Apache Buildr

Build like you code

About Me

Alex Boisvert

French Canadian

Live in USA

Java

Scala

Ruby

Groovy

Buildr Committer

[PMC Chair]

bizo

Use Buildr

on daily basis

Contributions

scala support
plugins & extensibility
eclipse integration
many other improvements
and bug fixes

Practical

Introduction to Apache Buildr

Build system

that doesn't suck.

Where

it all started

Apache Ode

Large Java Entreprise Middleware

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

...

Maven2

5,433 lines of XML spread over 52 files (!)

“@&#! There's Got To Be A Better Way”

What We Really Wanted

No XML.

Please!

Flexible

easy to customize & extend

DRY Code

basic abstraction and structuring mechanisms

In other words,

a <u>real</u> scripting language.

Evolutionary

support existing conventions and practices

Result?

Before

52 files
5,433 lines of XML.

After

Single file.
486 lines of Ruby.

Bonus!

Twice as fast.

How did

we do it?

Buildr	projects, lifecycle, artifacts, plugins
↓	
Rake	tasks, files, dependencies
↓	
Ruby	awesome scripting language

why ruby?

Scripting

easy file manipulation
native regexp,
lightweight syntax
exec() friendly
etc.

Expressive

great host for embedded domain-specific language

JVM Friendly ;)

JRuby &
Ruby-Java Bridge

Rake

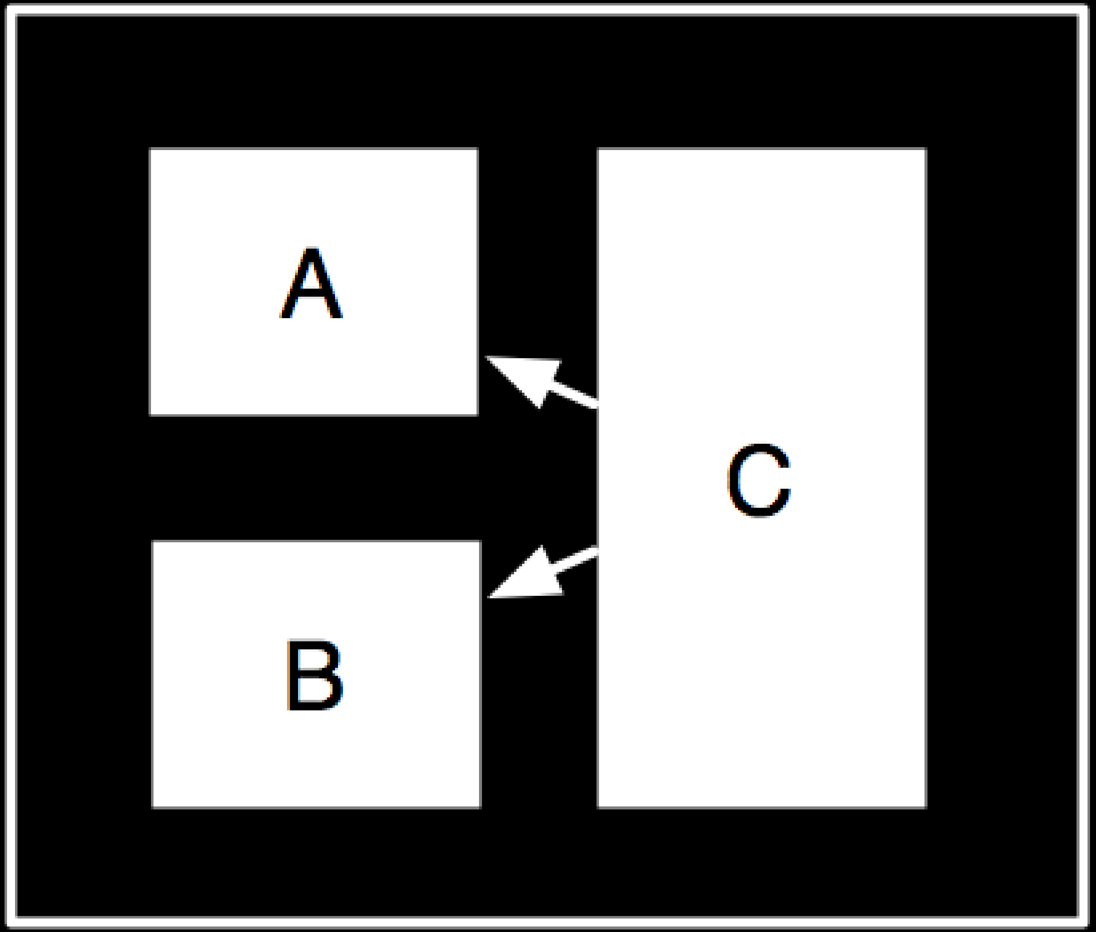
The Ruby Make

Rake

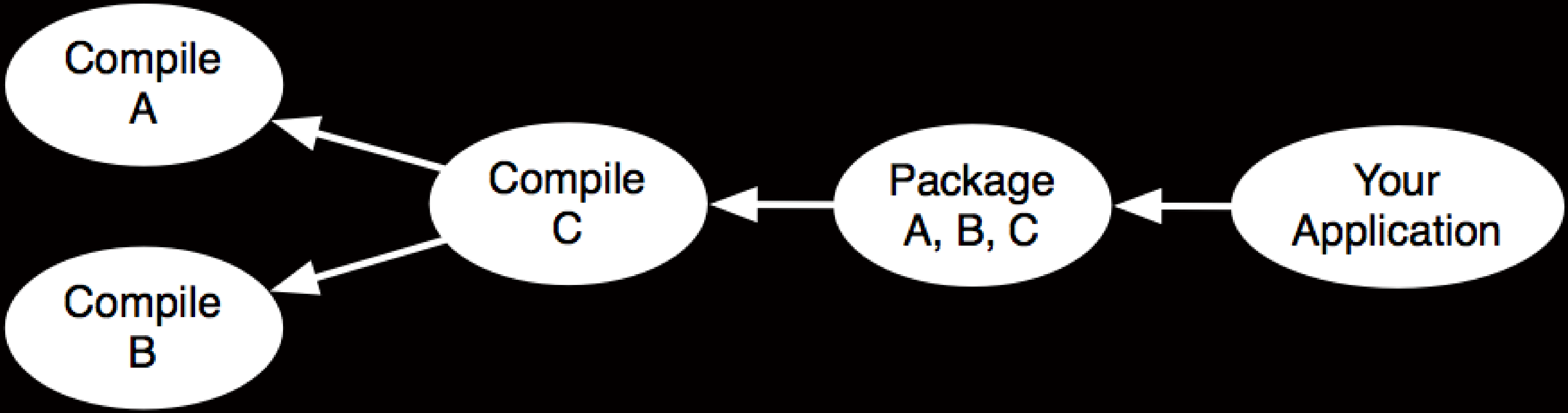
The Ruby ~~Make~~

“Ant”

Your Application



modules



graph of dependencies

```
# This is Rake code
task "compile A" do
  # code to compile A
End

task "compile B" do
  # code to compile B
End

task "compile C" => ["compile A", "compile B"] do
  # code to compile C
End

task "package A,B,C" => ["compile A", "...", "compile C"] do
  # code to package A, B and C
end

task :default => “package A, B, C”
```

```
$ rake
(in /home/buildr/example)
compiling A ...
compiling B ...
compiling C ...
packaging A,B,C ...
```

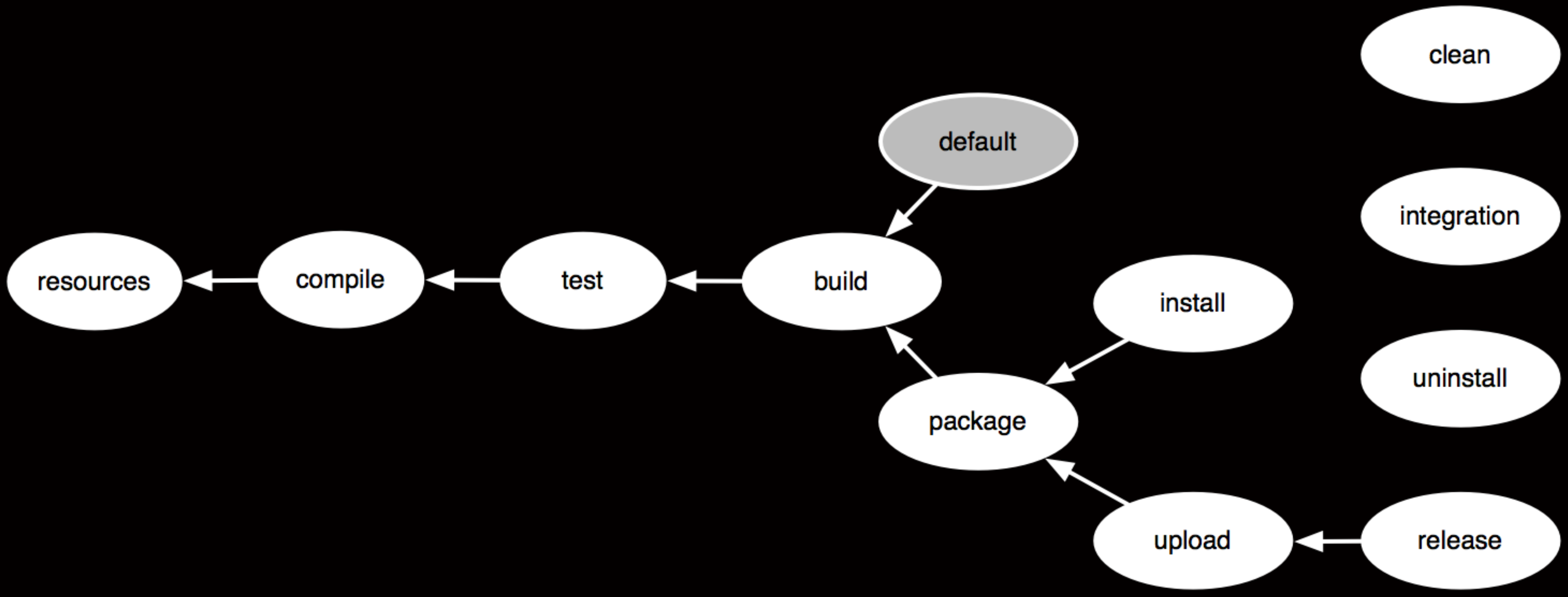
```
# This is Buildr code

define "My application" do
  define "A" do
    package :jar
  end

  define "B" do
    package :jar
  end

  define "C" do
    compile.with projects("A", "B")
    package :jar
  end

  package(:war).using :libs => projects("A", "B", "C")
end
```

```
$ buildr package
(in /home/boisvert/tmp/buildr-example, development)
Building buildr-example
Compiling buildr-example:a into
/home/boisvert/tmp/buildr-
example/a/target/classes
Compiling buildr-example:b into
/home/boisvert/tmp/buildr-
example/b/target/classes
Packaging buildr-example-a-1.0.0.jar
Packaging buildr-example-b-1.0.0.jar
Compiling buildr-example:c into
/home/boisvert/tmp/buildr-
example/c/target/classes
Packaging buildr-example
Packaging buildr-example-c-1.0.0.jar
Packaging buildr-example-1.0.0.war
Running integration tests...
Completed in 0.779s
```

All about

tasks.

Standard tasks



artifacts

and repositories

```ruby
# Buildfile

repositories.remote << "http://www.ibiblio.org/maven2/"

LOG4J = "log4j:log4j:jar:1.2.15"

define 'my-library' do
  manifest['Copyright'] = 'Acme Inc (C) 2008'
  compile.options.target = '1.5'
  compile.with LOG4J
  package :jar
end
```

all your repos
are belong
to us!

```
# Buildfile

repositories.remote << "http://www.ibiblio.org/maven2/"

LOG4J = "log4j:log4j:jar:1.2.15"

define 'my-library' do
  manifest['Copyright'] = 'Acme Inc (C) 2008'
  compile.options.target = '1.5'
  compile.with LOG4J
  package :jar
end
```

artifacts are tasks, too.

```
# Buildfile

repositories.remote << "http://www.ibiblio.org/maven2/"

LOG4J = "log4j:log4j:jar:1.2.15"

define 'my-library' do
  manifest['Copyright'] = 'Acme Inc (C) 2008'
  compile.options.target = '1.5'
  compile.with LOG4J
  package :jar
end
```

Languages

we got 'em

Example: Scala plugin

```
require 'buildr/scala'

# brings in:
#
#   - automatic detection
#     (src/main/scala, src/test/scala, src/spec/scala)
#
#   - incremental compilation
#
#   - mixed java + scala compilation
#
#   - scaladoc generation
#
#   - scalatest, specs and scalacheck testing
#
#   - continuous compilation (experimental)
```

Demo Time!

Java + Scala + Groovy Mixed Project

Ant Example: XMLBeans

```
<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>
```

Ant Example: XMLBeans

```
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
```

```
# 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

# dependency chain:
#
# db_xml (x5) ← partial_sql (x5) ←| sql (x5) ← build
#                 header (x1)    ←|
# 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]


# dependency chain:
#
# BIRT_WAR ← unzip (and compile.with) ← compile
```

Calling Java classes

```
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
```

Testing Your Build

```
check  package(:war).entry('META-INF/MANIFEST'), 'should
have license' do
  it.should contain(/Copyright (C) 2007/)
end

check file('target/classes/killerapp/Code.class'), 'should
exist' do
  it.should exist
end
```

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
```

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


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

More Stuff

layouts, profiles,
code coverage, notifications
more plugins, more languages
+ more awesome.

Only one thing
to remember.

Build like you code.

join us!

http://buildr.apache.org

alex.boisvert@gmail.com
twitter: boia01