adesso business. people. technology.

Apache Buildr in Action

A short intro

gearconf 2012

Felix Müller, adesso AG

25.06.12

About me

- Software Engineer @ adesso AG, Berlin
- Main focus on
  - Web development on the JVM
  - Build Management
  - Continuous Integration / Delivery / Inspection
- Open Source enthusiast

- @fmueller_bln
- felix.mueller@adesso.de

Agenda

- Why another Build System?
- A bit history
- Buildr catchwords
- Tasks
- Dependency management
- Testing
- Other languages
- Extending

Any aggressive Maven fanboys here?

http://www.flickr.com/photos/bombardier/19428000/

Collected quotes about Maven

“Maven is such a pain in the ass”

http://appwriter.com/what-if-maven-was-measured-cost-first-maven-project

Maven sucks...

- Configuration over Convention
  - Inconsistent application of convention rules
  - High effort needed to configure
- Documentation
  - Which documentation? (ok, gets better)
- “Latest and greatest” plugins
  - Can cause: not reproducible builds

Other buildsystems

- Ant
  - Still good and useful, can do everything... but XML
- Gradle
  - Groovy based
  - Easy extensible
  - Many plugins, supported by CI-Tools
- Simple Build Tool
  - In Scala for Scala (but does it for Java, too)

Apache Buildr

- Entered 2007 Apache Incubator
- Since 2009 Top-Level Project
- Coming from Apache Ode as Maven doesn't fit the needs
  - Like Ant (Tomcat) and Maven (Turbine) before

Why Buildr in Apache Ode?

~ 5.500 lines of POM XML

~ 500 Buildr lines

Apache Buildr catchwords

- Why Buildr?
  - Unmaintainable, inflexible Maven Scripts
  - Missing loops
  - Missing DRY
  - Missing Scripting
- Using Rake?
  - Buildsystem for Ruby, now part of standard Ruby Libs
  - Usage of Ruby code blocks
  - Knows not enough about Java
  - Put Buildr on top of Rake

Ease of use

- Simplest project definition:

```
define 'gear-app'
```

- A bit more complicated

```
define 'gear-app' do
  project.version = '0.1.0'
  package :jar
end
```

- And now: do it!

```
$> buildr compile
```

Installation

- You have to install Ruby and some other stuff

```
$ sudo apt-get install ruby-full ruby1.8-dev
      libopenssl-ruby rubygems
```

- And then install Buildr gem:

```
$ sudo gem install buildr
```

- For Windows users:
  - One click install for Ruby
  - Install buildr gem
- JRuby supported, too

Ruby

- Buildr is Rake is Ruby based
- Ruby used as scripting language to extend Buildr
- Ruby
  - Lightweight
  - Easy syntax
  - Easy file manipulation
  - Native regular expressions
  - Already used in many infrastructure projects like Chef

Basic Buildr default tasks

- artifacts
- build
- clean
- compile
- install
- package
- release
- ...and many more:

```
$ buildr -T
```

File Structure

- File Structure is based on Apache defaults
  - Known as Maven project structure
- Can be overridden by own layouts

```
my_layout = Layout.new

my_layout[:source, :main, :java] = 'java'


define 'gear-app', :layout=>my_layout
```

Multimodule support

- Multimodule support is out of the box

```
define 'gear-proj' do
   define 'gear-api' do
     package :jar
   end
   define 'gear-backend' do
     compile.with projects('gear-api')
     package :jar
   end
   package(:jar).using :libs =>
                    projects('gear-api','gear-backend')
end
```

Dependency management

- Built in dependency management
- Using Maven Repositorys

```
repositories.remote << 'http://repo1.maven.org/maven2'
LOG4J = 'log4j:log4j:jar:1.2.15'
JUNIT = 'junit:junit:jar:4.7'

define 'gear-app' do
  project.version = '0.1.0'
  Compile.with LOG4J,JUNIT
  package :jar
end
```

More on dependency management

- Transitive dependency's supported

```
Compile.with transitive(SPRING_WEBMVC)
```

- But doesn't support excludes and conflict handling
  - Try using Log4j w/o exclusions...
- Many Artifact POMs in awful quality → this is the real drawback
- How to handle?
  - Take your dependency's under full control
  - Or use Ivy (via Ivy extension)

Last words about DM

- You can work w/o an repository manager
- Just pick JAR's from a directory

```
OWN = Dir[File.join(OWN_HOME, '*.jar')]
```

- Or download the artifacts

```
url = "http://download.mylib.org/mylib-1.0.0.zip"
download(artifact("hgu:mylib:jar:1.0.0")=>url)
```

- Prepare for offline mode

```
$ buildr artifacts
```

Library's

- Library scopes under full control
- Define yourself which libraries are packaged
  - By default all which are used to compile
- Include / exclude Libraries

```
package(:war).libs -= artifacts(JAVAX.servletapi)

package(:war).libs += artifacts(JETTYLIBS)
```

Testing

- By default, JUnit is assumed to be used as test framework
- Dependency's same as compile plus testframework plus JMock
- Using TestNG instead of JUnit

```
test.using :testng
```

- Skipping tests

```
$ buildr test=no
```

- Excluding tests depending on environment

```
test.exclude '*gui**' unless $stdout.isatty
```

Profies and Environments

- Profiles supported
  - Configuration by Yaml files
  - Selected by current environment

```
$ buildr -e dev01

dev01:
  db: hsql
  jdbc: hsqldb:mem:devdb

test:
  db: oracle
  jdbc: oracle:thin:@localhost:1521:test
```

Other Languages...

- Scala

```
require 'buildr/scala'

scala.version: 2.9.2
```

- Supports Scalatest and Specs2
  - Specs2 is default

```
test.using(:scalatest)
```

Other Languages

- Groovy support as like Scala

```
require 'buildr/groovy'
```

- Testing
  - EasyB supported
- Ruby
  - May you'll stick with Rake
  - Good support for JRuby, tight integration
  - Support for different testing frameworks

Languages

- Buildr supports polyglot projects out of the box

```
define "groovymodule" do
  compile.using(:groovyc)
  package(:jar)
end

define "javamodule" do
  package(:jar)
end

define "scalamodule" do
  compile.using(:scalac)
  package(:jar)
end
```

Extending

- Extending by writing Rake tasks
- Extending in Ruby using Ruby Modules
  - Incorporate 'Extension'
  - Callback's: `first_time`, `before_define`, `after_define`

```
module MyExtension
 include Extension

 first_time do
  desc 'Do my stuff'
  # do something
 end
...
end
```

Wrap up

- Apache Buildr is mature
- The dependency management is a bit different as in Maven
- IDE integration is limited
- Documentation is limited, you need to look into the code
  - But a good active community
- Lets you “code” your build like Gradle and SBT
- Full extensibility using Ruby as scripting language

Wir suchen Sie als

- Software-Architekt (m/w)
- Projektleiter (m/w)
- Senior Software Engineer (m/w)

jobs@adesso.de
www.AAAjobs.de

adesso business. people. technology.

Vielen Dank für Ihre Aufmerksamkeit.

info@adesso.de
www.adesso.de