Point

- It does not matter how the artifact is built
- It does not matter what tool retrieves the artifact
- What matters is that it is a shared repository
- And that correct metadata is provided (the POM)
- A Maven repository uses a standardized API.

Available Repository Managers

- Sonatype Nexus
  http://www.sonatype.org/nexus/
- Artifactory
  http://www.jfrog.com/
- Apache Archiva
  http://archiva.apache.org/

Getting Started with Nexus

- Extract archive
- ./bin/nexus console
- settings.xml
- Watch start of proxying during a Maven build

Exploring Nexus

- Repositories
  - Proxy
  - Hosted
  - Group
- Search
- Settings
- Security
- Lots more

Using external dependencies

Add dependency

Everything else happens “automagically”

Repository Manager can host or proxy artifacts

Roboguice – Astroboy example

Other external dependencies

Google Guice IoC
KSOAP2-Android
Jackson JSON Parser
RoboGuice
ActionBarSherlock
Robolectric
Robotium
GoogleAPI's, CompatibilityPackage, AdMob..
and so on and so on

How did it find android.jar?

Deployed to the Central Repository

Built from AOSP with Android4Maven

Or

Deployed to repository with
Maven Android SDK Deployer

Let's do a demo of that shortly...

Google Maps jar

Can't be in Central Repository (not open source)

Use Maven Android SDK Deployer

Copies jars from SDK install to repository

Same for usb.jar, compatibility libraries...

Maps Dependency

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

Dependency Scopes

Default = compile

provided

test

Use Case 1

- Maven Android SDK Deployer
  - Copies artifacts from Android SDK to repository
  - Configure settings.xml with username/password
  - mvn deploy, not install!

- Now everybody in your organization has access to the android jars, admob, maps and so on
- Demo time!

Use Case 2

- HelloFlashLight application
  - Deployment with snapshot version
  - Deployment with release version
  - Staging suite has tiered deployment and approval process
  - Deployment of site as well
- Repository manager provides an access point anybody to download the latest apk
- Demo time!

Use Case 3

- ksoap2-android
  - Not in Central Repository
  - Add repository to pom is bad practice
  - Proxy repository in Nexus instead

- By adding proxy repository to Nexus, one step makes it available to all your developers and CI servers and so on
- Demo time!

Use Case 4

- Patch for ActionBarSherlock
  - Assume you have a patch for ABS
  - Jake is busy and has no included it yet
  - You need to use it now!
  - Easy – add classifier to version and build and deploy ABS with your patch to your repository manager
  - Later when Jake took your patch and cut a new release you just change the version

Back to the Android Maven Plugin

Efficient usage patterns

- IDE integration in Eclipse, IntelliJ and Netbean
  - Showcase archetype creation in Eclipse
- Command line alias
  - Part of materials – demo time
- Bash command line completion
  - Part of materials – demo time
- Run CI server like Hudson in background
  - More about that later

Working with the Android emulator

mvn android:emulator-start

mvn android:emulator-stop

mvn android:emulator-stop-all

With the parameters
android.emulator.avd
android.emulator.wait
android.emulator.options

Demo time

Manfred Moser
simpligility.com

Configuring the Plugin

Using the plugin configuration in

- pom.xml
- settings.xml

```
<plugin>
    <groupId>com.jayway.maven.plugins.android.generation2</groupId>
    <artifactId>android-maven-plugin</artifactId>
    <configuration>
        <sdk>
            <platform>8</platform>
        </sdk>
        <emulator>
            <avd>22</avd>
        </emulator>
```

Configuring the Plugin

Using properties

- in Maven files

```
<properties>
    <android.emulator.avd>22</android.emulator.avd>
</properties>
```

- on command line

```
mvn android:emulator-start -Dandroid.emulator.avd=22
```

Device Interaction

android.device parameter ->
usb, emulator or serialnumber

Works against all adb attached devices!

- deploy, undeploy and redeploy
- run
- pull and push
- instrument
- devices (coming today!)

Demo

Install and Run Apk

Goals

- deploy
- undeploy
- redeploy
- run

undeployBeforeDeploy

remove old version of application before testing

Deploy any apk

- Deploy an application as part of e.g. preparing avd for CI build
  - Droidreader for PDF testing
  - OpenOffice document reader for odf
- Example droidreader apk install (android.file parameter)

Pull and Push

android:pull and android:push goals

Works against all attached devices

Upload database or image resources to device or emulator avd

Download performance .trace files

Testing

Unit & Integration Testing

Various tools Junit, TestNG, EasyMock, Robolectric, Robotium ...

All as part of build

Tests run by default

-DskipTests or -Dmaven.test.skip=true

Instrumentation Testing

SDK Instrumentation as well as Robotium

Automatically runs on all attached devices

Produces junit xml reports per device

Lets see that later on the CI server

AndroidManifest Changes

manifest-update goal

Support for

- manifest.versionName
- manifest.versionCode
- manifest.versionCodeAutoIncrement
- manifest.versionCodeUpdateFromVersion
- manifest.sharedUserId
- manifest.debuggable

Use in release profile or manually

See Morseflash example

Maven resource filtering

Define properties or use an existing one

Define filtered resources

Embed properties for replacement

Can also be used to exclude files (e.g. SVG source for images)

Development vs Production

Resource filtering with different values for properties

Depending on build time switch (e.g. automatic or provided at invocation)

Uses profiles

Morseflash example

Profiles

Separate build definition
e.g. properties
but also
plugin configuration/execution/...

Activation via switches, files, OS...

Releasing Application

Perform release with Maven Release Plugin

Will handle scm operations, version changes...

Take advantage of repo server for artifact storage

HelloFlashLight demo

Sign apk

Maven Jarsigner Plugin

keys, password ... settings.xml

HelloFlashLight example

Zipalign

Android Maven Plugin

zipalign goal

Needs to be bound to lifecycle phase

HelloFlashLight example

Proguard

Obfuscate, shrink and optimize

Use proguard goal of Android Maven Plugin

Only in release profile

HelloFlashLight example

Native Application Build

Fully supported with Android NDK install

Various examples as part of Android Maven Plugin Samples – native

Scala

special use of a library jar

enables Scala programming language

scala plugin for compilation

proguard for shrinking

Scala example

Screenshots

- Feature of Android Maven Plugin
- Invoked from instrumentation test code
  - poseForScreenshot()
  - MorseFlash example
- Saved in target folder

Screen size, platform level

emulator-start, emulator-stop

different profiles

different parameters for deploy

deploy to different emulators/devices

Code Coverage

- Using Emma
- Coverage file can be downloaded to computer
- Set up in apidemos-android-10

Produce project website

Maven site plugin with javadoc and so on

Can be deployed to Nexus

Sonar for historical trending and more

Static code analysis

As part of build

Findbugs

PMD

Checkstyle

Reuse projects

Plain java library project from server side application

Android projects – dependency type “apklib”

Android library projects – with r14+ of SDK jar files

Android NDK .so files – as normal dependency

Reuse project examples

- Plenty out there in the wild
  - Android Maven Plugin samples project
  - RoboGuice Astroboy example
  - All ActionBarSherlock examples
  - Gaug.es from GitHub
  - blueBill Mobile Android version
  - and many more

What is Continuous Integration?

- Run build and tests for each check in
- Setup for various development branches
- Run release build as one click action
- Create website with project details as well as analysis of build history
- Provide user interface for non development user e.g. to publish release

Why Continuous Integration

- Avoid “works on my machine” problems - Reproducibility
- Free up developer machine/time – I don't have time to run all tests before each commit
- No IDE dependency (less setup problems)
- Rapid feedback in team
- Improved communication

Continuous Integration Servers

- Hudson/Jenkins
- Cruise Control
- Bamboo
- TeamCity
- and lots more

Commonalities for Setup

- Need to get tool chain on master/slaves
- Get a command line focused build working
- Integrate with SCM
- Look for specific plugins that might help
- Work with deployment platform like web browser, emulator, actual hardware

Installation of CI

- Headless install of Android SDK and build tools
- Install
- Configure
- Watch it run and be notified

Options for Install

- On demand on development machine
- Local networked server
- Virtual machine in cloud
- Commercial offering

Example Eclipse Hudson

- Easy to install
- Large community
- Android plugin
- Commercial offering as hosted
- Open source

Eclipse Hudson and Android

- JDK
- Android
  - SDK (for Java and normal API type Android app)
  - And potentially NDK (for C support, e.g. games)
  - And platform versions (1.6, 2.1, 2.2, 3.0...)
- Apache Ant or Apache Maven for command line build
- Install manually, via scripts, via puppet or VM snapshots..

Building an Android App

- Default build
  - Within Eclipse/Android Development Toolkit ADT
  - Optionally command line Apache Ant
- Better with Apache Maven
  - Dependency management
  - Work with repo for release management
  - Advanced features for testing
  - and lots more
- Possible Gradle, SBT, bash, scons...

Why do we need emulators/devices?

- No need for normal plain build
- But build should have testing!
  - Instrumentation testing runs on emulator/device

Android Emulator Plugin

- Create, start and stop emulator(s) for each run
- Capture logcat from emulator/device

Android Maven Plugin

- Start and stop emulators
- Automatically deploy to all attached devices
- Run tests on all attached devices
- Produces junit reports compatible xml on build machine

CI for Other Platforms

- As long as you can run a build on the command line..
- Characteristics of setup for other platform will be similar
  - Java ME
  - iOS
  - Windows Mobile 7
  - Phone Gap and other cross platform tools
- More or less painful depending on operating system requirements, tool chain needs, ...

What lies ahead?

- Automated UI tests using screenshot diffs
- Install and upgrade tests
- Performance tests
- Stability tests (low memory, memory usage...)
- ... towards
- Deployment automation to many markets
- Continuous deployment

Your Android build needs

Apache Maven
and
Android Maven Plugin can help!

Easy to script more features (Ant, Groovy,...)

All open source tools

We take pull requests!

Summary

Proven, ready alternative

More flexibility and power out of the box

Tools beyond build available

Better for test driven development
and continuous integration

Better reuse of components
and therefore in team environment

The End

Thank you for your attention.

Contact me for consulting
and open source hacking
and more

@simpligility

manfred@simpligility.com

http://simpligility.com

You Want More...

Lightning Talk
Tuesday, 5pm

Fireside Chat
“Tricks of the Trade”
Wednesday, 7:30pm

Free signed copy
“Repository Management with Nexus”
Thursday, 11:30am

Training

- Apache Maven and Nexus classes with Sonatype
- Remote classes with me as instructor
- On site possible too

Resources Apache Maven

Apache Maven
http://maven.apache.org

Maven: The Complete Reference
and other free books
http://www.sonatype.com/Support/Books

M2 Eclipse
http://eclipse.org/m2e/

Maven Bash Completion
https://github.com/juven/maven-bash-completion

Resources Android Maven

Maven Android Developers Mailing List

http://groups.google.com/group/maven-android-developers

Android Maven Plugin

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

M2 Eclipse Android Integration

http://rgladwell.github.com/m2e-android/

Maven Android SDK Deployer

https://github.com/mosabua/maven-android-sdk-deployer

Maven: The Complete Reference

Android Chapter

http://www.sonatype.com/index.php/Support/Books/Maven-The-Complete-Reference
http://www.sonatype.com/books/mvnref-book/reference/android-dev.html

Resources Android Maven

Android4Maven
http://sourceforge.net/projects/android4maven/

Android Maven Archetypes
https://github.com/akquinet/android-archetypes/wiki

Maven RIndirect Plugin
http://www.github.com/akquinet/android-rindirect

Android Maven Examples

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

RoboGuice
http://roboguice.org

Robotium
http://robotium.org

Robolectric
http://robolectric.org

ActionBarSherlock
http://actionbarsherlock.com/

Some more examples at
http://code.google.com/p/maven-android-plugin/wiki/PeopleUsingThis

and many many more

Resources Repository Managers

- Sonatype Nexus
  http://www.sonatype.org/nexus/
- Repository Management with Nexus
  http://sonatype.com/books/nexus-book/reference/
- Artifactory
  http://www.jfrog.com/
- Apache Archiva
  http://archiva.apache.org/

Resources Continuous Integration

Hudson
http://hudson-ci.org/
http://eclipse.org/hudson

Jenkins
http://jenkins-ci.org/

CruiseControl
http://cruisecontrol.sourceforge.net/

AtlassianBamboo
http://www.atlassian.com/software/bamboo/

JetBrains TeamCity
http://www.jetbrains.com/teamcity/

and many more

Hudson/Jenkins Resources

- The Hudson Book
  http://www.eclipse.org/hudson/hudsonbook
- Jenkins: The Definitive Guide
  http://www.wakaleo.com/books/jenkins-the-definitive-guide
- Android Emulator Plugin
  https://wiki.jenkins-ci.org/display/JENKINS/Android+Emulator+Plugin