3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Apache Buildr

Die Maven-Alternative

Tammo van Lessen

innoQ Deutschland GmbH

About me: Tammo van Lessen

innoQ

APACHE ODE

BPMN

innoQ

The talk today

What?

Why?

- Motivation & History of Buildr

How?

- The Architecture
- The Principles
- The Basics

Now What?

innoQ

Apache Buildr

- Ruby-based Build Tool for Java
  - „~~The build tool that does not suck~~“
  - „Build like you code“
- First Ruby Project @ ASF
- Incubated: 2007-11-01
- Top Level Project since 2008-12
- Most Recent Version: Buildr 1.4.7 (2012-05-29)

innoQ

Apache ODE

BPEL-based workflow engine,
enterprisey,
complex build

35+ modules
9 databases
120+ dependencies
3 distributions
heavy tooling

Maven2

6739 lines of XML
in 53 files.

innoQ


LOVE
HATE

What‘s good with Maven?

- Maven Central!
- Provides standard ways to build arbitrary projects
- Broad acceptance
- …

What‘s wrong with Maven?

- Transitive Dependency Resolution?
- Reproducible?
- „Maven Uncertainty Principle“
  - With POMs
  - With Artifacts
  - With Plugins
- No Scriptabilitity
  - 34 lines of XML to merge to SQL files?
- Extensibility is a mess!
- …

Our requirements

- Best of Maven
- Native support for scripting
- No XML
- Flexible
- Extensible
- Smart

innoQ

Buildr

912 lines in 3 files

Buildr

And twice as fast!

Architecture

Buildr → **Rake** → **Ruby**

projects, lifecycle, artifacts, plugins

tasks, files, dependencies

arbitrary scripting

innoQ

Why Ruby?

- Awesome scripting language
- Easy file manipulation
- Native support for regular expressions
- Lightweight syntax
- Easy collection processing
- DSL friendly
- JVM friendly
- …

innoQ

Why Rake?

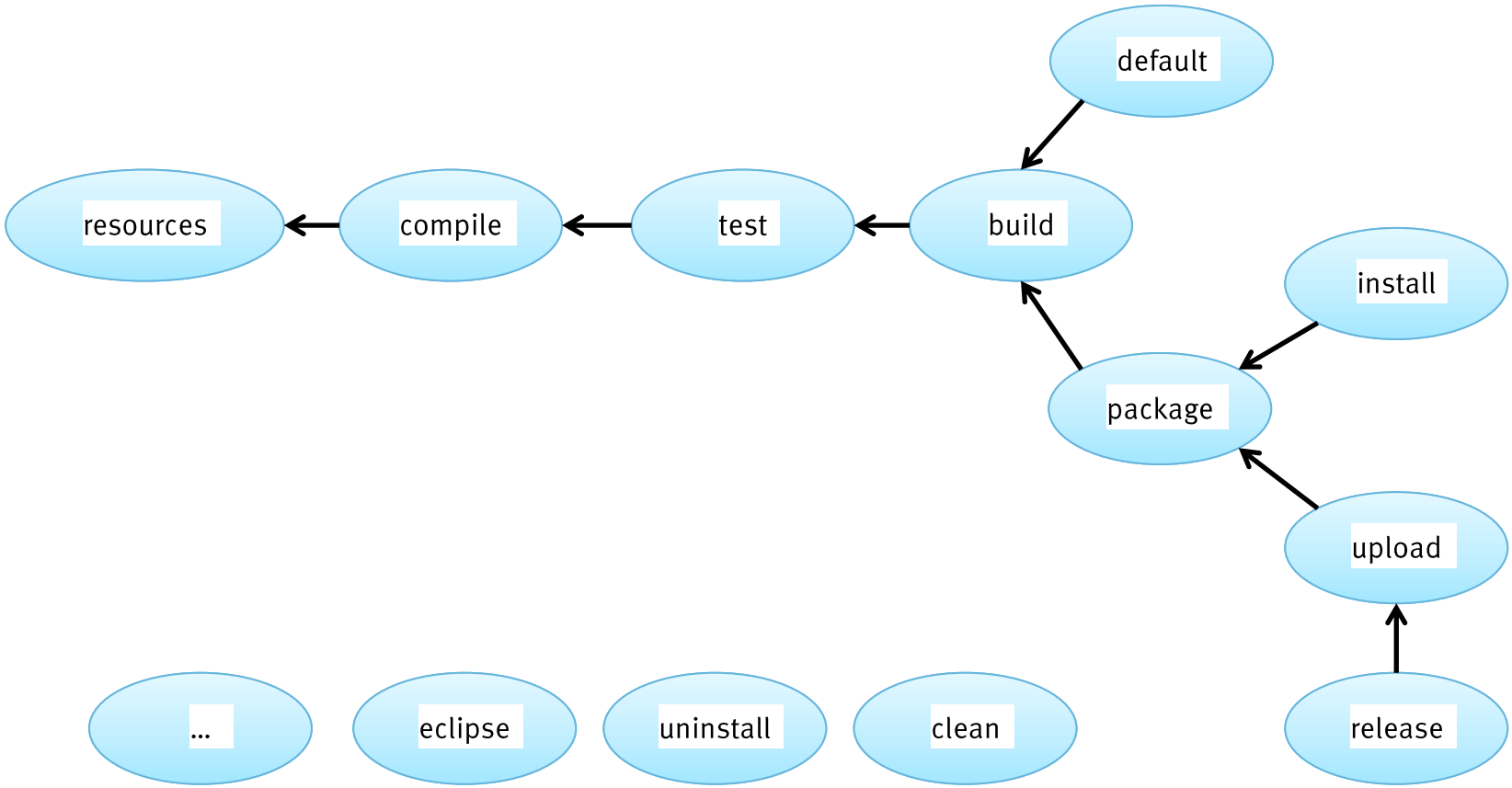
- „The Ruby Make“
- Inherently dependency driven
- Introduces various task types

Demo time!

File Task

```
class FileTask < Task
   # Is this file task needed? Yes if it doesn't exist, or if its time stamp
   # is out of date.
   def needed?
     return true unless File.exist?(name)
     return true if out_of_date?(timestamp)
     false
   end
   # Time stamp for file task.
   def timestamp
     if File.exist?(name)
       File.mtime(name.to_s)
     else
       Rake::EARLY
     end
   end
   private
   # Are there any prerequisites with a later time than the given time stamp?
   def out_of_date?(stamp)
     @prerequisites.any? { |n| application[n].timestamp > stamp}
   end
end
```

It‘s all about dependencies



Artifact Dependencies

- Native support for Maven repositories
- Support for local jars

- Transitive dependencies via Ivy and Aether (plugins)

innoQ

Specifying Artifacts

```
AXIS_VERSION = '1.2'
compile.with "org.apache.axis2:axis2:jar:#{AXIS_VERSION}"


AXIS2 = 'org.apache.axis2:axis2:jar:1.2'
compile.with AXIS2


AXIOM = group('axiom-api', 'axiom-impl', 'axiom-dom',
:under=>'org.apache.ws.commons.axiom', :version=>'1.2.4')


OPENJPA = ['org.apache.openjpa:openjpa:jar:1.2.1',
'net.sourceforge.serp:serp:jar:1.12.0']


JAVAX = struct( :activation
=>'javax.activation:activation:jar:1.1', :persistence
=>'javax.persistence:persistence-api:jar:1.0', :stream
=>'stax:stax-api:jar:1.0.1', )
```

Building: Compiling & Testing

- Polyglot
  - Java. Scala, Groovy, Ruby, Closure, roll your own
- Hooks for code generation
- Resources / Filters
- Tests
  - JUnit, TestNG
- Continuous Compilation

innoQ

Packaging

- ZIP
- JAR
- WAR
- AAR
- EAR
- OSGi Bundles
- TAR & TAR.GZ
- Javadocs
- Sources

innoQ

Integrating Ant

```
<taskdef name="xmlbean"
         classname="org.apache.xmlbeans.impl.tool.XMLBean"
         classpath="path/to/xbean.jar" />

<xmlbean classgendir="${build.dir}"
         classpath="${class.path}"
         failonerror="true">
    <fileset basedir="src" excludes="**/*.xsd"/>
    <fileset basedir="schemas" includes="**/*.*"/>
</xmlbean>
```

Integrating Ant (2)

```
def xmlbeans(files) do
    Buildr.ant "xmlbeans" do |ant|
        ant.taskdef \
            :name => "xmlbeans",
            :classname => "org.apache.xmlbeans.impl.tool.XMLBean",
            :classpath => 'org.apache.xmlbeans:xmlbeans:jar:2.3.0'
        ant.xmlbeans \
            :classpath => project.compile.dependencies,
            :srcgendir => project._('target/generated')
            :failonerror => "true" do
                files.flatten.each do |file|
                    ant.fileset File.directory?(file) ? { :dir => file }
                                                      : { :file => file }
            end
    end
end
```

Releasing

1. Check that the version to be released and the next version are different
2. Check that the project is being tracked by Git, SVN or HG
3. Package, test and deploy the artifacts using THIS_VERSION value minus the -SNAPSHOT suffix (if any)
4. Tag the repository with the released version number
5. Update the value of THIS_VERSION in the buildfile with the next version number

Demo Time!

Some more examples

```
# Generate SQL DDL schemas for all databases
%w{ derby mysql oracle sqlserver postgres }.each do |db|
    db_xml = _("src/main/descriptors/persistence.#{db}.xml")
    partial_sql = file("target/partial.#{db}.sql"=>db_xml) do
        OpenJPA.mapping_tool \
            :properties => db_xml,
            :action => "build",
            :sql => db.to_s,
            :classpath => projects("store", "dao")
    end
    # Add Apache license header
    header = _("src/main/scripts/license-header.sql")
    sql = concat(_("target/#{db}.sql") => [header, partial_sql])
    build sql
end
```

Some more examples (2)

```
# Compile using all Eclipse BIRT libraries
BIRT_WAR = artifact(“org.eclipse.birt:birt:war:1.4.1”)

unzip_birt = (unzip _("target/birt") => BIRT_WAR).tap do |t|
    compile.with Dir[_("target/birt/WEB-INF/lib") + "/*.jar"]
end

compile.enhance [unzip_birt]
```

Some more examples (3)

```
Java.classpath << [ "org.antlr:antlr:jar:3.0", "antlr:antlr:jar:2.7.7",
"org.antlr:stringtemplate:jar:3.0" ]

Java.org.antlr.Tool.new("-i #{input} -o #{output}").process
```

Example Extension

```
module GitVersion
    include Extension

    @version = `git log -1 --pretty=format:%H`

    after_define do |project|
        project.packages.each do |jar|
            f = file project._("target/git-version.txt") do |f|
              Buildr.write f.to_s, @version
            end
            jar.enhance [f]
            jar.include f, :as => "META-INF/git-version.txt"
        end
    end
end
```

innoQ

Example Extension (2)

```
# apply extension to a single project
define 'my-project' do
    extend GitVersion
end

# apply extension to all projects
class Buildr::Project
    include GitVersion
end
```

And more...

Layouts, Profiles, Environments, Code Coverage,
Notifications, Eclipse, IntelliJ, GPG,
CheckStyle, FindBugs, JDepend, Sonar, ...

Libs

Antlr, BND, GWT, Hibernate, OpenIPA, Jetty,
XMLBeans, ProtoBuf, ...

Plugins

Android, OSGi, JavaScript, UnitTesting, AspectJ,
Flex, ...

Best practices

Monkey see, Monkey do!

Beware of the DSL

Ask for support!

buildr.apache.org

Enjoy faster builds!

innoQ

Build like you code!

innoQ

Vielen Dank!

Tammo van Lessen
innoQ Deutschland GmbH