Guillaume Laforge

Jfokus 2011

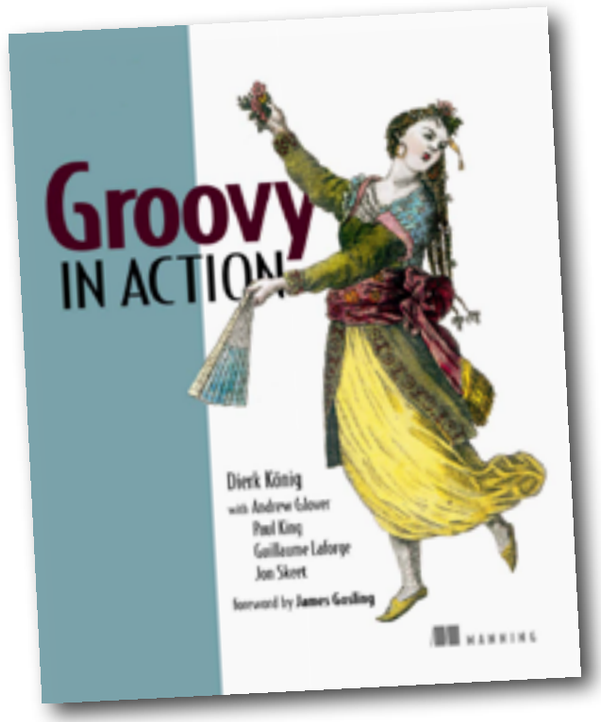
Groovy

a very **rich** and **lively** **ecosystem**

Guillaume Laforge / @glaforge

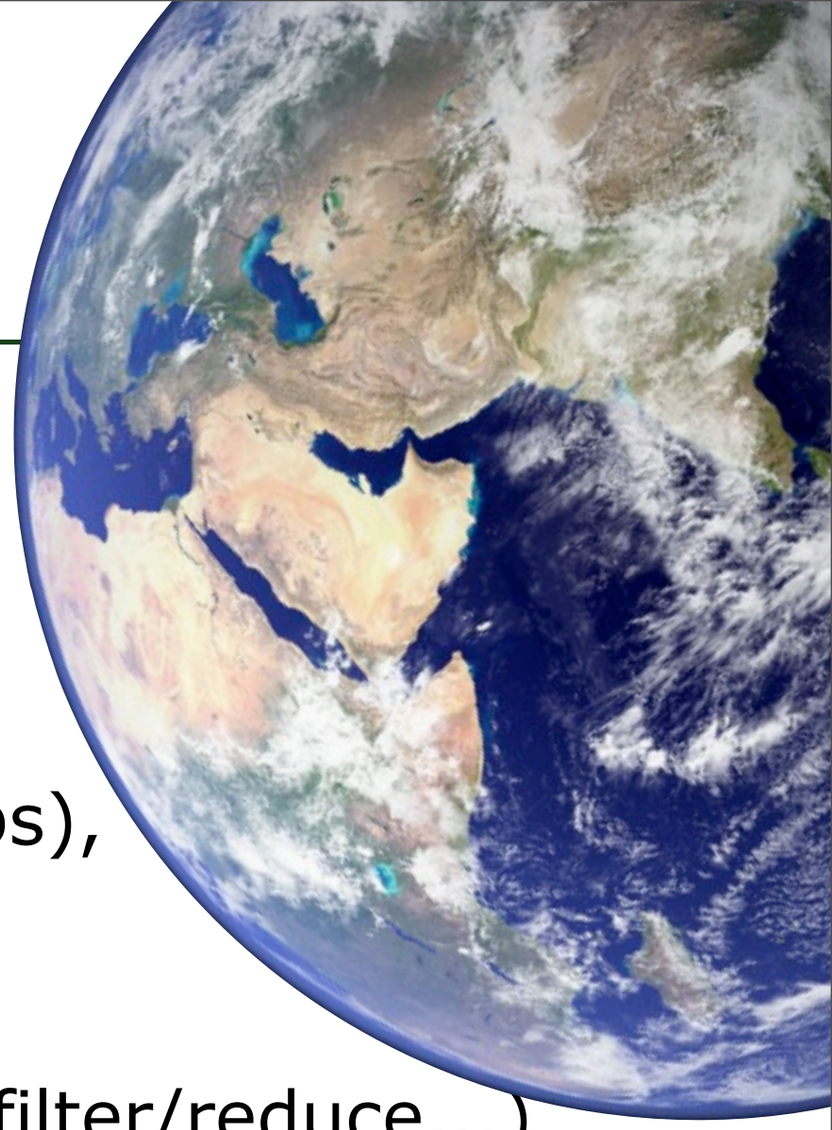
springsource
A division of vmware

- **Groovy Project Manager**
- Head of Groovy Development at **SpringSource**
- Initiator of the **Grails** framework
- Founder of the **Gaelyk** toolkit
- Co-author of **Groovy in Action**
- Member of «**Les Cast Codeurs**» podcast



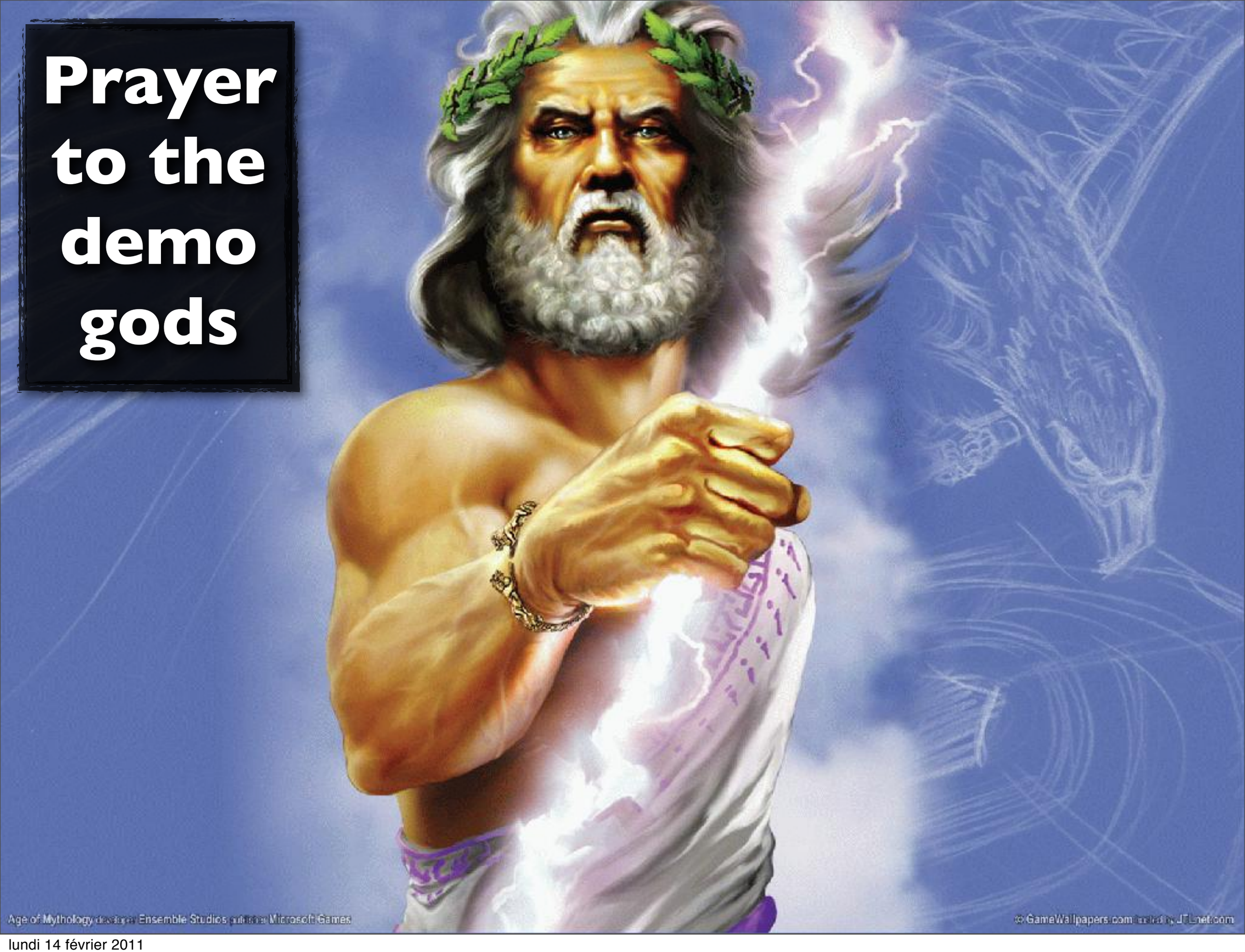
- Speaker: JavaOne, QCon, JavaZone, Sun TechDays, Devoxx, The Spring Experience, SpringOne2GX, JAX, Dynamic Language World, IJTC, and more...

The Groovy Ecosystem

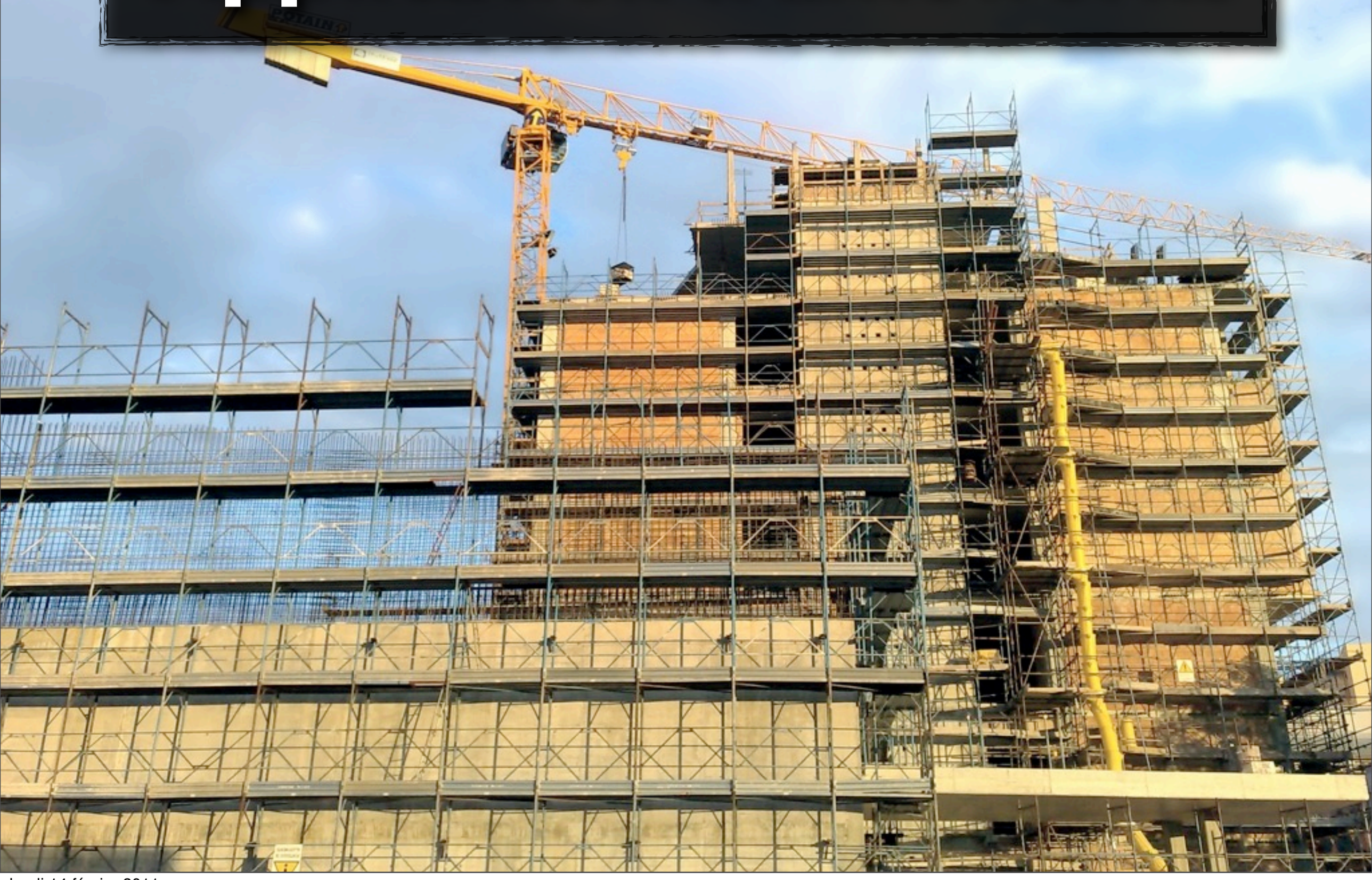


- Many projects based on Groovy!
- Serve various purposes:
  - build applications (ie. frameworks)
    - **Grails** (web apps), **Griffon** (Swing apps), **Gaelyk** (Google App Engine)
  - tame new concurrency paradigms
    - **GPars** (actors, agents, fork/join, map/filter/reduce...)
  - enhanced testing
    - **Easyb** (BDD), **Spock** (BDD&mock), **Gmock** (mocking)
  - help building projects
    - **Gant** (ant sugar), **Gradle** (adhoc build system)
  - web services interaction
    - **HTTPBuilder** (REST), **GroovyWS** (SOAP)

Prayer to the demo gods



Application frameworks



Grails (1/4)


springsource
A division of vmware
GRAILS
Products
Services
Training
Case Studies
Community
Downloads
Documentation
GRAILS
the search is over.
RAPID
Have your next Web 2.0 project done in weeks instead of months. Grails delivers a new age of Java web application productivity.
DYNAMIC
Get instant feedback, see instant results. Grails is the premier dynamic language web framework for the JVM.
ROBUST
Powered by Spring, Grails out performs the competition. Dynamic, agile web development without compromises.

Grails (2/4)

springsource
A division of vmware

- Built-on **proven** and **rock-solid OSS** technologies

- **Spring**
  - IoC/DI, MVC, WebFlow
- **Hibernate**
  - ORM mapping layer
  - H2: embedded Java database
- **Tomcat**
  - Local servlet container
- **Groovy**
  - To glue everything together!
- SiteMesh, Quartz...

Grails (3/4)

- Grails is an **MVC** web application stack
  - Domain classes mapped to datastores (RDBMS, NoSQL)
    - Transparent mapping
  - Controllers and service classes in Groovy
  - GSP (JSP-like template engine), with powerful taglibs

- **Smart reloading** as you make changes live

- Comes with its own command-line interface
  - Creating artifacts, building/testing the project, etc.

Grails (4/4)

- **Hundreds of plugins** available (559!)
  - Which let you expand your application easily
  - Providing functionality you won't have to implement
  - You may hear about '*Plugin Oriented Architecture*'

- Some of the most used ones
  - Searchable: add Google-like **search** interface
  - Mail: easily **send emails**
  - Quartz: for your **batch jobs**
  - GraniteDS / Flex Scaffold: use a **Flex UI frontend**
  - Spring Security: **authentication** / **authorization**
  - and 554 other ones :-)

Griffon (1/4)

GRIFFON

springsource
A division of vmware

- Griffon is a Grails-like application framework for developing **rich desktop applications**
  - MVC, DRY, CoC patterns
  - Easy **threading**, **data binding** and **observing**
  - Support in IDEs
  - Lost of **plugins** too!

Griffon (2/4)

- Let’s create a simple demo console for executing Groovy scripts

- A data model

```
import groovy.beans.Bindable

class DemoConsoleModel {
    String scriptSource
    @Bindable scriptResult
    @Bindable boolean enabled = true
}
```

Griffon (3/4)

springsource
A division of vmware

- A controller

```
class DemoConsoleController {
    GroovyShell shell = new GroovyShell()

    // automatically injected by Griffon
    def model, view

    def executeScript(evt = null) {
        model.enabled = false
        doOutside {
            def result = shell.evaluate(model.scriptSource)
            edt {
                model.enabled = true
                model.scriptResult = result
            }
        }
    }
}
```

Griffon (4/4)

springsource
A division of vmware

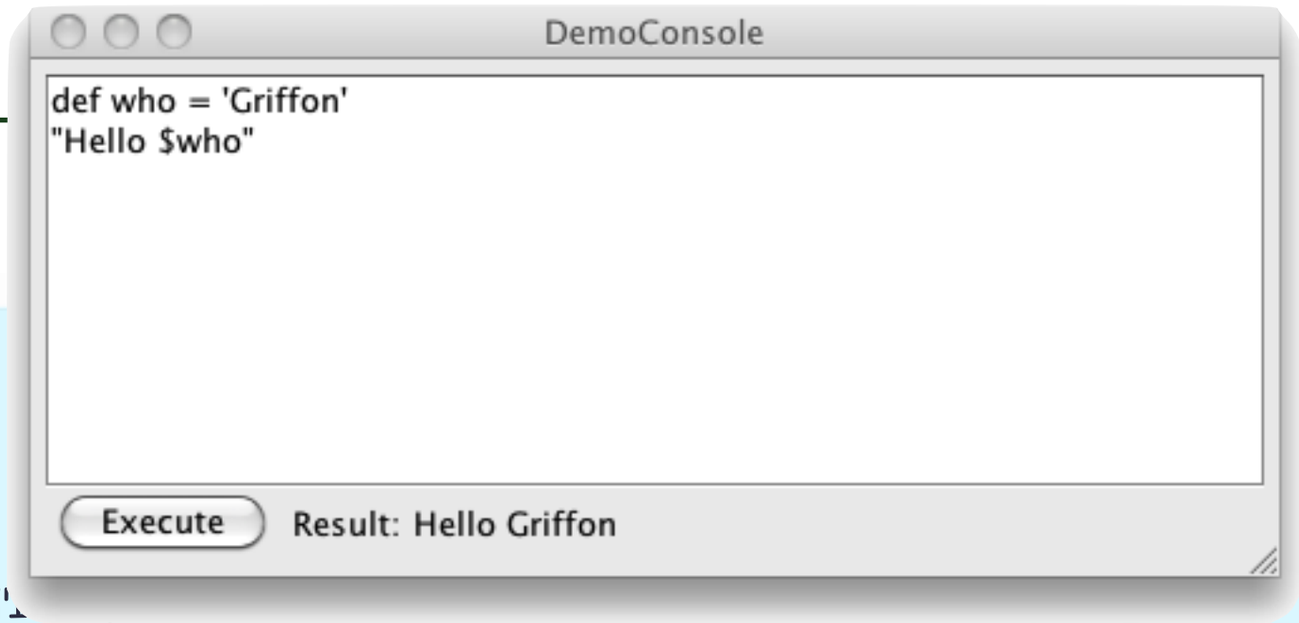
- And a view

```
application(title: 'DemoConsole', pack: true, locationByPlatform: true) {
  panel(border: emptyBorder(6)) {
    borderLayout()

    scrollPane(constraints: CENTER) {
        textArea(text: bind(target: model, targetProperty: 'scriptSource'),
                 enabled: bind { model.enabled },
                 columns: 40, rows: 10)
    }

    hbox(constraints: SOUTH) {
        button("Execute", actionPerformed: controller.&executeScript,
               enabled: bind { model.enabled })
        hstrut 5
        label "Result:"
        hstrut 5
        label text: bind { model.scriptResult }
    }
  }
}
```

Griffon (4/4)



- And a view

```
application(title: 'DemoConsole                                    {
  panel(border: emptyBorder(6))
    borderLayout()

    scrollPane(constraints: CEN                                    ),
        textArea(text: bind(target: model, targetProperty: 'scriptSource'),
                 enabled: bind { model.enabled },
                 columns: 40, rows: 10)
    }

    hbox(constraints: SOUTH) {
        button("Execute", actionPerformed: controller.&executeScript,
               enabled: bind { model.enabled })
        hstrut 5
        label "Result:"
        hstrut 5
        label text: bind { model.scriptResult }
    }
  }
}
```

Gaelyk

springsource
A division of vmware

- **Lightweight Groovy toolkit** for building and deploying applications on **Google App Engine**

  - On top of the **Groovy servlet**
    - executing scripts as mere servlets
    - for the controllers

  - On top of the **Groovy template engine**
    - JSP-flavored templates
    - for the views

  - Plus lots of **sugaring** of GAE SDK APIs

Gaelyk

springsource
A division of vmware

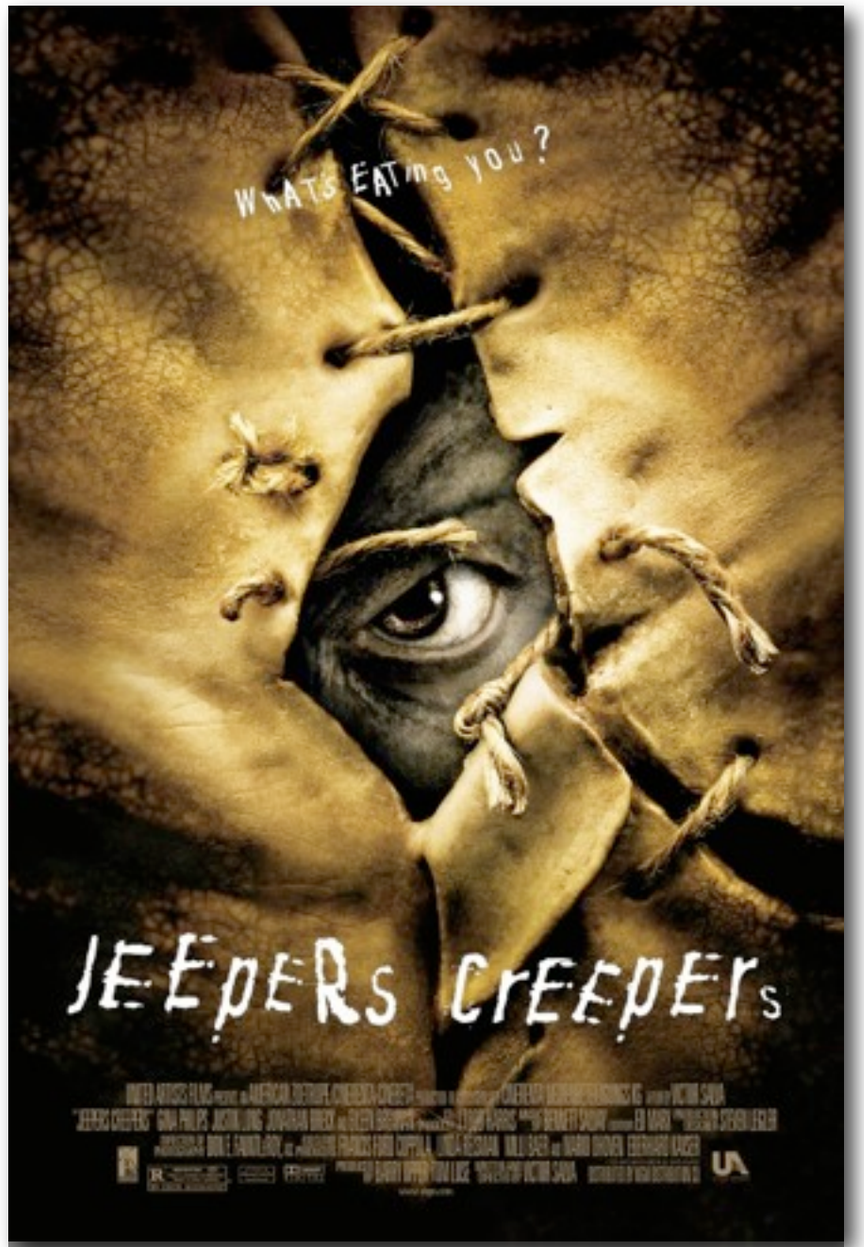
- Don’t miss my Gaelyk presentation!
  - **Tuesday, 13:00-13:50, Room C1**

Taming concurrency

GPars (1/6)

springsource
A division of vmware

- Concurrency library
- Multiple high-level abstractions
  - actors
  - map / filter / reduce
  - fork / join
  - asynchronous closures
  - agents
  - dataflow concurrency
- Can be used with Java!



GPars (2/6)

springsource
A division of vmware

- Improving over JSR-166y's **ParallelArray**

```
GParsPool.withPool { // using fork/join
    // a map-reduce functional style
    def smallestSelfPortrait =
        images.parallel.filter { it.contains me }
                       .map    { it.resize() }
                       .min    { it.sizeInMB }
}
```

- Asynchronous closures returning **Futures**

```
GParsExecutorsPool.withPool { // using JDK executors
    // adds callAsync() to closures
    assert 6 == { it * 2 }.callAsync(3).get()
}
```

GPars (3/6)

springsource
A division of vmware

- Improving over JSR-166y's **Fork / Join**
  - Think devide and conquer problems

```
withPool(2) { pool ->
    println "Number of files: " +
        runForkJoin(new File("./src")) { File file ->
            long count = 0
            file.eachFile {
                if (it.isDirectory()) {
                    println "Forking a child task for $it"
                    // fork a child task
                    forkOffChild(it)
                } else {
                    count++
                }
            }
            // use results of children tasks
            // to calculate and store own result
            return count + childrenResults.sum(0)
        }
}
```

GPars (4/6)

- **Actors**

```
loop {
    println 'Waiting for a gift'
    react { gift ->
        if (myWife.likes(gift)) {
            reply 'Thank you!'
        } else {
            reply 'Try again, please'
            react { anotherGift ->
                if (myChildren.like(anotherGift))
                    reply 'Thank you!'
            }
        }
    }
}
```

  - (see also dynamic dispatch actors, and more)

GPars (5/6)

- **DataFlow** concurrency

```
new DataFlows().with {
    task {
        z = x + y
        println "Result: ${z.val}"
    }

    task {
        x = 10
    }

    task {
        y = 5
    }
}
```

GPars (6/6)

- **Agents**
  - thread-safe
    non-blocking
    shared
    mutable
    state
    wrappers

```
class Conference extends Agent<Integer> {
    def Conference() { super(0) }
    private register(long num)   { data += num }
    private unregister(long num) { data -= num }
}

final Agent conference = new Conference()

// send commands in parallel threads
final Thread t1 = Thread.start {
    conference << { register 10 }
}
final Thread t2 = Thread.start {
    conference << { register 5 }
}
final Thread t3 = Thread.start {
    conference << { unregister 3 }
}

[t1, t2, t3]*.join()
assert 12 == conference.val
```

Testing

Easyb (1/4)

springsource
A division of vmware

- From the website:
  - *«Easyb is a **behavior driven development** (BDD) framework for the Java platform. By using a specification based Domain-Specific Language, Easyb aims to enable executable, yet readable documentation»*

- Write specifications in Groovy
- BDD == given / when / then paradigm
- Run them from CLI, Maven, or Ant
  - provides various report formats

Easyb (2/4)

- First, let’s write a story

```
given "an invalid zip code"

and "given the zipcodevalidator is initialized"

when "validate is invoked with the invalid zip code"

then "the validator instance should return false"
```

Easyb (3/4)

- Let’s write the test itself

```
given "an invalid zip code", {
    invalidzipcode = "221o1"
}

and "given the zipcodevalidator is initialized", {
    zipvalidate = new ZipCodeValidator()
}

when "validate is invoked with the invalid zip code", {
    value = zipvalidate.validate(invalidzipcode)
}

then "the validator instance should return false", {
    value.shouldBe false
}
```

- Then write code that makes the test pass

Easyb (4/4)

- There’s also the specification format

```
before "initialize zipcodevalidator instance", {
    zipvalidate = new ZipCodeValidator()
}

it "should deny invalid zip codes", {
    ["221o1", "2210", "22010-121o"].each { zip ->
        zipvalidate.validate(zip).is false
    }
}

it "should accept valid zip codes", {
    ["22101", "22100", "22010-1210"].each { zip ->
        zipvalidate.validate(zip).shouldBe true
    }
}
```

Spock (1/3)

- From the horse's mouth:
  - *« Spock is a testing and specification framework for Java and Groovy applications. What makes it stand out from the crowd is its beautiful and highly expressive specification language. »*

```
class HelloSpock extends Specification {
    def "can you figure out what I'm up to?"() {
        expect:
        name.size() == size

        where:
        name      | size
        "Kirk"    | 4
        "Spock"   | 5
        "Scotty"  | 6
    }
}
```

Spock (2/3)

- Extensible (but transparent) use of AST transformations to «pervert» the Groovy language
  - reuse of labels for the BDD actions

- Spock brought Groovy 1.7's enhanced asserts

```
Condition not satisfied:

max(a, b) == c
|   |  |  |  |
3   1  3  |  2
          false
```

Spock (3/3)

- Another example of the expressive language
  - with powerful mocking capabilities

```
def "subscribers receive published events at least once"() {
    when: publisher.send(event)
    then: (1.._) * subscriber.receive(event)
    where: event << ["started", "paused", "stopped"]
}
```

Geb — Browser testing (1/2)

springsource
A division of vmware

```groovy
import geb.*

Browser.drive("http://google.com/ncr") {
  assert title == "Google"

  // enter wikipedia into the search field
  $("input", name: "q").value("wikipedia")

  // wait for the change to results page to happen
  // (google updates the page without a new request)
  waitFor { title.endsWith("Google Search") }

  // is the first link to wikipedia?
  def firstLink = $("li.g", 0).find("a.l")
  assert firstLink.text() == "Wikipedia"

  // click the link
  firstLink.click()

  // wait for Google's javascript to redirect
  // us to Wikipedia
  waitFor { title == "Wikipedia" }
}
```

- Selenium based
- Usages
  - Web testing
  - Screen scraping
  - Process automation
- JQuery-like syntax for matching

Geb — Browser testing (2/2)

- Integrated with
  - Spock
  - EasyB
  - JUnit 3/4
- An example with Spock

```
import geb.spock.GebSpec

class MySpec extends GebSpec {

    def "test something"() {
        when:
        go "/help"
        then:
        $("h1").text() == "Help"
    }

    // base URL to avoid repetition
    String getBaseUrl() {
        "http://myapp.com"
    }
    // chose a specific browser
    WebDriver createDriver() {
        new FirefoxDriver()
    }
}
```

GMock (1/3)

- GMock is a mocking framework for Groovy

```
File mockFile = mock(File, constructor("/a/path/file.txt"))
mockFile.getName().returns("file.txt")
play {
    def file = new File("/a/path/file.txt")
    assertEquals "file.txt", file.getName()
}
```

- Mocking capabilities
  - method calls
  - property access
  - static calls
  - partial mocks
  - constructor calls
  - time matching
  - order matching
  - regex method matching

GMock (2/3)

- Mocking method calls

```
def loader = mock()
loader.put("fruit").returns("apple")

play {
    assertEquals "apple", loader.put("fruit")
}
```

- Mock static calls

```
def mockMath = mock(Math)
mockMath.static.random().returns(0.5)

play {
    assertEquals 0.5, Math.random()
}
```

GMock (3/3)

- Expecting exceptions

```
loader.put("throw exception").raises(RuntimeException, "an exception")
```

- Time matching
  - never, once, atLeastOnce, atMostOnce, stub, times, atLeast, atMost

```
mockLoader.load(2).returns(3).atLeastOnce()
play {
    assertEquals 3, mockLoader.load(2)
    assertEquals 3, mockLoader.load(2)
}
```

Building applications

Groovy's own AntBuilder

springsource
A division of vmware

- Groovy comes with a DSL wrapper around Ant
  - so you can reuse any Ant trask with a Groovy syntax

```
new AntBuilder().sequential {
    mkdir dir: "target/classes"
    taskdef name: "groovyc", classname: "org.codehaus.groovy.ant.Groovyc"
    groovyc srcdir: "src/main", destdir: "target/classes", {
        classpath {
            fileset dir: "lib", {
                include name: "*.jar"
            }
            pathelement path: "target/classes"
        }
        javac source: "1.5", target: "1.5", debug: "on"
    }
    jar basedir: "target/classes", destfile: "jarname.jar", {
        manifest {
            attribute name: "Implementation-Title", value: "My project"
            attribute name: "Implementation-Version", value: "1.2.3"
        }
    }
    echo "BUILD SUCCESSFUL"
}
```

Gant

- Gant is a tool for scripting Ant tasks using Groovy instead of XML to specify the logic

```
includeTargets << gant.targets.Clean
cleanPattern << ['**/*~', '**/*.bak']
cleanDirectory << 'build'

target(stuff: 'A target to do some stuff.') {
    println 'Stuff'
    depends clean
    echo message: 'A default message from Ant.'
    otherStuff()
}

target(otherStuff: 'A target to do some other stuff') {
    println 'OtherStuff'
    echo message: 'Another message from Ant.'
    clean()
}

setDefaultTarget stuff
```

Gradle (1/3)

springsource
A division of vmware

- Gradle is an **enterprise-grade build** system
  - A very flexible general purpose build tool like Ant
  - Switchable, build-by-convention frameworks à la Maven, for Java, Groovy and Scala projects
  - **Groovy build scripts**
  - Powerful **support for multi-project builds**
  - **Dependency management** based on Ivy
  - Full support for existing Maven or Ivy repository infra
  - Support for **transitive dependency management**
  - Ant tasks and builds as first class citizens

Gradle (2/3)

- For example, for a classical Java project

```
apply plugin: 'java'

repositories {
    mavenCentral()
}

dependencies {
    compile group: 'commons-collection',
            module: 'commons-collection', version: '3.2'
    testCompile group: 'junit',
                module: 'junit', version: '4.+'
}
```

Gradle (3/3)

- Major projects migrate to Gradle
  - Hibernate
  - Various SpringSource projects
  - Groovy -based projects
    - Grails, Groovy, GPars, Gaelyk
  - Customers with 200+ multi-projects

Other build and quality tools

springsource
A division of vmware

- You can use Ant, Gant, Maven (GMaven plugin) or Gradle to build your projects
  - fits nicely in any existing build and **continuous integration** infrastructure

- A few Groovy-friendly tools provide handy metrics
  - **CodeNarc**: provides various rules for static analysis of your Groovy codebase *(style, size, braces, naming...)*
  - **GMetrics**: code metrics
  - **Simian**: detects duplication in your Groovy code base
  - **Cobertura**, **Clover**: give code coverage metrics

Quality

Metrics

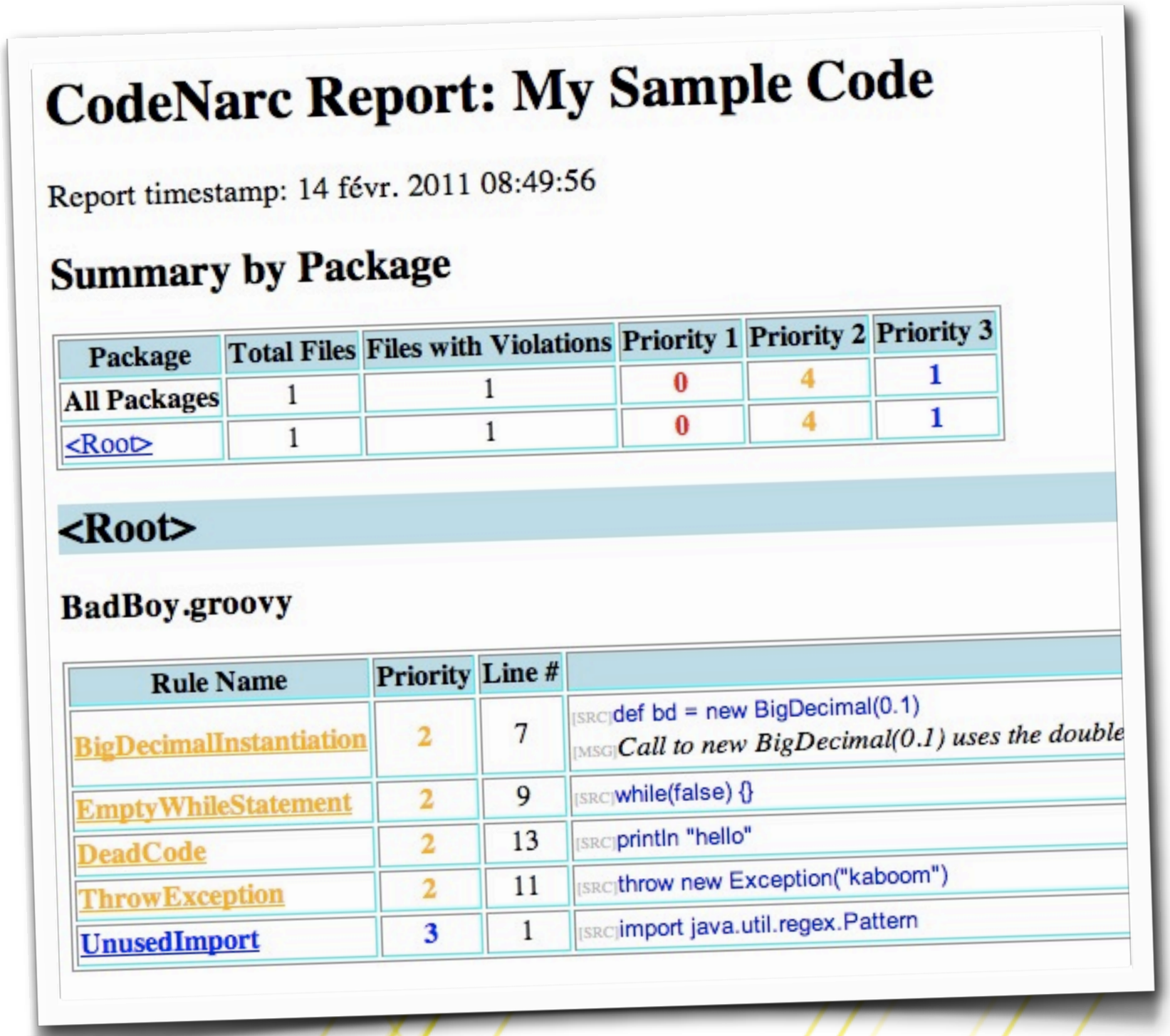
springsource
A division of vmware

- **Static code analysis** for Groovy code source
  - Think **PMD**, **checkstyle**, **FindBugs** for Groovy

- Helps you find:
  - defects, bad practices, inconsistencies, style issues

- Several useful rules:
  - Basic, braces, concurrency, design, DRY, exception handling, generics, imports, JUnit usage, logging, naming conventions, size/complexity, unnecessary/ unused code

CodeNarc (2/2)



CodeNarc Report: My Sample Code

Report timestamp: 14 févr. 2011 08:49:56

Summary by Package

Package	Total Files	Files with Violations	Priority 1	Priority 2	Priority 3
All Packages	1	1	0	4	1
<Root>	1	1	0	4	1

<Root>

BadBoy.groovy

Rule Name	Priority	Line #	
BigDecimalInstantiation	2	7	[SRC]def bd = new BigDecimal(0.1) [MSG]*Call to new BigDecimal(0.1) uses the double*
EmptyWhileStatement	2	9	[SRC]while(false) {}
DeadCode	2	13	[SRC]println "hello"
ThrowException	2	11	[SRC]throw new Exception("kaboom")
UnusedImport	3	1	[SRC]import java.util.regex.Pattern

GMetrics (1/2)

springsource
A division of vmware

- Provides calculations and reporting of **size** and **complexity metrics**

- Calculations
  - Cyclomatic complexity
  - ABC size and complexity
  - Lines per method
  - Lines per class

GMetrics (2/2)

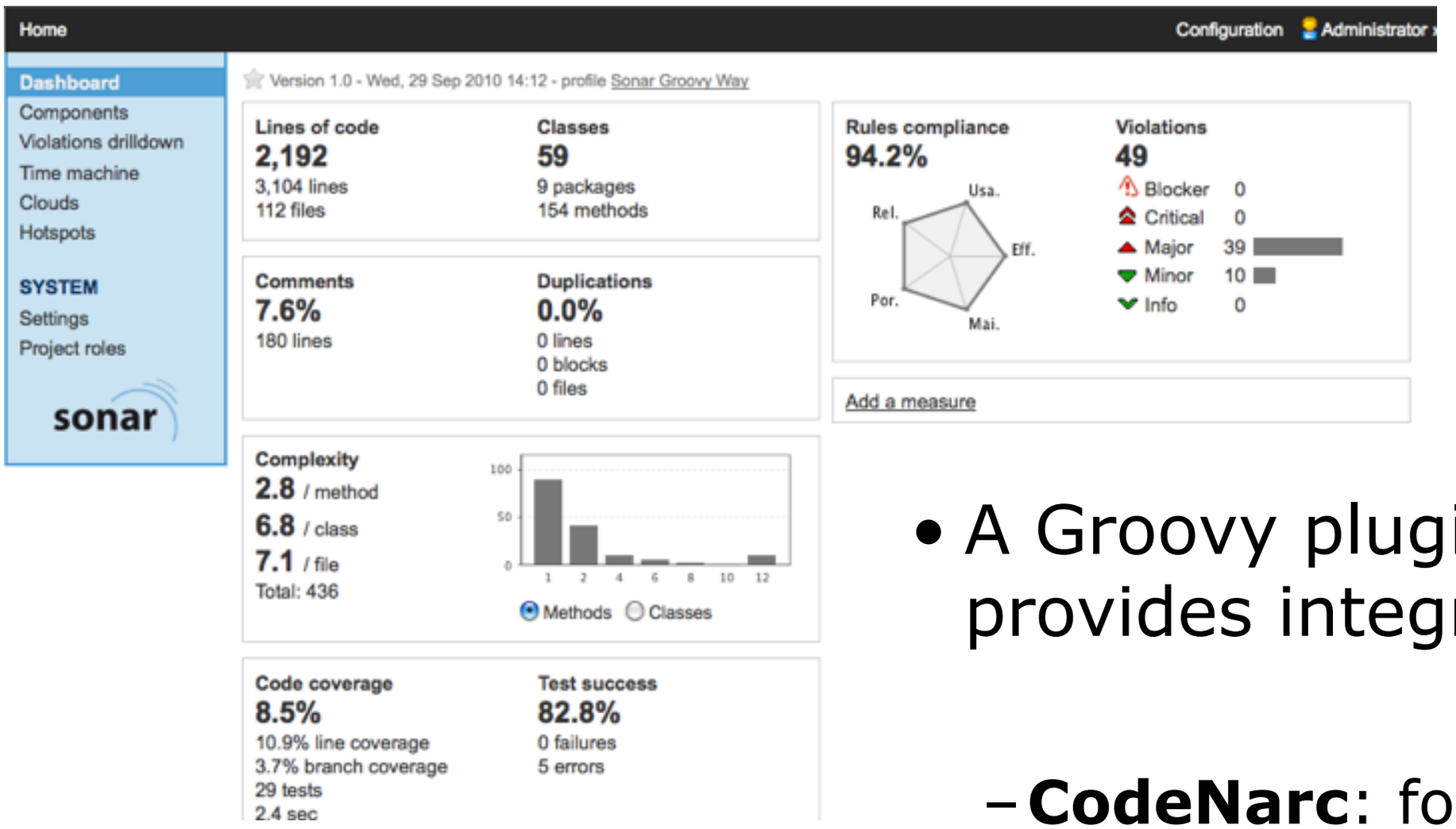
GMetrics Report: Sample

Report timestamp: 14 févr. 2011 08:56:47

Metric Results

Package/Class/Method	Complexity (total)	Complexity (average)	Class Lines (total)	Class Lines (average)	Method Lines (total)	Method Lines (average)
[p] All packages	*2*	*2.0*	*15*	*15.0*	*11*	*11.0*
[c] BadBoy	2	2.0	15	15	11	11.0
[m] someViolations	2	2	N/A	N/A	11	11

Sonar integration



sonar

- A Groovy plugin for **Sonar** provides integration for:
  - **CodeNarc**: for code quality
  - **GMetrics**: for metrics
  - **Cobertura**: for code coverage

Web services

HTTPBuilder (1/3)

- HTTPBuilder provides a convenient builder API for complex HTTP requests

```
def http = new HTTPBuilder( 'http://ajax.googleapis.com' )

http.request(GET, JSON) {
    uri.path = '/ajax/services/search/web'
    uri.query = [v: '1.0', q: 'Calvin and Hobbes']

    headers.'User-Agent' = 'Mozilla/5.0 Ubuntu/8.10 Firefox/3.0.4'

    response.success = { resp, json ->
        println resp.statusLine
        json.responseData.results.each {
            println "  ${it.titleNoFormatting}: ${it.visibleUrl}"
        }
    }
}
```

HTTPBuilder (2/3)

- Posting to a URL

```
import groovyx.net.http.HTTPBuilder
import static groovyx.net.http.ContentType.URLENC

def http = new HTTPBuilder( 'http://twitter.com/statuses/' )
def postBody = [status: 'update!', source: 'httpbuilder']

http.post( path: 'update.xml', body: postBody,
           requestContentType: URLENC ) { resp ->

    println "Tweet response status: ${resp.statusLine}"
    assert resp.statusLine.statusCode == 200
}
```

HTTPBuilder (3/3)

- Posting XML content to a URL

```
http.request(POST, XML) {
    body = {
        auth {
            user 'Bob'
            password 'pass'
        }
    }
}
```

GroovyWS (1/4)

springsource
A division of vmware

DISCLAIMER

« I always feel a bit dirty, speaking about SOAP »

GroovyWS (2/4)

- Publishing and consuming WS-I compliant SOAP web services

```
import groovyx.net.ws.WSClient

def proxy = new WSClient(
    "http://www.w3schools.com/webservices/tempconvert.asmx?WSDL",
    this.class.classLoader)
proxy.initialize()

def result = proxy.CelsiusToFahrenheit(0)
println "You are probably freezing at ${result} degrees Farhenheit"
```

- Can also deal with complex objects

GroovyWS (3/4)

- Publishing a service
  - considering a service

```
class MathService {
    double add(double arg0, double arg1) { arg0 + arg1 }
    double square(double arg0) { arg0 * arg0 }
}
```

  - publishing the service

```
import groovyx.net.ws.WSServer

def server = new WSServer()
server.setNode("MathService", "http://localhost:6980/MathService")
server.start()
```

GroovyWS (4/4)

springsource
A division of vmware

- Make sure to check **soapUI**
  - lets you write SOAP Web Services testing
  - using Groovy scripts to define a scenario!

- You'll feel much less dirty :-)

Summary

Summary

- «Groovy» is way more than just a language!
- It’s also a very rich and active **ecosystem**!
  - Grails, Griffon, Gradle, GPars, Spock, Gaelyk...

Summary

springsource
A division of vmware

Thanks for your attention!

- **References**:
  - http://groovy.codehaus.org/
  - http://grails.org
  - http://gaelyk.appspot.com/

Images used in this presentation

Zeus: http://clarte.eu.com/religions/articles/LArtdetrouveretderoberleFeuspirituelmultiplicateur/zeus-greek-mythology-687267_1024_768.jpg
Building: http://www.morguefile.com/archive/display/211651
Announce: http://www.napleswebdesign.net/wp-content/uploads/2009/06/press_release_11.jpg
Processors: http://www.morguefile.com/archive/display/12930
Trafic lights: http://www.flickr.com/photos/mad_house_photography/4440871380/sizes/l/in/photostream/
Bolts: http://www.morguefile.com/archive/display/29038
Web: http://www.morguefile.com/archive/display/137778
Light bulb: https://newsline.llnl.gov/retooling/mar/03.28.08_images/lightBulb.png
Quality metrics: http://www.morguefile.com/archive/display/14314
Duke ok GAE http://weblogs.java.net/blog/felipegaucho/archive/ae_gwt_java.png
Green leaf: http://www.flickr.com/photos/epsos/3384297473/
Jeepers creepers: http://www.flickr.com/photos/27663074@N07/3413698337/
Earth: http://www.flickr.com/photos/wwworks/2222523978/
Trafic light: http://rihancompany.com/var/243/35581-Traffic%20light%20drawing.jpg
Griffon: http://aldaria02.a.l.pic.centerblog.net/lz2levrz.jpg
Soap: http://www.flickr.com/photos/geishaboy500/104137788/
Disclaimer: http://4.bp.blogspot.com/_OyM252mHWvo/TT53ltKn7xI/AAAAAAAAC-A/YUNqmzfwce0/s1600/disclaimer.jpg.png