USING THE GROOVY ECOSYSTEM FOR RAPID JVM DEVELOPMENT

Schalk W. Cronjé

ABOUT ME

- Email: ysb33r@gmail.com
- Twitter / Ello : @ysb33r


SDK MAN!

SDKMAN

- Manages parallel version of multiple SDKs
- Mostly for (but not limited to) JVM-related systems
- Windows users can use Posh-GVM (Powershell)
  - Windows 10 Bash ??

```
curl -s http://get.sdkman.io | bash
```

SDKMAN DEMO

```
Usage: sdk <command> [candidate] [version]
       sdk offline <enable|disable>

   commands:
       install   or i    <candidate> [version]
       uninstall or rm   <candidate> <version>
       list      or ls   [candidate]
       use       or u    <candidate> [version]
       default   or d    <candidate> [version]
       current   or c    [candidate]
       outdated  or o    [candidate]
       version   or v
       broadcast or b
       help      or h
       offline           [enable|disable]
       selfupdate        [force]
       flush             <candidates|broadcast|archives|temp>

   candidate  :  the SDK to install: groovy, scala, grails, akka, etc.
                 use list command for comprehensive list of candidates
                 eg: $ sdk list

   version    :  where optional, defaults to latest stable if not provided
                 eg: $ sdk install groovy
```

- **SdkMan**: http://sdkman.io
- **Posh-GVM**: https://github.com/flofreud/posh-gvm
- @sdkmanager

APACHE GROOVY

A dynamic & static typed language for the JVM with REPL capability.

GROOVY VS JAVA

In Groovy:

- All class members are public by default
- No need to create getters/setters for public fields
- Both static & dynamic typing supported
- `def` means `Object`

CALLING METHODS

```
class Foo {
  void bar( def a,def b ) {}
}

def foo = new Foo()

foo.bar( '123',456 )
foo.bar '123', 456

foo.with {
  bar '123', 456
}
```

CALLING METHODS WITH CLOSURES

```
class Foo {
  void bar( def a,Closure b ) {}
}

def foo = new Foo()

foo.bar( '123',{ println it } )

foo.bar ('123') {
  println it
}

foo.bar '123', {
  println it
}
```

MAPS IN GROOVY

Hashmaps in Groovy are simple to use

```
def myMap = [ plugin : 'java' ]
```

Maps are easy to pass inline to functions

```
project.apply( plugin : 'java' )
```

Which can also be written as

```
project.with {
  apply plugin : 'java'
}
```

LISTS IN GROOVY

- Lists in Groovy are simple too

```
def myList = [ 'clone', 'http://github.com/ysb33r/GradleLectures' ]
```

- This makes it possible write a method call as

```
args 'clone', 'http://github.com/ysb33r/GradleLectures'
```

CLOSURE DELEGATION IN GROOVY

- When a symbol cannot be resolved within a closure, Groovy will look elsewhere
- In Groovy speak this is called a **Delegate**.
- This can be programmatically controlled via the `Closure.delegate` property.

CLOSURE DELEGATION IN GROOVY

```groovy
class Foo {
  def target
}

class Bar {
  Foo foo = new Foo()
  void doSomething( Closure c ) {
    c.delegate = foo
    c()
  }
}

Bar bar = new Bar()
bar.doSomething {
  target = 10
}
```

MORE CLOSURE MAGIC

If a Groovy class has a method `call(Closure)`, the object can be passed a closure directly.

```
class Foo {
  def call( Closure c) { /* ... */ }
}

Foo foo = new Foo()
foo {
  println 'Hello, world'
}

// This avoids ugly syntax
foo.call({ println 'Hello, world' })
```

SMOOTH OPERATOR: ELVIS

?:

```
// Selecting a default value if variable is not set

String a
String b = 'def'

// Prints 'foo'
println a ?: 'foo'

// Print 'def'
println b ?: 'foo'
```

SMOOTH OPERATOR: SAFE NAVIGATION

?.

```
// Returns a value if not-null, otherwise null

String a
String b = 'def'

assert a?.size() == null

assert b?.size() == 3

println a.size() // Throws NPE
```

SMOOTH OPERATOR: SPREAD

*.

```
// Apply operation to each member in collection

def a = [ 'a','bb','c' ]

assert a*.size() == [ 1,2,1 ]
```

STATIC VS DYNAMIC

- By default code in Groovy is compiled with dynamic typing.
- Typing can be modifed by selective application of annotations:
  - `@CompileStatic` - compile static typing
  - `@CompileDynamic` - compile dynamic typing
  - `@TypeChecked` - compile dynamic, but check types
- Apply to classes & methods.

IS GROOVY SLOWER THAN JAVA?

- Yes and No.
- Context is critical
  - Russell Winder has done number of experiments
- JSON parsing in Groovy is faster than Java.
- Develop first, measure, then optimise!
  - Remember C++ → C → Assembler optimisation?

APACHE GROOVY

- Apache Groovy home
  - http://groovy-lang.org/
- Documentation
  - http://groovy-lang.org/documentation.html
- Twitter
  - @ApacheGroovy

4.15

5.1

GRADLE

- Very modern, next-generation build & deployment pipeline tool
- DSL is based on Groovy
- Vast collection of plugins

GRADLE

```
apply plugin : 'groovy'

repositories {
  jcenter()
}

dependencies {
    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
    testCompile "org.gebish:geb-spock:0.13.0"
    testCompile "org.seleniumhq.selenium:selenium-api:2.53.0"
    testRuntime "org.seleniumhq.selenium:selenium-support:2.53.0"
    testRuntime "org.seleniumhq.selenium:selenium-firefox-driver:2.53.0"
}
```

GRADLE

- Make it easy to integrate development and complex testing
- Deploy local & to remote sites via plugins
- Use Docker local & remote
- "I do not have means to test on my machine" becomes less of an argument.

GRADLE JVM POLYGOT

- Build a distributable application packaged as as ZIP
- Runnable via shell script or batch file
- Contains classes written Java, Groovy & Kotlin source
- Test source code with Spock Framework

GRADLE JVM POLYGOT

```
apply plugin : 'java'
apply plugin : 'groovy'
apply plugin : 'com.zoltu.kotlin'
apply plugin : 'application'

repositories {
    jcenter()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.3'
    compile 'org.jetbrains.kotlin:kotlin-stdlib:1.0.1-1'
    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}

version = '1.0'
mainClassName = "gradle.workshop.HelloJava"
```

GRADLE DEPENDENCY MANAGEMENT

- Easy to use
- Flexible to configure for exceptions
- Uses `dependencies` closure
- First word on line is usually name of a configuration.
  - Configurations are usually supplied by plugins.
- Dependencies are downloaded from repositories
- Maven coordinates are used as format

GRADLE REPOSITORIES

- Specified within a `repositories` closure
- Processed in listed order to look for dependencies
- `jcenter()` preferred open-source repo.
- `mavenLocal()`, `mavenCentral()`, `maven {}`
- Ivy repositories via `ivy {}`
- Flat-directory repositories via `flatDir`

GRADLE REPOSITORIES

```
repositories {
    jcenter()
    mavenCentral()
    maven { url "https://plugins.gradle.org/m2/" }
    ivy {
     url 'file://path/to/repo'
     layout 'pattern', {
        artifact '[module]/[revision]/[artifact](.[ext])'
        ivy '[module]/[revision]/ivy.xml'
     }
    }
}
```

BUILDSCRIPT

- The `buildscript` closure is special
- It tells Gradle what to load into the classpath before evaluating the script itself.
- It also tells it where to look for those dependencies.
- Even though Gradle 2.1 has added a new way of adding external plugins, `buildscript` are much more flexible.

EXTENSIONS

- Extensions are global configuration blocks added by plugins.
- Example: The `jruby-gradle-base` plugin will add a `jruby` block.

```
apply plugin: 'com.github.jruby-gradle.base'

jruby {
  defaultVersion = '1.7.11'
}
```

GRADLE TASKS

Can be based upon a task type

```
task runSomething ( type : Exec ) {
    command 'git'
    args 'clone', 'https://bitbucket.com/ysb33r/GradleWorkshop'
}
```

Can be free-form

```
task hellowWorld << {
  println 'Hello, world'
}
```

GRADLE: BUILDING C++

```
apply plugin : 'cpp'
```

GRADLE: BUILDING C++

- Need to change convention from traditional C++ projects
- `.cpp` files go in `src/${name}/cpp`
- Exported headers files go in `src/${name}/headers`
- Local header files should be in `src/${name}/cpp`
- Object files will end up in `${buildDir}/objs`
- Binary files will end up in `${buildDir}/binaries`

BUILDING C++: PROJECT LAYOUT

```
├── build.gradle
└── src
    └── hello
        ├── cpp
        │   └── hello.cpp
        │
        └── headers
            └── hello.hpp
```

BUILDING C++: EXISTING PROJECTS

- Source directories can be adjusted
- Alternative compiler locations

BUILDING C++: ALTERNATIVE SOURCE

```
sources {
    cpp {
        source {
            srcDir "myDir"
            include "**/*.cpp"
        }
    }
}
```

BUILDING C++: TOOL SUPPORT

Operating System	Tool Chain	Official
Linux	gcc, clang	Y
MacOS X	Xcode	Y
	gcc-macports, clang-macports	N
Windows	Visual C++, gcc-cygwin32, gcc-mingw	Y
	gcc-cygwin64	N

Unix-like	gcc, clang	N

BUILDING C++: EXECUTABLE

```
model {
    components {
        hello(NativeExecutableSpec)
    }
}
```

BUILDING C++: CUSTOM COMPILER

```
model {
    toolChains {
        gcc(Gcc) {
            path '/installed/in/foo/dir/gcc'

            eachPlatform {
                cppCompiler.withArguments { args ->
                    args << "-DNDEBUG"
                }
            }
        }
    }
}
```

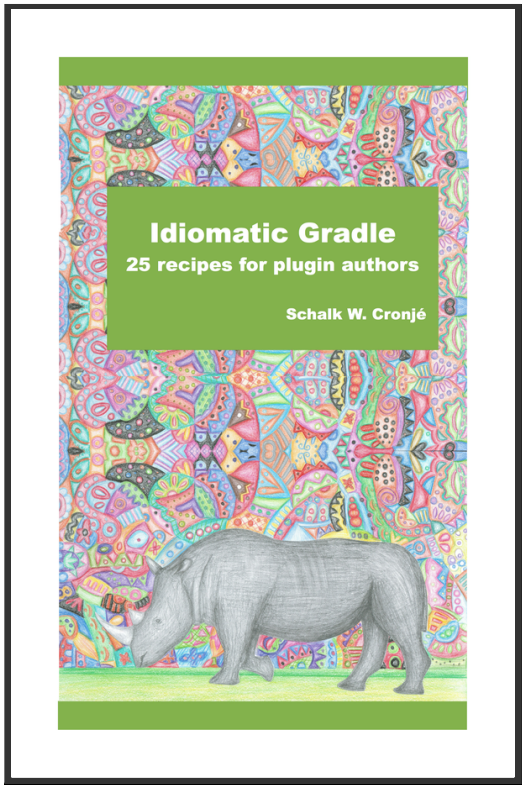
BUILDING C++: OTHER FEATURES

- Cross compilation
- Multi-architecture targets
- Set compiler & linker flags
- Multi-variant builds

BUILDING C++: WEAKNESS

- Currently only have built-in support for CUnit
- Only platform configuration tool support is `CMake`
- No Autotools equivalent
- DSL can be slicker

- **Website**: http://gradle.org
- **Plugins Portal**: http://plugins.gradle.org
- **User Forum**: http://discuss.gradle.org
- @gradle
- @DailyGradle



http://leanpub.com/idiomaticgradle

LET'S REVIEW OUR TESTS

Summary:

Created on Mon Apr 04 15:05:57 BST 2016 by schalkc

Executed features	Failures	Errors	Skipped	Success rate	Time
1	0	0	0	100.0%	0.042 seconds

Features:

- A calculator must be able to add numbers

A calculator must be able to add numbers — Return

Given: That I have a calculator

When: I add 2, 3 and -20

Then: The answer should be -15

LET'S GET UNIT TESTING

```
class CalculatorSpec extends Specification {
    def "A calculator must be able to add numbers"() {

        given: 'That I have a calculator'
        def calc = new Calculator()

        when: "I add 2, 3 and -20"
        def answer = calc.plus 2,3,-20

        then: "The answer should be -15"
        answer == -15
    }
}
```

WHEN THIS TEST FAILS

Class fasttrackvm.calculator.CalculatorSpec

all > fasttrackvm.calculator > CalculatorSpec

1	1	0	0.205s
tests	failures	ignored	duration

0%
successful

Failed tests Tests

A calculator must be able to add numbers

```
Condition not satisfied:

answer == -13
|      |
-15    false

        at fasttrackvm.calculator.CalculatorSpec.A calculator must be able to add numbers(CalculatorSpec.groovy:21)
```

WHEN THIS TEST FAILS

Features:

- A calculator must be able to add numbers

A calculator must be able to add numbers Return

Given: That I have a calculator

When: I add 2, 3 and -20

Then: The answer should be -15

The following problems occurred:

- `Condition not satisfied:`

```
answer == -13
|      |
-15    false
```

SPOCK FRAMEWORK

- Built on top of `JUnit 4.x`.
- Use it with all your `JUnit` tools!
- Use for more than just unit tests.
- Mocks & stubs included
- Easily test Java, Groovy, Kotlin, Scala etc.

DATA-DRIVEN TESTS

```
@Unroll
def "A calculator must be able to multiply"() {

    given: 'That I have a multiplying calculator'
    def calc = new Calculator()

    expect: "The answer to be #answer"
    answer == calc.multiply (a,b,c)

    where: "The operation is #a * #b * #c"
    a | b | c || answer
    3 | 7 | 5 || 105
    1 | 3 | 0 || 0
    2 | -1| 1 || -2
}
```

ANOTHER COOL FAILURE REPORT

A calculator must be able to multiply[7]

Given: That I have a multiplying calculator

Expect: The answer to be 1

Where: The operation is 1 * 3 * 0

The following problems occurred:

- Condition not satisfied:

```
answer == calc.multiply (a,b,c)
|      |  |    |         | | |
1      |  |    0         1 3 0
       |  fasttrackjvm.calculator.Calculator@27df0d8
       false
```

HANDLING EXCEPTIONS

```
def "Dividing by zero should throw an exception"() {
    given: 'That I have a dividing calculator'
    def calc = new Calculator()

    when: "I divide by zero"
    calc.divide 1,0

    then: "I expect an error"
    thrown(ArithmeticException)
}
```

MOCK OUT INTERFACES

```
public interface RemoteCalculator {
    public Number plus(Number... args);
}
```

```
def "Remote calculator"() {
    given: "A remote calculator is available"
    def calc = Mock(RemoteCalculator)

    when: "Multiple items are sent for addition"
    calc.plus 1,2,3,4,5

    then: "The calculator is only called once"
    1 * calc.plus(_)
}
```

SPOCK FRAMEWORK

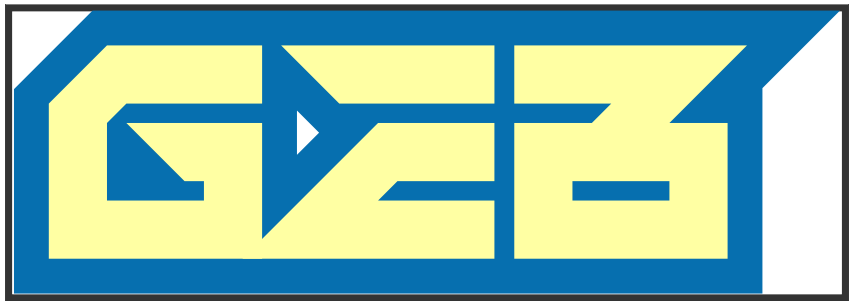
- Spock Framework:
  - http://spockframework.github.io/spock/docs/1.0/index.html
- Spock Reports
  - https://github.com/renatoathaydes/spock-reports

WEBSITE TESTING

- One of the most predominant area of testing of this decade
- Test-driven webpage development is less effort in long run
- Selenium is the leading tool
- Selenium tests can be overly verbose

TRIVIAL TEST EXAMPLE


```
def "Learn about testing checkboxes"() {
    when: "I go to that the-internet site"
    go "${webroot}/checkboxes"

    then: 'I am expecting Checkbox page'
    $('h3').text() == 'Checkboxes'

    and: 'The checkbox states are no & yes'
    $(By.id('checkboxes')).$('input')*.@checked == ['','true']
}
```



8.2

9.1

GEB

- Integrates with
  - Spock Framework
  - JUnit
  - TestNG
  - Cucumber-JVM
- Makes Selenium readable
  - Anything you can do in Selenium you can do in Geb

TRIVIAL TEST EXAMPLE COMPLETE

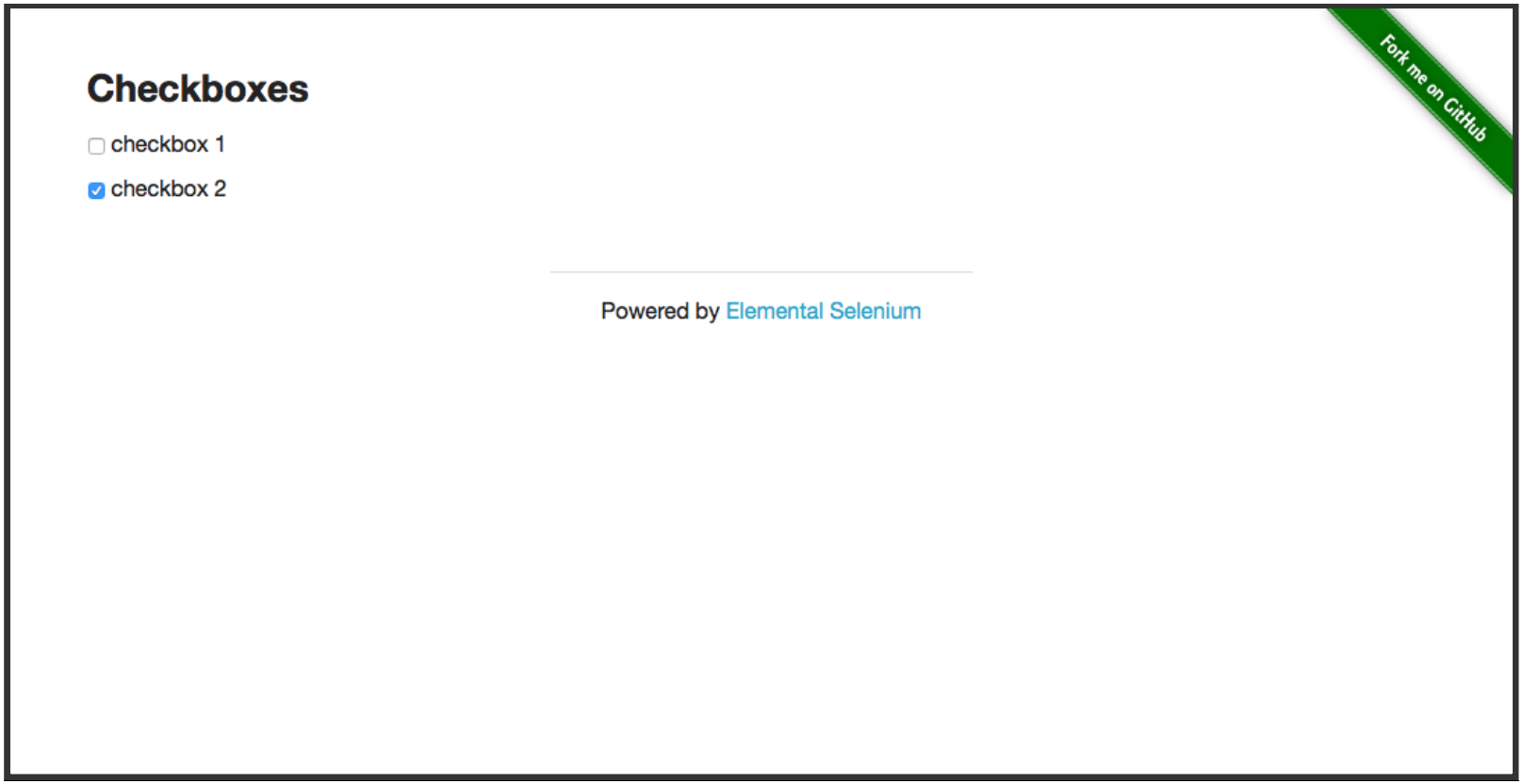
```
class CheckboxExampleSpec extends GebSpec {
    def "Learn about testing checkboxes"() {
        when: "I go to that the-internet site"
        go "${webroot}/checkboxes"

        then: 'I am expecting Checkbox page'
        $('h3').text() == 'Checkboxes'

        and: 'The checkbox states are no & yes'
        $(By.id('checkboxes')).$('input')*.@checked == ['','true']
    }
}
```

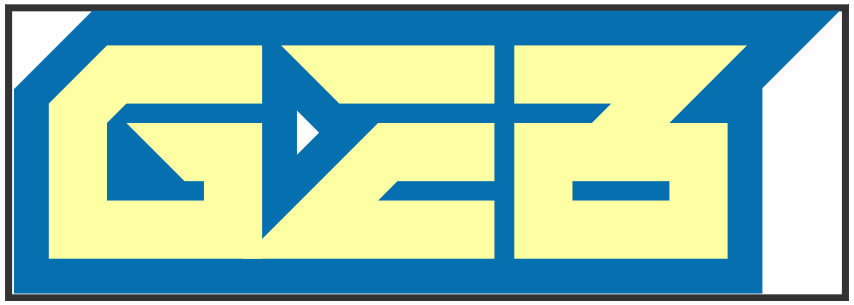
GRABBING SCREENSHOTS

```
class CheckboxReportingSpec extends GebReportingSpec {
    def 'Learn about testing checkboxes'() {
        when: 'I go to that the-internet site'
        go "${webroot}/checkboxes"
        report 'checkbox-screen'

        then: 'I am expecting Checkbox page'
        $('h3').text() == 'Checkboxes'

        and: 'The checkbox states are no & yes'
        $(By.id('checkboxes')).$('input')*.@checked == ['','true']
    }
}
```

- Keep record during test
- Stores HTML & PNG.

GRABBING SCREENSHOTS





- http://gebish.org
- @GemFramework

9.6

10.1

RATPACK FRAMEWORK

- Set of Java libraries for building modern HTTP applications.
- Built on Java 8, Netty and reactive principles.
- Groovy syntax for those who prefer it.

RATPACK QUICKSTART SERVER

```
@Grapes([
    @Grab('io.ratpack:ratpack-groovy:1.2.0'),
    @Grab('org.slf4j:slf4j-simple:1.7.12')
])
import static ratpack.groovy.Groovy.ratpack

ratpack {
    handlers {
        get {
            render "Hello World!\n\n"
        }
        get(":name") {
            render "Hello $pathTokens.name!\n\n"
        }
    }
}
```

RATPACK'S TESTHTTPCLIENT

- Can be used standalone
- Use maven coordinates:
  - `io.ratpack:ratpack-test:1.2.0`

RATPACK'S TESTHTTPCLIENT

```
ApplicationUnderTest app = new ApplicationUnderTest() {
    @Override
    URI getAddress() { "http://127.0.0.1:${PORT}".toURI() }
}

@Delegate TestHttpClient client = TestHttpClient.testHttpClient(app)
```

```
def "The echo path should return what is send to it"() {
    given: "A simple text request"
    requestSpec { pay ->
          pay.body.type(MediaType.PLAIN_TEXT_UTF8).text('नमस्ते दुनिया')
    }

    when: "The data is posted"
    post '/'

    then: "It should be echoed back"
    response.statusCode == 200
    response.body.text == 'You said: नमस्ते दुनिया'
}
```

QUICK HTTP TESTING

- Quickly point to a remote or in-process server
- Build payload in a simplistic manner
- Execute the verb
- Check the response in a readable manner

QUICK HTTP TESTING

```
def "The echo path should return what is send to it"() {
    given: "A simple text request"
    requestSpec { pay ->
          pay.body.type(MediaType.PLAIN_TEXT_UTF8).text('नमस्ते दुनिया')
    }

    when: "The data is posted"
    post '/'

    then: "It should be echoed back"
    response.statusCode == 200
    response.body.text == 'You said: नमस्ते दुनिया'
}
```

OTHER GR8-UNIVERSE TOOLS

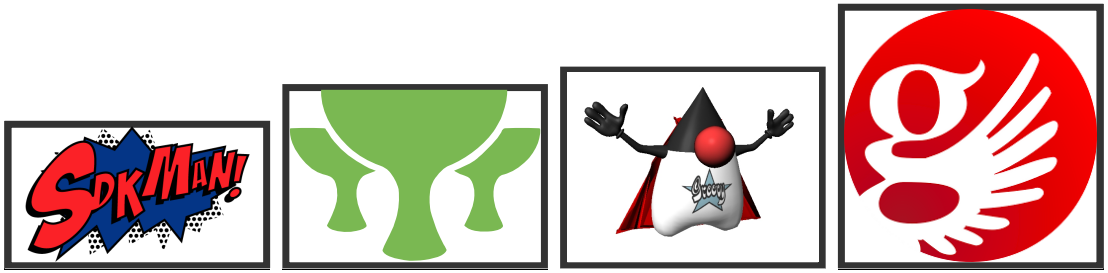
- **Grails** - Build web applications fast
- **Griffon** - Best way to build desktop apps
- **Grooscript** - Groovy to Javascript
- **Groovy VFS** - File operations on remote servers using variety of protocols
- **Sshoogr** - SSH & SCP
- **Groowin** - WinRM

THANK YOU

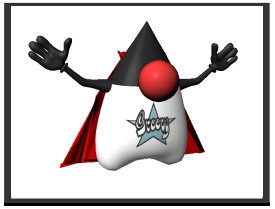
Schalk W. Cronjé

Email: ysb33r@gmail.com

Twitter: @ysb33r



BITS & PIECES



GROOVY VFS

```
def vfs = new VFS()

vfs {
  cp 'ftp://some.server/pub/file0.txt?vfs.ftp.passiveMode=1',
    'sftp://user:pass@another.server/path/file1.txt',
      overwrite : true
}
```

https://github.com/ysb33r/groovy-vfs

SSHOOGR

```
remoteSession('user:pass@another.server') {
  exec 'rm -rf ~/tmp/*'
  exec 'touch this.file'
  remoteFile('this.file').text = "enabled=true"
}
```

https://github.com/aestasit/sshoogr

GROOSCRIPT

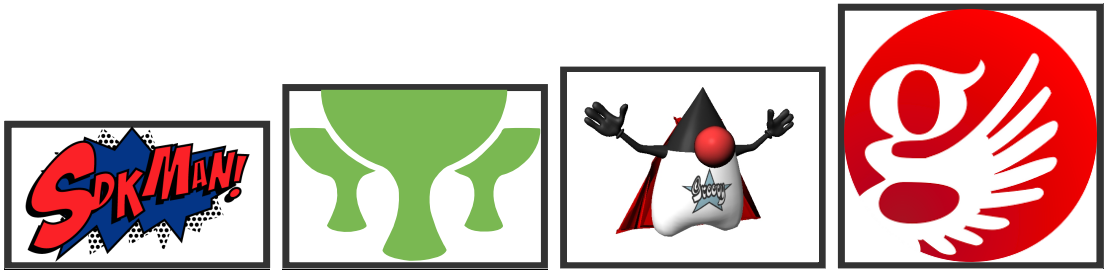
http://grooscript.org/

THANK YOU

Schalk W. Cronjé

Email: ysb33r@gmail.com

Twitter: @ysb33r



ABOUT THIS PRESENTATION

- Written in Asciidoctor (1.5.3.2)
- Styled by asciidoctor-revealjs extension
- Built using:
  - Gradle
  - gradle-asciidoctor-plugin
  - gradle-vfs-plugin
- All code snippets tested as part of build