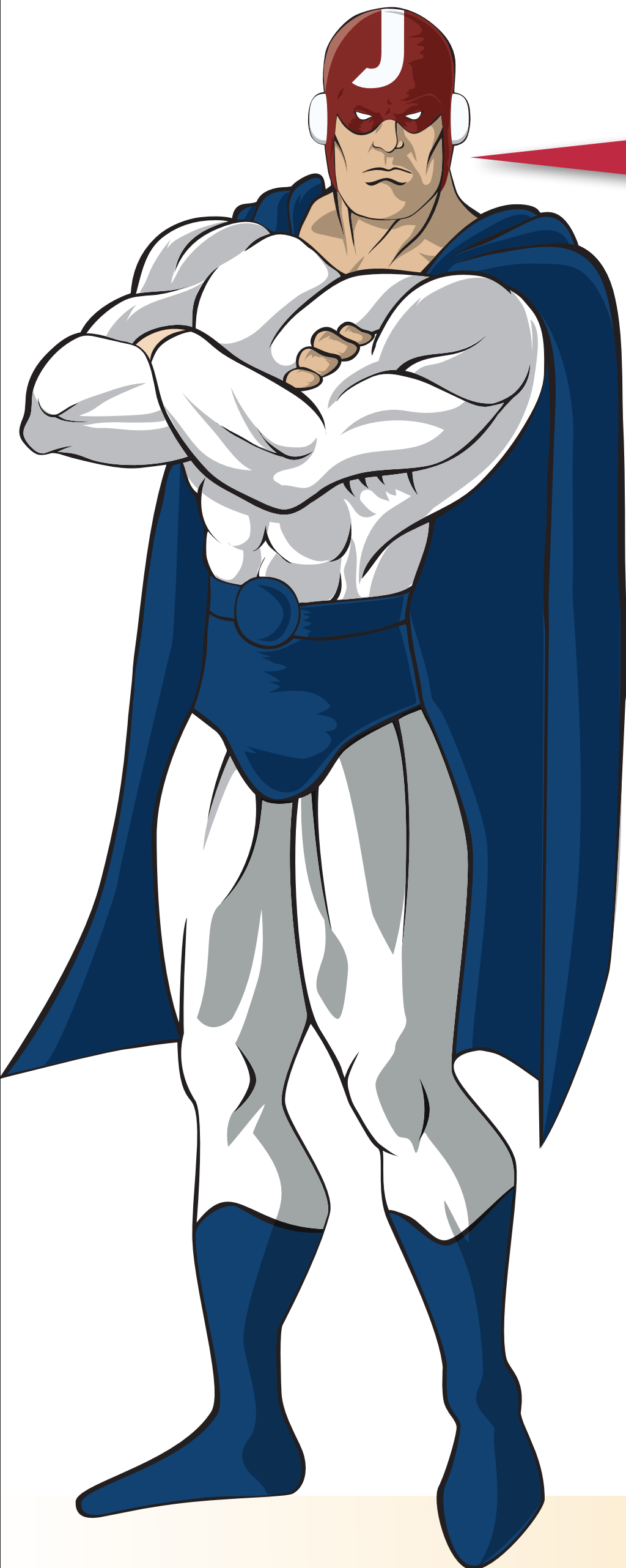


# Benefit from Groovy now, when, why and how

Guillaume Laforge  
@glaforge 

Idolville™



By the way, you know  
there's a Groovy User Group  
here in Stockholm?

[http://www.meetup.com/  
Stockholm-Groovy-User-  
Group/](http://www.meetup.com/Stockholm-Groovy-User-Group/)

**Join now,**

**when, why and how**

**Guillaume Laforge**

**@glaforge** 

**Idol Grove™**

# Guillaume Laforge

Groovy project lead  
at **Pivotal**<sup>TM</sup>



@glaforge



<http://glaforge.appspot.com>





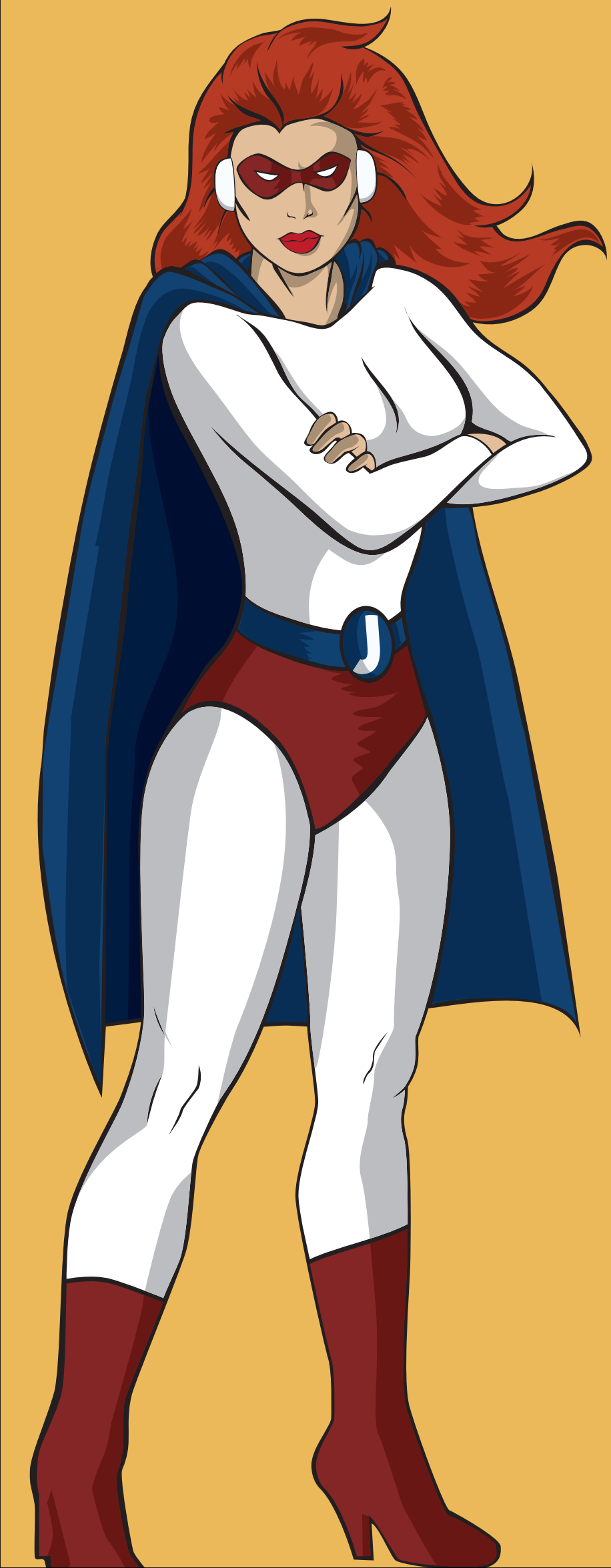


# Introduction

---



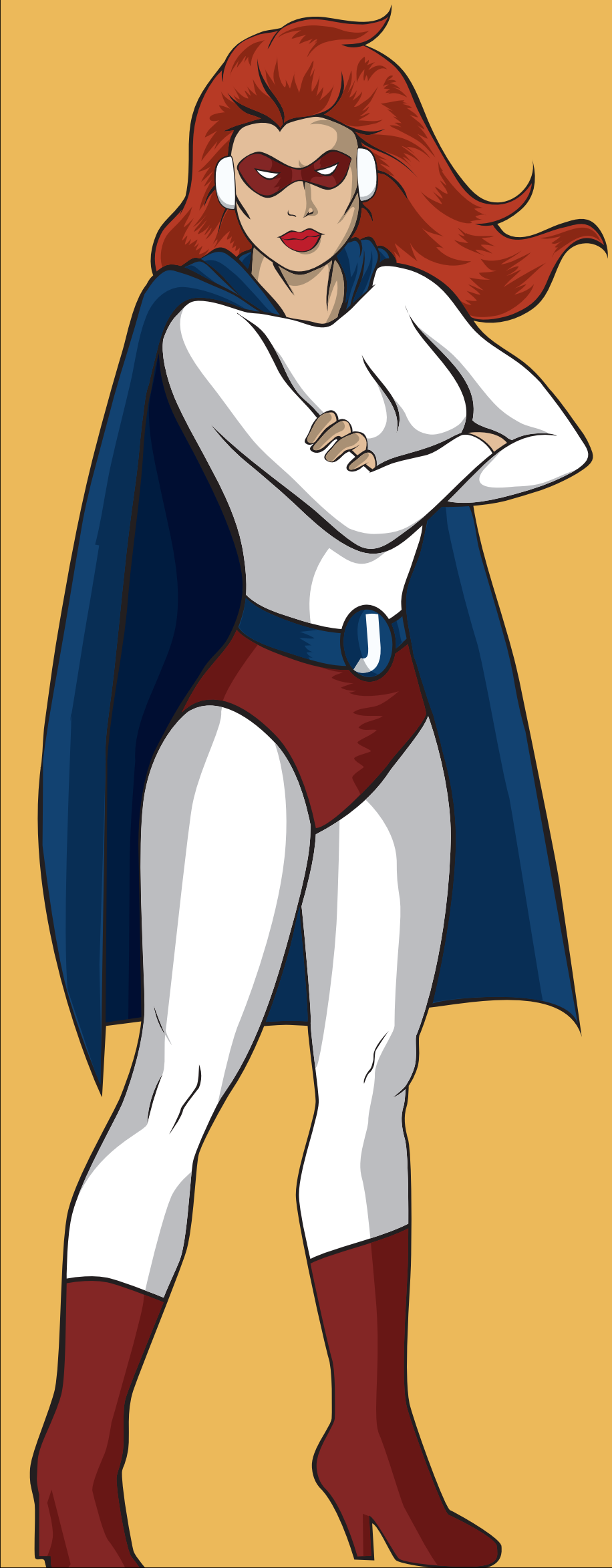
Groovy in 5 minutes



# Groovy...

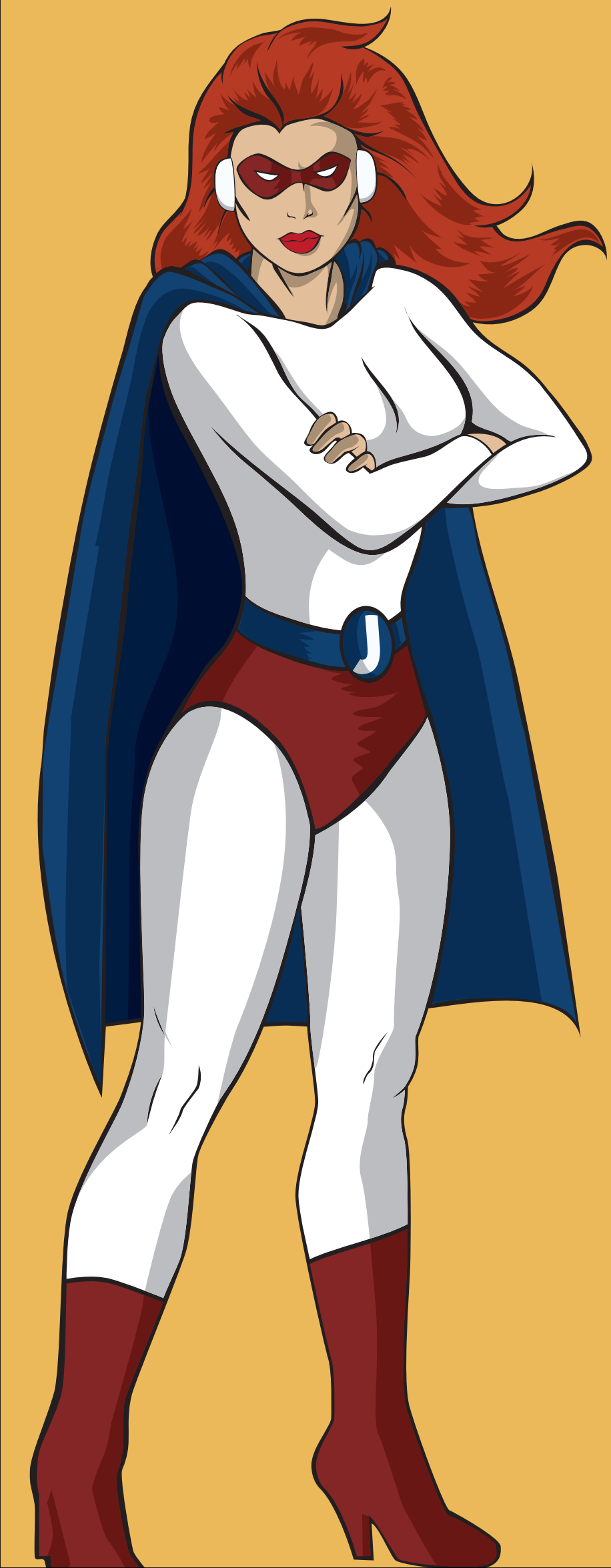
Open Source  
alternative language  
for the JVM





# Groovy...

Object-oriented, dynamic,  
with a functional flavor



# Groovy...

...also supports  
static type checking &  
static compilation





million  
downloads  
in 2013



3

million  
downloads  
in 2013



Twice as much  
as in 2012!



3

million  
downloads  
in 2013









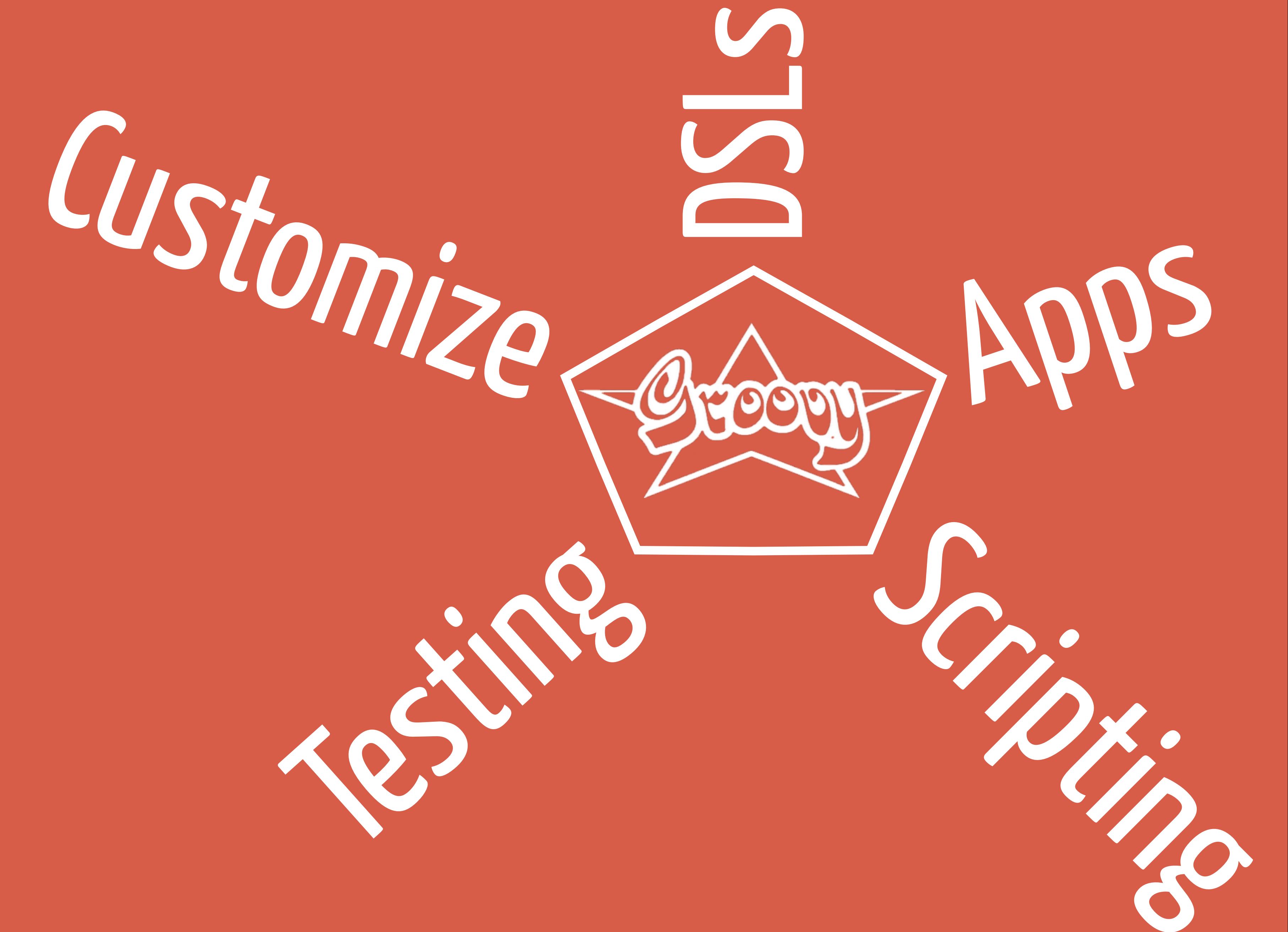


Yup, we're all  
using Groovy!



# BIG USE CASES

---





# BIG USE CASES

---

Customize

STSD

Apps



Testing

Scripting

# BIG USE CASES

---

Customize

STSD



Apps

Testing

Scripting

# BIG USE CASES

*Customize*



*Testing*

*Scripting*

*DSLs*

*Apps*

# BIG USE CASES

STSD



Customize

Apps

Testing

Scripting

# BIG USE CASES



Customize

STSD



Apps

Testing

Scripting



# What for?

# **What for?**

*for what kind of tasks*

*What for?*

*for what kind of tasks*

*Who?*

**What for?**

*for what kind of tasks*

**Who?**

*projects, companies  
using that pattern*

**What for?**

*for what kind of tasks*

**Who?**

*projects, companies  
using that pattern*

**How?**



**What for?**

*for what kind of tasks*

**Who?**

*projects, companies  
using that pattern*

**How?**

*do it yourself*

Case 1



# Scripting

---

Task automation  
Build automation  
DevOps

- Groovy is also a scripting language, you can easily...
  - run a Groovy script with the groovy command
  - @Grab & import third-party libraries
    - no need for a dedicated project or a build file
    - leverage the wealth of Ant tasks with the Groovy Ant builder
  - invoke other command-line tools

# Who?

## Scripting: task automation



# Who?

## Scripting: task automation

You or  
anybody :-)





# Who?

## Scripting: task automation

You or  
anybody :-)

Or me for grepping  
through the logs to  
compute download  
stats



# Who?

## Scripting: task automation

You or  
anybody :-)

Or me for grepping  
through the logs to  
compute download  
stats



Anyone who's  
not a Bash guru!

# Who?

## Scripting: task automation

You or  
anybody :-)

Anyone who's  
not a Bash guru!

Or me for grepping  
through the logs to  
compute download  
stats

Groovy team exports  
Confluence wiki to html,  
then transforms to  
AsciiDoc format

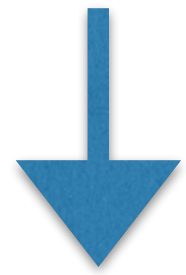




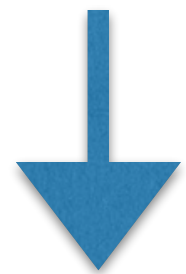
# How?

# Scripting: task automation

 Confluence



<html/>



AsciiDoctor

```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

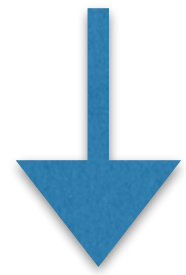
src.toFile().eachFileRecurse { f ->
    def relative = src.relativeize(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagNode, htmlNode ->
            tagNode?.attributes?.remove 'class'
            if ('td' == tagNode?.name || 'th' == tagNode?.name) {
                tagNode.name = 'td'
                String txt = tagNode.text
                tagNode.removeAllChildren()
                tagNode.insertChild(0, new ContentNode(txt))
            }

            true
        } as TagNodeVisitor)
        serializer.writeToFile(
            result, tmpHtml.absolutePath, "utf-8"
        )
        "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
        tmpHtml.delete()
    }
}
```

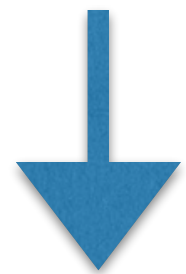
# How?

# Scripting: task automation

 Confluence



<html/>



AsciiDoctor

```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

src.toFile().eachFileRecurse { f ->
    def relative = src.relativeTo(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagNode, htmlNode ->
            tagNode?.attributes?.remove 'class'
            if ('td' == tagNode?.name || 'th' == tagNode?.name) {
                tagNode.name = 'td'
                String txt = tagNode.text
                tagNode.removeAllChildren()
                tagNode.insertChild(0, new ContentNode(txt))
            }

            true
        } as TagNodeVisitor)
        serializer.writeToFile(
            result, tmpHtml.absolutePath, "utf-8"
        )
        "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
        tmpHtml.delete()
    }
}
```

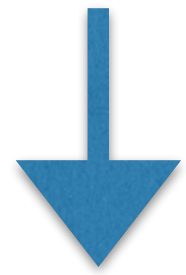
Use HtmlCleaner to clean the Confluence wiki HTML export



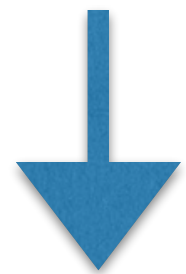
# How?

# Scripting: task automation

 Confluence



<html/>



AsciiDoctor

```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

src.toFile().eachFileRecurse { f ->
    def relative = src.relativeTo(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagNode, htmlNode ->
            tagNode?.attributes?.remove 'class'
            if ('td' == tagNode?.name || 'th' == tagNode?.name) {
                tagNode.name = 'td'
                String txt = tagNode.text
                tagNode.removeAllChildren()
                tagNode.insertChild(0, new ContentNode(txt))
            }

            true
        } as TagNodeVisitor)
        serializer.writeToFile(
            result, tmpHtml.absolutePath, "utf-8"
        )
        "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
        tmpHtml.delete()
    }
}
```

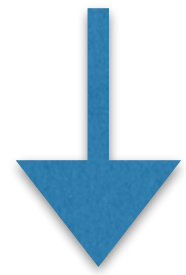
Use HtmlCleaner to clean the Confluence wiki HTML export

Transform the cleaned HTML into AsciiDoctor format

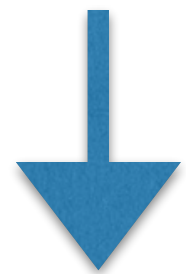
# How?

# Scripting: task automation

 Confluence



<html/>



AsciiDoctor

```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

src.toFile().eachFileRecurse { f ->
    def relative = src.relativeize(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagNode, htmlNode ->
            tagNode?.attributes?.remove 'class'
            if ('td' == tagNode?.name || 'th' == tagNode?.name) {
                tagNode.name = 'td'
                String txt = tagNode.text
                tagNode.removeAllChildren()
                tagNode.insertChild(0, new ContentNode(txt))
            }

            true
        } as TagNodeVisitor)
        serializer.writeToFile(
            result, tmpHtml.absolutePath, "utf-8"
        )
        "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
        tmpHtml.delete()
    }
}
```



```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

src.toFile().eachFileRecurse { f ->
    def relative = src.relativeTo(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        f.toFile().textReader().use { it }
```



```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
```

```
import org.htmlcleaner.*
```

This is how you can grab  
a dependency...

```
def src = new File('html').toPath()
```

```
def dst = new File('asciidoc').toPath()
```

```
def cleaner = new HtmlCleaner()
```

```
def props = cleaner.properties
```

```
props.translateSpecialEntities = false
```

```
def serializer = new SimpleHtmlSerializer(props)
```

```
src.toFile().eachFileRecurse { f ->
```

```
    def relative = src.relativeTo(f.toPath())
```

```
    def target = dst.resolve(relative)
```

```
    if (f.isDirectory()) {
```

```
        target.toFile().mkdir()
```

```
    } else if (f.name.endsWith('.html')) {
```

```
        def tmpHtml = File.createTempFile('clean', 'html')
```

```
        println "Converting $relative"
```



```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*
```

This is how you can grab  
a dependency...

```
def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()
```

```
def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)
```

...and use the library  
right away, without a  
build file or IDE

```
src.toFile().eachFileRecurse { f ->
    def relative = src.relativeize(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
```



```
@Grab('net.sourceforge.htmlcleaner:htmlcleaner:2.4')
import org.htmlcleaner.*

def src = new File('html').toPath()
def dst = new File('asciidoc').toPath()

def cleaner = new HtmlCleaner()
def props = cleaner.properties
props.translateSpecialEntities = false
def serializer = new SimpleHtmlSerializer(props)

src.toFile().eachFileRecurse { f ->
    def relative = src.relativeTo(f.toPath())
    def target = dst.resolve(relative)
    if (f.isDirectory()) {
        target.toFile().mkdir()
    } else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        f.copyTo(tmpHtml)
        tmpHtml.renameTo(target.toFile())
    }
}
```



```
def serializer = new SimpleHtmlSerializer(props)
```

```
src.toFile().eachFileRecurse { f ->
  def relative = src.relativeTo(f.toPath())
  def target = dst.resolve(relative)
  if (f.isDirectory()) {
    target.toFile().mkdir()
  } else if (f.name.endsWith('.html')) {
    def tmpHtml = File.createTempFile('clean', 'html')
    println "Converting $relative"
    def result = cleaner.clean(f)
    result.traverse({ tagNode, htmlNode ->
      tagNode?.attributes?.remove 'class'
      if ('td' == tagNode?.name || 'th' == tagNode?.name) {
        tagNode.name = 'td'
        String txt = tagNode.text
        tagNode.removeAllChildren()
        tagNode.insertChild(0, new ContentNode(txt))
      }
    })
  }
}
```



```
def serializer = new SimpleHtmlSerializer(props)
```

```
src.toFile().eachFileRecurse { f ->
  def relative = src.relativeTo(f.toPath())
  def target = dst.resolve(relative)
  if (f.isDirectory()) {
    target.toFile().mkdir()
  } else if (f.name.endsWith('.html')) {
    def tmpHtml = File.createTempFile('clean', 'html')
    println "Converting $relative"
    def result = cleaner.clean(f)
    result.traverse({ tagNode, htmlNode ->
      tagNode?.attributes?.remove 'class'
      if ('td' == tagNode?.name || 'th' == tagNode?.name) {
        tagNode.name = 'td'
        String txt = tagNode.text
        tagNode.removeAllChildren()
        tagNode.insertChild(0, new ContentNode(txt))
      }
    })
  }
}
```

Traverse the file  
system easily



```
def serializer = new SimpleHtmlSerializer(props)
```

```
src.toFile().eachFileRecurse { f ->
  def relative = src.relativeTo(f.toPath())
  def target = dst.resolve(relative)
  if (f.isDirectory()) {
    target.toFile().mkdir()
  } else if (f.name.endsWith('.html')) {
    def tmpHtml = File.createTempFile('clean', 'html')
    println "Converting $relative"
    def result = cleaner.clean(f)
    result.traverse({ tagName, htmlNode ->
      tagName?.attributes?.remove 'class'
      if ('td' == tagName?.name || 'th' == tagName?.name) {
        tagName.name = 'td'
        String txt = tagName.text
        tagName.removeAllChildren()
        tagName.insertChild(0, new ContentNode(txt))
      }
    })
  }
}
```



```
file().eachFileRecurse { f ->
    relative = src.relativeize(f.toPath())
    target = dst.resolve(relative)
    (f.isDirectory()) {
        target.toFile().mkdir()
    }
    else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagName, htmlNode ->
            tagName?.attributes?.remove 'class'
            if ('td' == tagName?.name || 'th' == tagName?.name) {
                tagName.name = 'td'
                String txt = tagName.text
                tagName.removeAllChildren()
                tagName.insertChild(0, new ContentNode(txt))
            }

            true
        } as TagNodeVisitor)
        serializer.writeToFile(
            result, tmpHtml.absolutePath, "utf-8"
        )
        "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
        tmpHtml.delete()
    }
}
```



```
file().eachFileRecurse { f ->
    relative = src.relativeize(f.toPath())
    target = dst.resolve(relative)
    (f.isDirectory()) {
        target.toFile().mkdir()
    }
    else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagName, htmlNode ->
            tagName?.attributes?.remove 'class'
            if ('td' == tagName?.name || 'th' == tagName?.name) {
                tagName.name = 'td'
                String txt = tagName.text
                tagName.removeAllChildren()
                tagName.insertChild(0, new ContentNode(txt))
            }
        }, true)
    } as TagNodeVisitor
    serializer.writeToFile(
        result, tmpHtml.absolutePath, "utf-8"
    )
    "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
    tmpHtml.delete()
}
```

Execute external  
commands



```
file().eachFileRecurse { f ->
    relative = src.relativeize(f.toPath())
    target = dst.resolve(relative)
    (f.isDirectory()) {
        target.toFile().mkdir()
    }
    else if (f.name.endsWith('.html')) {
        def tmpHtml = File.createTempFile('clean', 'html')
        println "Converting $relative"
        def result = cleaner.clean(f)
        result.traverse({ tagName, htmlNode ->
            tagName?.attributes?.remove 'class'
            if ('td' == tagName?.name || 'th' == tagName?.name) {
                tagName.name = 'td'
                String txt = tagName.text
                tagName.removeAllChildren()
                tagName.insertChild(0, new ContentNode(txt))
            }
        })
        true
    } as TagNodeVisitor
    serializer.writeToFile(
        result, tmpHtml.absolutePath, "utf-8"
    )
    "pandoc -f html -t asciidoc -R -S --normalize -s $tmpHtml -o ${target}.adoc".execute().waitFor()
    tmpHtml.delete()
}
```

Execute external  
commands

Wait for the end of  
the process



- Groovy's APIs feature
  - an **Ant task scripting** API to reuse all existing Ant tasks
    - ftp, ssh, file copying/moving...
  - a **template engine**
    - for generating files, source code
  - easy creation & parsing of **XML** or **JSON** payloads
  - **Sql** facility to simplifying accessing a **JDBC** compliant relational DB
  - **exposing** and **accessing JMX** beans

# How?

## Scripting: task automation

```
def writer = new StringWriter()
def mkp = new MarkupBuilder(writer)

mkp.html {
    head { title 'Build notification' }
    body { p 'Your build was successful' }
}

new AntBuilder().mail(mailhost: 'localhost',
    messagemimetype: 'text/html',
    subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to    address: 'dev@mycompany.org'

    message writer

    attachments {
        fileset dir: 'dist', {
            include name: '**/logs*.txt'
        }
    }
}
```



```
def writer = new StringWriter()
def mkp = new MarkupBuilder(writer)

mkp.html {
    head { title 'Build notification' }
    body { p 'Your build was successful' }
}

new AntBuilder().mail(mailhost: 'localhost',
    messagemimetype: 'text/html',
    subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to address: 'dev@mycompany.org'
```



```
def writer = new StringWriter()
def mkp = new MarkupBuilder(writer)

mkp.html {
    head { title 'Build notification' }
    body { p 'Your build was successful' }
}

new AntBuilder(
    message: 'Build notification',
    subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to address: 'dev@mycompany.org'
```

Use the Groovy Markup  
Builder class to generate  
HTML or XML payloads



```
def writer = new StringWriter()
def mkp = new MarkupBuilder(writer)

mkp.html {
    head { title 'Build notification' }
    body { p 'Your build was successful' }
}

new AntBuilder().mail(mailhost: 'localhost',
    messagemimetype: 'text/html',
    subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to address: 'dev@mycompany.org'
```



```
new AntBuilder().mail(mailhost: 'localhost',
    messagemimetype: 'text/html',
        subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to    address: 'dev@mycompany.org'

    message writer

    attachments {
        fileset dir: 'dist', {
            include name: '**/logs*.txt'
        }
    }
}
```



```
new AntBuilder().mail(mailhost: 'localhost',  
    messagemimetype: 'text/html',  
        subject: 'Build notification') {  
  
    from address: 'ci@mycompany.org'  
    to    address: 'dev@mycompany.org'  
  
    message writer  
  
    attachments {  
        fileset dir: 'dist',  
            include name: '**/logs*.txt'  
    }  
}
```

Use the Ant Builder, reusing  
the Mail Ant task to send a  
build notification



```
new AntBuilder().mail(mailhost: 'localhost',
    messagemimetype: 'text/html',
        subject: 'Build notification') {

    from address: 'ci@mycompany.org'
    to    address: 'dev@mycompany.org'

    message writer

    attachments {
        fileset dir: 'dist', {
            include name: '**/logs*.txt'
        }
    }
}
```



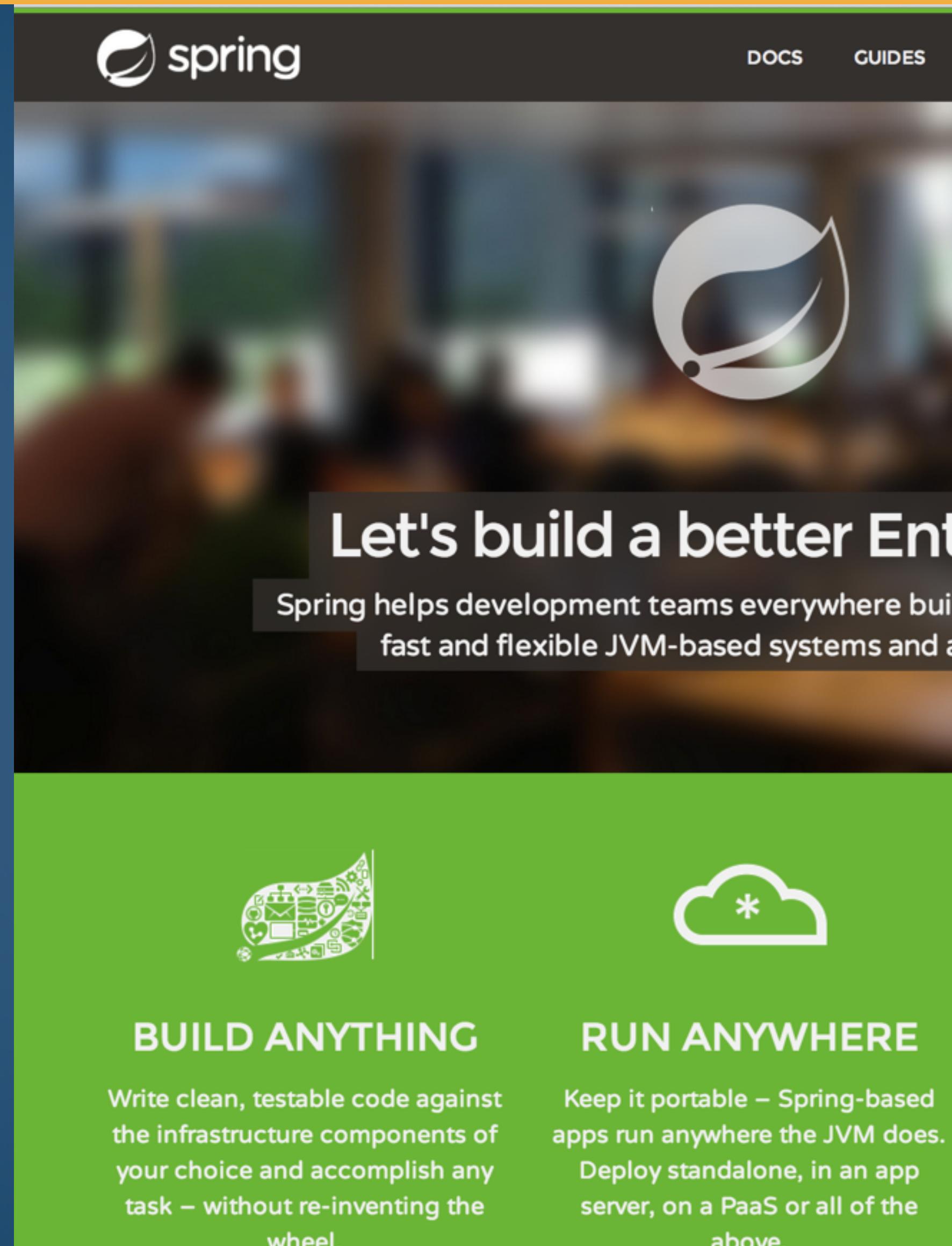
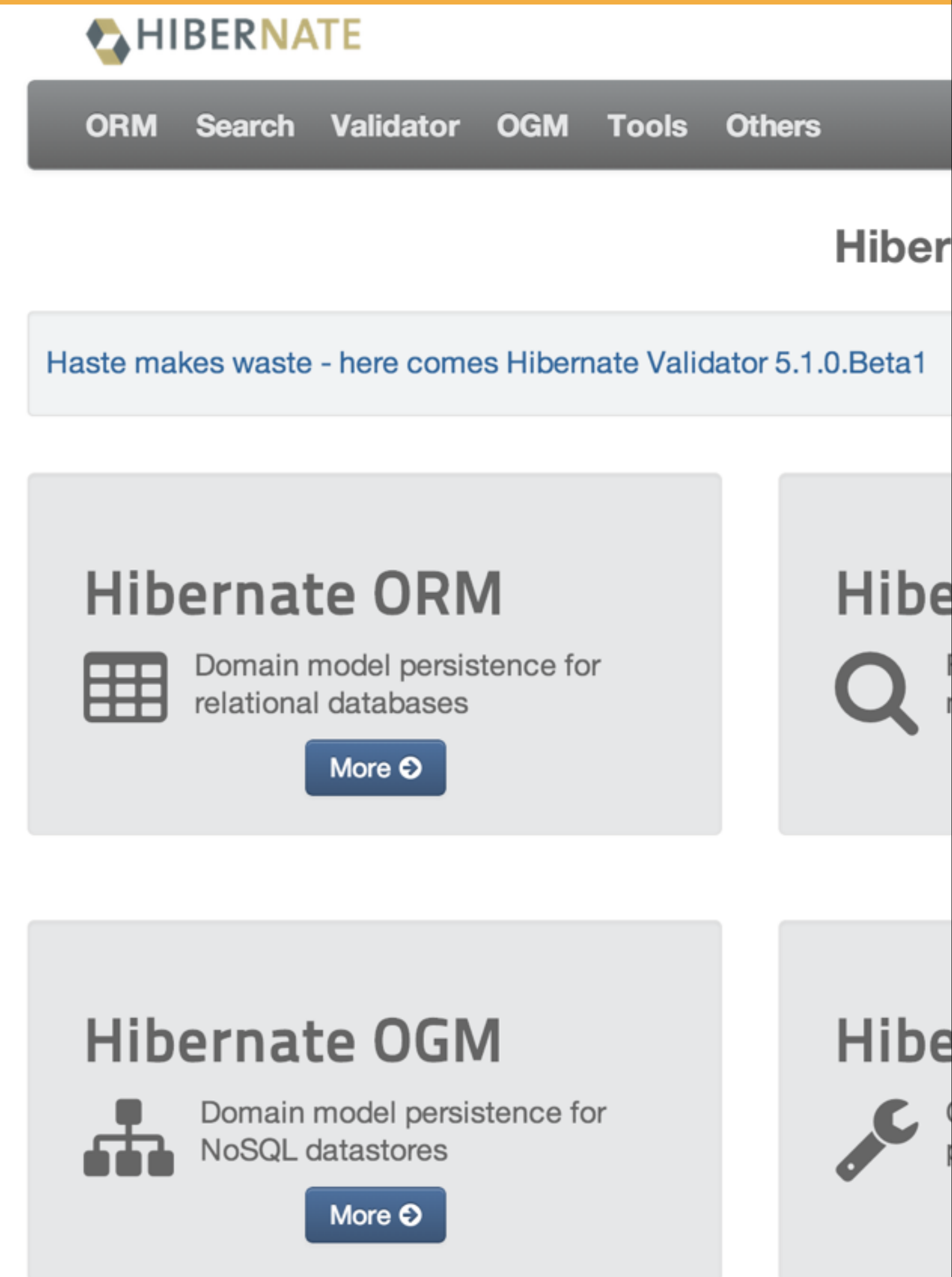
- **Gradle** is a powerful build automation solution
  - uses **Groovy** for its automation language
- You can
  - build / package / deploy Java, Groovy, Scala, C/C++ projects
  - handle dependency management cleanly
  - define and enforce your own enterprise build conventions
  - extend Gradle declaratively and programmatically





# Who?

# Scripting: build automation

The Spring website banner. At the top is the Spring logo and navigation links for "DOCS" and "GUIDES". The main section has a blurred background image of a crowd and a large white leaf icon. The text reads: "Let's build a better Enterprise" and "Spring helps development teams everywhere build fast and flexible JVM-based systems and applications". Below this are two green boxes: "BUILD ANYTHING" with a gear icon and "RUN ANYWHERE" with a cloud icon. The "BUILD ANYTHING" box says: "Write clean, testable code against the infrastructure components of your choice and accomplish any task – without re-inventing the wheel." The "RUN ANYWHERE" box says: "Keep it portable – Spring-based apps run anywhere the JVM does. Deploy standalone, in an app server, on a PaaS or all of the above."The Hibernate website banner. At the top is the Hibernate logo and navigation links for "ORM", "Search", "Validator", "OGM", "Tools", and "Others". Below this is a blue link: "Haste makes waste - here comes Hibernate Validator 5.1.0.Beta1". The main section features two grey boxes: "Hibernate ORM" with a grid icon and "Domain model persistence for relational databases", and "Hibernate OGM" with a tree icon and "Domain model persistence for NoSQL datastores". Both boxes have a "More" button with an external link icon.



## How?

## Scripting: build automation

```
apply plugin: 'java'

repositories {
    mavenCentral()
}

dependencies {
    compile 'commons-collections:commons-collections:3.2'
    testCompile 'junit:junit:4.+'
}
```



# How?

## Scripting: build automation

```
apply plugin: 'java'
```

Use the 'Java' plugin

```
repositories {  
    mavenCentral()  
}
```

```
dependencies {  
    compile 'commons-collections:commons-collections:3.2'  
    testCompile 'junit:junit:4.+'  
}
```



# How?

## Scripting: build automation

```
apply plugin: 'java'
```

Use the 'Java' plugin

```
repositories {  
    mavenCentral()  
}
```

Fetch your dependencies  
from Maven Central

```
dependencies {  
    compile 'commons-collections:commons-collections:3.2'  
    testCompile 'junit:junit:4.+'  
}
```



# How?

## Scripting: build automation

```
apply plugin: 'java'
```

Use the 'Java' plugin

```
repositories {  
    mavenCentral()  
}
```

Fetch your dependencies  
from Maven Central

```
dependencies {  
    compile 'commons-collections:commons-collections:3.2'  
    testCompile 'junit:junit:4.+'  
}
```

Declare your dependencies and their scope



# How?

## Scripting: build automation



```
buildscript {
    repositories { mavenCentral() }
    dependencies {
        classpath 'com.android.tools.build:gradle:0.7.0'
    }
}

apply plugin: 'android'

android {
    compileSdkVersion 19
    buildToolsVersion "19.0.0"
    // ...
    flavorGroups "abi", "version"

    productFlavors {
        freeapp {
            flavorGroup "version"
            // ...
        }
        x86 {
            flavorGroup "abi"
            // ...
        }
    }
}
```





```
buildscript {  
    repositories { mavenCentral() }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:0.7.0'  
    }  
}  
  
apply plugin: 'android'  
  
android {  
    compileSdkVersion 19  
    buildToolsVersion "19.0.0"  
    // ...  
    flavorGroups "abi", "version"  
  
    productFlavors {  
        freeapp {  
            flavorGroup "version"  
            // ...  
        }  
    }  
}
```





```
buildscript {  
    repositories { mavenCentral() }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:0.7.0'  
    }  
}  
  
apply plugin: 'android'  
  
android {  
    compileSdkVersion 19  
    buildToolsVersion "19.0.0"  
    // ...  
    flavorGroups "abi", "version"  
  
    productFlavors {  
        freeapp {  
            flavorGroup "version"  
            // ...  
        }  
    }  
}
```

Declare and use the  
'android' plugin





```
,  
apply plugin: 'android'  
  
android {  
    compileSdkVersion 19  
    buildToolsVersion "19.0.0"  
    // ...  
    flavorGroups "abi", "version"  
  
    productFlavors {  
        freeapp {  
            flavorGroup "version"  
            // ...  
        }  
        x86 {  
            flavorGroup "abi"  
            // ...  
        }  
    }  
}
```





```
    },
    apply plugin: 'android'

    android {
        compileSdkVersion 19
        buildToolsVersion "19.0.0"
        // ...
        flavorGroups "abi", "version"

        productFlavors {
            freeapp {
                flavorGroup "version"
                // ...
            }
            x86 {
                flavorGroup "abi"
                // ...
            }
        }
    }
}
```

Gradle plugins can  
create their own  
declarative mini-  
language



- More Groovy-based initiatives around the DevOps theme
- Goals
  - handle Amazon **AWS** configuration, provisioning and **deployments**
  - talking with **remote servers through SSH**, and automating server configuration, working with scp and more



- Learn more:
  - <http://slideshare.net/aestasisit/groovy-dev-ops-in-the-cloud>
  - sshoogr: a Groovy-based DSL for working with remote SSH servers
  - grammazon: a Groovy library and Gradle plugin for working with Amazon EC2 instances
- <https://github.com/danveloper/provisioning-gradle-plugin>
- a Gradle plugin for driving server provisioning through configuration



**Who?**

**Scripting:** DevOps





# Who?

## Scripting: DevOps

Anyone who needs to  
automate cloud  
deployments





# Who?

## Scripting: DevOps

sshogr & gramazon  
are from



**aestas/IT**

Anyone who needs to  
automate cloud  
deployments



**amazon**  
**web services™**



# How?

## Scripting: DevOps with sshoogr

```
@Grab('com.aestasit.infrastructure.sshoogr:sshogr:0.9.15')
import static com.aestasit.ssh.DefaultSsh.*

remoteSession('user2:654321@localhost:2222') {
    exec 'rm -rf /tmp/*'
    exec 'touch /var/lock/my.pid'

    remoteFile('/var/my.conf').text = "enabled=true"

    //...
    scp {
        from { localDir "./target/application" }
        into { remoteDir '/var/bea/domain/application' }
    }
}
```



# How?

## Scripting: DevOps with sshoogr

```
@Grab('com.aestasit.infrastructure.sshoogr:sshogr:0.9.15')
import static com.aestasit.ssh.DefaultSsh.*

remoteSession('user2:654321@localhost:2222') {
    exec 'rm -rf /tmp/*'
    exec 'touch /var/lock/my.pid'

    remoteFile('/var/my.conf').text = "enabled=true"

    //...
    scp {
        from { localDir "./target/application" }
        into { remoteDir '/var/bea/domain/application' }
    }
}
```

Remote execution



# How?

## Scripting: DevOps with sshoogr

```
@Grab('com.aestasit.infrastructure.sshoogr:sshogr:0.9.15')
import static com.aestasit.ssh.DefaultSsh.*

remoteSession('user2:654321@localhost:2222') {
    exec 'rm -rf /tmp/*'
    exec 'touch /var/lock/my.pid'

    remoteFile('/var/my.conf').text = "enabled=true"

    //...
    scp {
        from { localDir './target/application' }
        into { remoteDir '/var/bea/domain/application' }
    }
}
```

Remote execution

SSH Secure copy



## How?

## Scripting: DevOps with gramazon Gradle plugin

```
task startInstance(type: StartInstance) {  
    keyName           'cloud-do'  
    securityGroup     'cloud-do'  
    instanceName      'gramazon/cloud-do'  
    stateFileName     'cloud-do.json'  
    ami               'ami-6f07e418'  
    instanceType      't1.micro'  
    waitForStart      true  
}
```



## How?

## Scripting: DevOps with gramazon Gradle plugin

```
task startInstance(type: StartInstance) {  
    keyName           'cloud-do'  
    securityGroup     'cloud-do'  
    instanceName      'gramazon/cloud-do'  
    stateFileName     'cloud-do.json'  
    ami               'ami-6f07e418'  
    instanceType      't1.micro'  
    waitForStart      true  
}
```

Start EC2 instances from your  
Gradle automation build



# How?

## Scripting: DevOps

<https://github.com/aestasisit>



**sshogr**

**p-unit**



**gramazon**



**imgr**



## How?

# Scripting: DevOps with Provisioning Gradle plugin

```
import static gradle.plugins.provisioning.types.BootProto.*

apply plugin: 'provisioning'

version = "1.0.RELEASE"

buildscript {
    repositories {
        mavenLocal()
        mavenCentral()
    }
    dependencies {
        classpath 'com.danveloper:provisioning-gradle-plugin:0.1-SNAPSHOT'
    }
}
```



## How?

# Scripting: DevOps with Provisioning Gradle plugin

```
import static gradle.plugins.provisioning.types.BootProto.*  
  
apply plugin: 'provisioning'  
  
version = "1.0.RELEASE"  
  
buildscript {  
    repositories {  
        mavenLocal()  
        mavenCentral()  
    }  
    dependencies {  
        classpath 'com.danveloper:provisioning-gradle-plugin:0.1-SNAPSHOT'  
    }  
}
```

Import and apply the  
Provisioning plugin for Gradle



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
provisioning {
    installImage = "http://www.gtlib.gatech.edu/pub/centos/6.4/isos/x86_64/" +
                  "CentOS-6.4-x86_64-netinstall.iso"

    // generated with "grub-crypt"
    rootpw      = '$6$M2N0GvDMV.hro4Nj$6/4W1SmGuWs8fscbdNLfp4fGFpEt93Y7kCN' +
                  'i8jnjN5JIkPy8YJGkkjCwImyXtCiheMyAkUR24IPgcrfeIliB7/'

    vbox {
        apiUrl = "http://localhost:18083"
        name    = "web"
        x64      = true
        memory   = 1024
        disk     = 8589934592
    }

    network {
        device("eth0") {
            bootproto = DHCP
            onboot     = true
            ipv6       = false
        }
    }
}
```



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
provisioning {
    installImage = "http://www.gtlib.gatech.edu/pub/centos/6.4/isos/x86_64/" +
                  "CentOS-6.4-x86_64-netinstall.iso"

    // generated with "grub-crypt"
    rootpw      = '$6$M2N0GvDMV.hro4Nj$6/4W1SmGuWs8fscbdNLfp4fGFpEt93Y7kCN' +
                  'i8jnjN5JIkPy8YJGkkjCwImyXtCiheMyAkUR24IPgcrfeIliB7/'

    vbox {
        apiUrl = "http://localhost:18083"
        name   = "web"
        x64    = true
        memory = 1024
        disk   = 8589934592
    }

    network {
        device("eth0") {
            bootproto = DHCP
            onboot     = true
            ipv6       = false
        }
    }
}
```

Virtual box configuration



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
provisioning {
    installImage = "http://www.gtlib.gatech.edu/pub/centos/6.4/isos/x86_64/" +
                  "CentOS-6.4-x86_64-netinstall.iso"

    // generated with "grub-crypt"
    rootpw      = '$6$M2N0GvDMV.hro4Nj$6/4W1SmGuWs8fscbdNLfp4fGFpEt93Y7kCN' +
                  'i8jnjN5JIkPy8YJGkkjCwImyXtCiheMyAkUR24IPgcrfeIliB7/'

    vbox {
        apiUrl = "http://localhost:18083"
        name   = "web"
        x64     = true
        memory = 1024
        disk    = 8589934592
    }

    network {
        device("eth0") {
            bootproto = DHCP
            onboot     = true
            ipv6       = false
        }
    }
}
```

Virtual box configuration

Network configuration



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
partitioning {  
    clear(init: true)  
  
    part {  
        mntpoint = "/"  
        fstype    = "ext4"  
        size      = 1  
        grow      = true  
    }  
  
    part {  
        mntpoint | = "swap"  
        recommended = true  
    }  
}
```



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
partitioning {  
    clear(init: true)  
  
    part {  
        mntpoint = "/"  
        fstype    = "ext4"  
        size      = 1  
        grow      = true  
    }  
  
    part {  
        mntpoint | = "swap"  
        recommended = true  
    }  
}
```

Disk partitioning definition



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
packages {  
    // or, perhaps more preferably, a network-local repo  
    url "http://www.gtlib.gatech.edu/pub/centos/6.4/os/x86_64/"  
  
    // some other repo  
    repo("extra") {  
        "http://192.168.0.106/project-repo"  
    }  
  
    // kickstart package groups  
    group "base"  
    group "core"  
    group "console-internet"  
    group "server-platform"  
  
    // These packages come from the "extra" repo  
    pkg "jdk"  
    pkg "apache-tomcat"  
    pkg "hello-webapp"  
}
```



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
packages {  
    // or, perhaps more preferably, a network-local repo  
    url "http://www.gtlib.gatech.edu/pub/centos/6.4/os/x86_64/"  
  
    // some other repo  
    repo("extra") {  
        "http://192.168.0.106/project-repo"  
    }  
  
    // kickstart package groups  
    group "base"  
    group "core"  
    group "console-internet"  
    group "server-platform"  
  
    // These packages come from the "extra" repo  
    pkg "jdk"  
    pkg "apache-tomcat"  
    pkg "hello-webapp"  
}
```

Packages to be installed



# How?

## Scripting: DevOps with Provisioning Gradle plugin

```
postInstall {  
    '''\  
        |echo "export JAVA_HOME=/usr/java/latest" >> /etc/profile.d/java.sh  
        |rm /usr/bin/java && ln -s /usr/java/latest/bin/java /usr/bin/java  
    '''  
    .stripMargin()  
}
```



# How?

## Scripting: DevOps with Provisioning Gradle plugin

### Post installation commands

```
postInstall {  
    '''\  
    |echo "export JAVA_HOME=/usr/java/latest" >> /etc/profile.d/java.sh  
    |rm /usr/bin/java && ln -s /usr/java/latest/bin/java /usr/bin/java  
    '''  
    .stripMargin()  
}
```



- Checkout the Provisioning Gradle plugin from Dan Woods:
  - <https://github.com/danveloper/provisioning-gradle-plugin>



Case 2

# Testing

---

More readable &  
expressive tests





## What for?

## Testing: More readable & expressive tests

- Groovy has a **malleable** and **streamlined syntax**
- Makes it nice for creating libraries and frameworks with **readable** and **expressive APIs**



- Many companies started integrating Groovy in their projects through testing, before any line of Groovy code in production
- Benefit
  - easier to add, evolve and maintain test cases



- We'll have a look at:
  - the Spock testing framework
  - the Geb web integration testing framework









I'm Spock...





I'm Spock...

...the Spock testing  
framework





I'm Spock...

...the Spock testing  
framework



# How?

## Testing: Spock test framework

```
@Grab('org.spockframework:spock-core:0.7-groovy-2.0')  
import spock.lang.*
```

```
class MathSpec extends Specification {  
    def "maximum of two numbers"() {  
        expect:  
            Math.max(a, b) == c
```

```
        where:
```

a		b			c
1		3			3
7		4			7
0		0			0

```
    }
```

```
}
```



# How?

## Testing: Spock test framework

```
@Grab('org.spockframework:spock-core:0.7-groovy-2.0')  
import spock.lang.*
```

```
class MathSpec extends Specification {  
    def "maximum of two numbers"() {  
        expect:  
            Math.max(a, b) == c
```

```
        where:
```

a	b	c
1	3	3
7	4	7
0	0	0

```
    }
```

```
}
```

Wiki-like notation for  
defining data driven tests



# How?

## Testing: Spock test framework

```
@Grab('org.spockframework:spock-core:0.7-groovy-2.0')
import spock.lang.*

class PublisherSpec extends Specification {
    Publisher publisher = new Publisher()
    Subscriber subscriber = Mock()
    Subscriber subscriber2 = Mock()

    def setup() {
        publisher.subscribers << subscriber
        publisher.subscribers << subscriber2
    }

    def "should send messages to all subscribers"() {
        when:
            publisher.send("hello")

        then:
            1 * subscriber.receive("hello")
            1 * subscriber2.receive("hello")
    }
}
```



```
def setup() {  
    publisher.subscribers << subscriber  
    publisher.subscribers << subscriber2  
}  
  
def "should send messages to all subscriber  
when:  
    publisher.send("hello")  
  
then:  
    1 * subscriber.receive("hello")  
    1 * subscriber2.receive("hello")  
}
```



```
def setup() {  
  publisher.subscribers << subscriber  
  publisher.subscribers << subscriber2  
}  
  
def "should send messages to all subscriber"  
when:  
  publisher.send("hello")  
  
then:  
  1 * subscriber.receive("hello")  
  1 * subscriber2.receive("hello")  
}
```

Define the mock  
interactions



```
def setup() {  
  publisher.subscribers << subscriber  
  publisher.subscribers << subscriber2  
}  
  
def "should send messages to subscribers"  
  when:  
    publisher.send("hello")  
  
  then:  
    1 * subscriber.receive("hello")  
    1 * subscriber2.receive("hello")  
}
```

Define the mock  
interactions

Possible to define  
cardinality, matching any  
number of parameter, or  
arbitrary method name



```
1 * subscriber.receive("hello") // exactly one call
0 * subscriber.receive("hello") // zero calls
(1..3) * subscriber.receive("hello") // between one and three calls (inclusive)
(1.._) * subscriber.receive("hello") // at least one call
(_..3) * subscriber.receive("hello") // at most three calls
_ * subscriber.receive("hello") // any number of calls, including zero
// (rarely needed; see 'Strict Mocking')

1 * subscriber.receive("hello") // a call to 'subscriber'
1 * _.receive("hello") // a call to any mock object
1 * subscriber.receive("hello") // a method named 'receive'
1 * subscriber./r.*e/("hello") // a method whose name matches the given regular exp
// (here: method name starts with 'r' and ends in 'e')

1 * subscriber.receive("hello") // an argument that is equal to the String "hello"
1 * subscriber.receive(!"hello") // an argument that is unequal to the String "hello"
1 * subscriber.receive() // the empty argument list (would never match in our
1 * subscriber.receive(_) // any single argument (including null)
1 * subscriber.receive(*_) // any argument list (including the empty argument l
1 * subscriber.receive(!null) // any non-null argument
1 * subscriber.receive(_ as String) // any non-null argument that is-a String
1 * subscriber.receive({ it.size() > 3 }) // an argument that satisfies the given predica
// (here: message length is greater than 3)

1 * subscriber._(*_) // any method on subscriber, with any argument list
1 * subscriber._ // shortcut for and preferred over the above

1 * _._ // any method call on any mock object
1 * _ // shortcut for and preferred over the above
```



# How?

## Testing: Geb web integration tests

```
import geb.Browser

Browser.drive {
  go "http://myapp.com/login"

  assert $("h1").text() == "Please Login"

  $("form.login").with {
    username = "admin"
    password = "password"
    login().click()
  }

  assert $("h1").text() == "Admin Section"
}
```



# How?

## Testing: Geb web integration

An HtmlUnit, FireFox or Chrome browser

```
import geb.Browser

Browser.drive {
  go "http://myapp.com/login"

  assert $("h1").text() == "Please Login"

  $("form.login").with {
    username = "admin"
    password = "password"
    login().click()
  }

  assert $("h1").text() == "Admin Section"
}
```



# How?

## Testing: Geb web integration

An HtmlUnit, FireFox or Chrome browser

```
import geb.Browser
```

```
Browser.drive {
```

```
  go "http://myapp.com/login"
```

```
  assert $("h1").text() == "Please Log
```

Drive the browser to a certain URL

```
  $("form.login").with {
```

```
    username = "admin"
```

```
    password = "password"
```

```
    login().click()
```

```
  }
```

```
  assert $("h1").text() == "Admin Section"
```

```
}
```



# How?

## Testing: Geb web integration

An HtmlUnit, FireFox or Chrome browser

```
import geb.Browser
```

```
Browser.drive {  
  go "http://myapp.com/login"
```

Drive the browser to a certain URL

Assert the title contains the text

```
  assert $("h1").text() == "Please Log
```

```
  $("form.login").with {  
    username = "admin"  
    password = "password"  
    login().click()  
  }
```

```
  assert $("h1").text() == "Admin Section"
```

```
}
```



# How?

## Testing: Geb web integration

An HtmlUnit, FireFox or Chrome browser

```
import geb.Browser
```

```
Browser.drive {  
  go "http://myapp.com/login"
```

Drive the browser to a certain URL

```
  assert $("h1").text() == "Please Log
```

Assert the title contains the text

```
  $("form.login").with {  
    username = "admin"  
    password = "password"  
    login().click()  
  }
```

Fill in a form, click the button to send it

```
  assert $("h1").text() == "Admin Section"
```

```
}
```



# How?

## Testing: Geb web integration tests with Spock

```
import geb.spock.GebSpec

class GoogleWikipediaSpec extends GebSpec {

    def "first result for wikipedia search should be wikipedia"() {
        given: to GoogleHomePage
        expect: at GoogleHomePage

        when: search.field.value("wikipedia")
        then: waitFor { at GoogleResultsPage }
        and: firstResultLink.text() == "Wikipedia"

        when: firstResultLink.click()
        then: waitFor { at WikipediaPage }
    }
}
```



# How?

## Testing: Geb web integration tests with Spock

```
import geb.spock.GebSpec

class GoogleWikipediaSpec extends GebSpec {

    def "first result for wikipedia search should be wikipedia"() {
        given: to GoogleHomePage
        expect: at GoogleHomePage

        when: search.field.value("wikipedia")
        then: waitFor { at GoogleResultsPage }
        and: firstResultLink.text() == "Wikipedia"

        when: firstResultLink.click()
        then: waitFor { at WikipediaPage }
    }
}
```

With page objects



# How?

## Testing: Geb web integration tests with Spock

```
import geb.spock.GebSpec

class GoogleWikipediaSpec extends GebSpec {

    def "first result for wikipedia search should be wikipedia"() {
        given: to GoogleHomePage
        expect: at GoogleHomePage

        when: search.field.value("wikipedia")
        then: waitFor { at GoogleResultsPage }
        and: firstResultLink.text() == "Wikipedia"

        when: firstResultLink.click()
        then: waitFor { at WikipediaPage }
    }
}
```

With page objects

BDD style: given/when/then



# How?

## Testing: Geb web integration tests with Spock

```
import geb.spock.GebSpec

class GoogleWikipediaSpec extends GebSpec {

    def "first result for wikipedia search should be wikipedia"() {
        given: to GoogleHomePage
        expect: at GoogleHomePage

        when: search.field.value("wikipedia")
        then: waitFor { at GoogleResultsPage }
        and: firstResultLink.text() == "Wikipedia"

        when: firstResultLink.click()
        then: waitFor { at WikipediaPage }
    }
}
```

With page objects

BDD style: given/when/then

Wait for slow loading pages



Case 3

# Customizing

---

## Extension points Configuration





- You need to...
  - **customize** an application for a particular customer
  - **configure** an application for a particular environment
  - create / update / **externalize business rules**
  - create **plugins** for extending a platform



**Who?**

**Customizing:** extension, configuration, plugins



# Jenkins



# SmartThings



THE BEST WAY TO ORGANIZE INFORMATION



**Who?**

**Customizing:** extension, configuration, plugins



# Jenkins

Continuous Integration server





# Who?

## Customizing: extension, configuration, plugins



# Jenkins

Continuous Integration server



# SmartThings

Internet of Things



THE BEST WAY TO ORGANIZE INFORMATION



# Who?

## Customizing: extension, configuration, plugins



# Jenkins

Continuous Integration server



# SmartThings

Internet of Things



THE BEST WAY TO ORGANIZE INFORMATION

Second-generation wiki





# Jenkins

- Jenkins provides two Groovy plugins:
  - the Groovy plugin  
<https://wiki.jenkins-ci.org/display/JENKINS/Groovy+plugin>
    - execute arbitrary Groovy scripts as Jenkins jobs
  - the Groovy postbuild plugin  
<https://wiki.jenkins-ci.org/display/JENKINS/Groovy+Postbuild+Plugin>
    - set build outcome, display badges, info or error messages, once a build completed



# How?

## Customizing: extension, configuration, plugins

```
if(manager.logContains(".*uses or overrides a deprecated API.*")) {  
    manager.addWarningBadge("Thou shalt not use deprecated methods.")  
    manager.createSummary("warning.gif")  
        .appendText("<h1>You have been warned!</h1>",  
                    false, false, false, "red")  
    manager.buildUnstable()  
}
```

 **Build History** [\(trend\)](#)

	#6	<a href="#">26.01.2010 13:33:13</a>
	#5	<a href="#">26.01.2010 13:32:32</a>
	#4	<a href="#">26.01.2010 13:32:12</a>
	#3	<a href="#">26.01.2010 13:31:53</a>

 [for all](#)  [for failures](#)

### Permalinks

- [Last build \(#5\), 38 sec ago](#)
- [Last stable build \(#5\), 38 sec ago](#)
- [Last build \(#5\), 38 sec ago](#)

Thou shalt not use deprecated methods.



# Jenkins



# How?

## Customizing: extension, configuration, plugins

- Home automation in the era of the Internet of Things



- Built-in IDE and simulator allow you to automate all your devices of your home with Groovy scripts  
<https://support.smarththings.com/entries/21603015-Introduction-to-Writing-SmartApps>
- Groovy supports allows to
  - set preferences, subscribe to events, set timers, handle events, send notifications, access people's presence, consume external services...



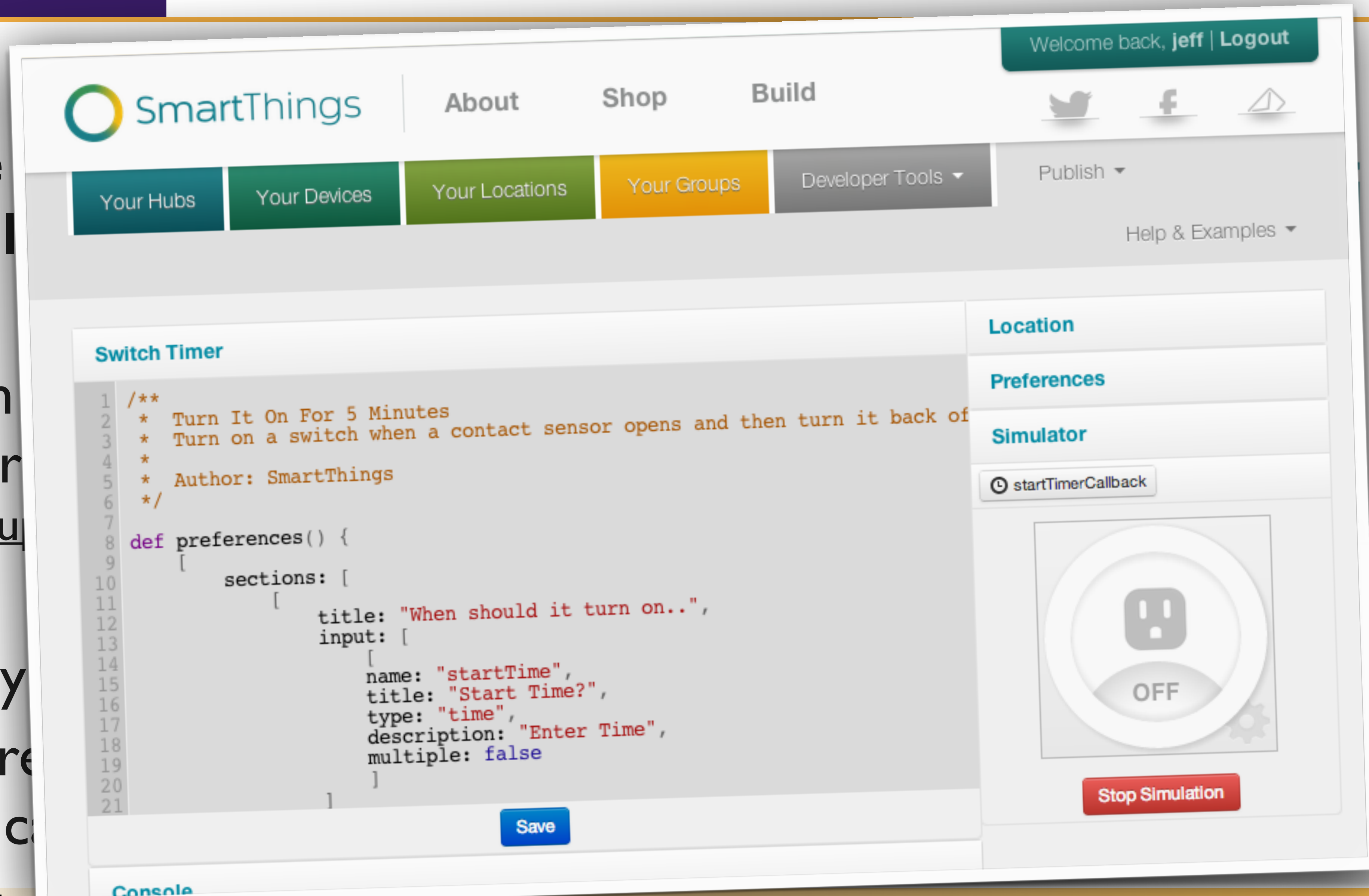
# How?

## Customizing: extension, configuration, plugins

- Home of the I

- Built-in all your <https://su>

- Groovy
  - set pre
  - notific



ngs

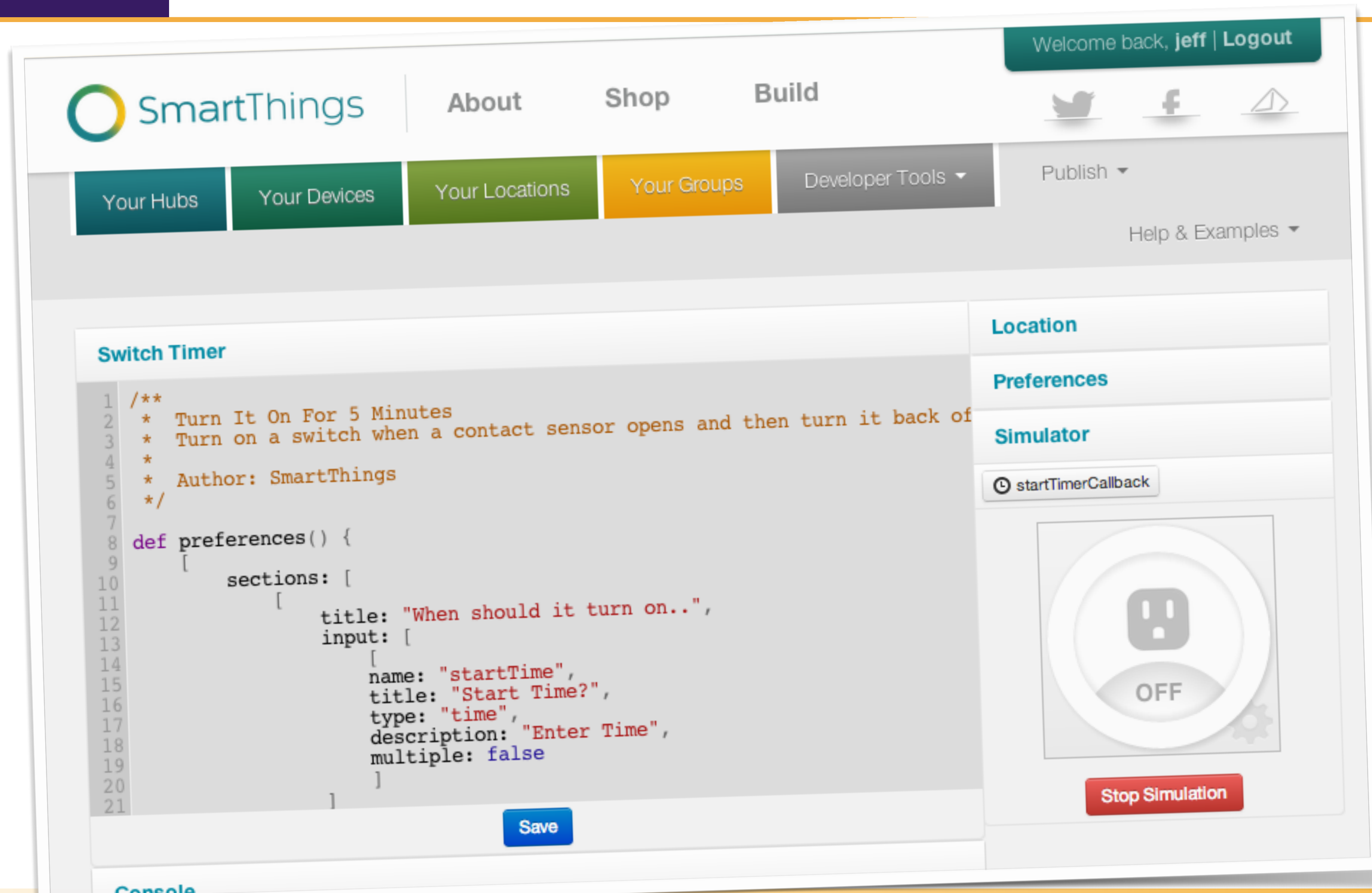
artApps

send  
ices...



# How?

# Customizing: extension, configuration, plugins





# How?

## Customizing: extension, configuration, plugins

- **XWiki** is a second-generation wiki  
<http://www.xwiki.org/>  
(XWiki case study: <http://bit.ly/xwiki-cs>)
  - not just content, but programmable
  - a platform for building your own apps
  - extensible with plugins and macros
- You can create your own plugins and macros in Groovy to extend the wiki, and add dynamic content





# How?

## Customizing: extension, configuration, plugins

```
{{cache}}
{{groovy}}
import groovy.json.*

def url = "https://api.github.com/users/xwiki/repos".toURL().text
def root = new JsonSlurper().parseText(url)

println "|=Project|=Description|=Use Wiki?|=Use Issues?"
root.each() { repo ->
    println "| [[${repo.name}]>>http://github.com/xwiki/${repo.name}]]" +
        "| ${repo.description} | ${repo.has_wiki} | ${repo.has_issues}"
}
{{/groovy}}
{{/cache}}
```



# How?

## Customizing: extension, configuration, plugins

A wiki user with programming rights can add this Groovy script in a page

```
{{cache}}
{{groovy}}
import groovy.json.*

def url = "https://api.github.com/users/xwiki/repos".toURL().text
def root = new JsonSlurper().parseText(url)

println "|=Project|=Description|=Use Wiki?|=Use Issues?"
root.each() { repo ->
    println "| [[${repo.name}]>>http://github.com/xwiki/${repo.name}]]" +
        "| ${repo.description} | ${repo.has_wiki} | ${repo.has_issues}"
}
{{/groovy}}
{{/cache}}
```



# How?

## Customizing: extension, configuration, plugins

A wiki user with programming rights can add this Groovy script in a page

```
{{cache}}
{{groovy}}
import groovy.json.*

def url = "https://api.github.com/users/xwiki/repos".toURL().text
def root = new JsonSlurper().parseText(url)

println "|=Project|=Description|=Use Wiki?|=Use Issues"
root.each() { repo ->
    println "| [[${repo.name}]>>http://github.com/xwiki/${repo.name}]|" +
        "| ${repo.description} | ${repo.has_wiki} | ${repo.has_issues}|"
}
{{/groovy}}
{{/cache}}
```

Uses Groovy's handy JSON support



# How?

## Customizing: extension, configuration, plugins

```
{{cache}}
{{groovy}}
import groovy

def url = "ht
def root = new

println "|=Pro
root.each() {
    println "|
}
{{/groovy}}
{{/cache}}
```

### List of Git repositories for the XWiki organization on GitHub

Last modified by Administrator on 2011/04/05 13:47

Comments (0) · Annotations (0) · Attachments (1) · History · Information

Project	Description	Use Wiki?	Use Issues?
xwiki-rendering	Rendering Engine	false	false
xwiki-commons-core		false	false
xwiki-commons-pom		false	false
xwiki-commons-tools		false	false
xwiki-commons		false	false
xwiki-manager		false	false
xwiki-enterprise		false	false
xwiki-platform-core		false	false
xwiki-platform-web		false	false
xwiki-extension-blog	The Blog application	false	false
xwiki-extension-activestream	The activity stream plugin	false	false
xwiki-extension-administration	The administration application	false	false
xwiki-extension-adwords	Google adwords integration plugin	false	false
xwiki-extension-alexa	Plugin using Alexa webservices to find out information about URLs	false	false
xwiki-extension-annotations	Annotations application	false	false
xwiki-extension-application-manager	XEM application manager	false	false

).text

ovy's handy  
support

.name}]]" +  
\$issues}"



- What Groovy also brings to the table:
  - a rich **API** for integrating and embedding Groovy in your app
    - GroovyShell, Binding, customizers...
  - dedicated **configuration API** (ConfigSlurper) with a special « builder » syntax with a hierarchical data structure
  - a **JMX bean & JMX builder** for interacting with and exposing JMX services



Case 4

# Domain-Specific Language

---



Expressive  
business rules



## What for?

# Domain-Specific Languages: business rules

- The goal of Groovy Domain-Specific Languages:
  - beyond just integrating Groovy scripts and classes: create a dedicated mini-language modeling your business
  - less technical boilerplate code, focus on the business semantics
  - write almost plain English-like sentences
- Groovy has...
  - a succinct, flexible & malleable syntax
  - both dynamic and compile-time metaprogramming capabilities
  - operator overloading



# Who?

# Domain-Specific Languages: business rules



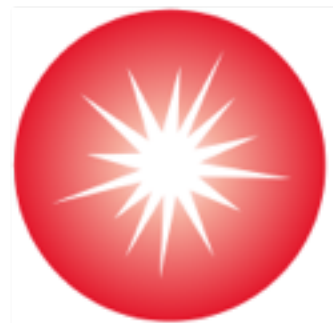
**Mutual of Omaha**



Europäisches  
Patentamt

European  
Patent Office

Office européen  
des brevets



**HYPOPORT**

THE FINANCE INTEGRATOR

**aMADEUS**

Your technology partner



# Who?

## Domain-Specific Languages: business rules



**Mutual of Omaha**



**aMADEUS**

Your technology partner

US Fortune 500 insurance actuaries  
write risk calculation rules in Groovy



# Who?

## Domain-Specific Languages: business rules



**Mutual of Omaha**



**aMADEUS**  
Your technology partner

US Fortune 500 insurance actuaries  
write risk calculation rules in Groovy

EPO built a data flow language atop Groovy to extract /  
transform / route patents across patent offices



# Who?

## Domain-Specific Languages: business rules



**Mutual of Omaha**



**aMADEUS**  
Your technology partner

US Fortune 500 insurance actuaries  
write risk calculation rules in Groovy

EPO built a data flow language atop Groovy to extract /  
transform / route patents across patent offices

1 billion € worth of loan granted through Hypoport's  
financial platform Groovy decision rules



# Who?

## Domain-Specific Languages: business rules



**Mutual of Omaha**

US Fortune 500 insurance actuaries write risk calculation rules in Groovy



EPO built a data flow language atop Groovy to extract / transform / route patents across patent offices



1 billion € worth of loan granted through Hypoport's financial platform Groovy decision rules



Customize experience of travel and hotel reservation with Groovy scripts and templates



# Who?

## Domain-Specific Languages: business rules

Case study:  
<http://bit.ly/epo-cs>



US Fortune 500 insurance actuaries write risk calculation rules in Groovy

EPO built a data flow language atop Groovy to extract / transform / route patents across patent offices

1 billion € worth of loan granted through Hypoport's financial platform Groovy decision rules

Customize experience of travel and hotel reservation with Groovy scripts and templates



# How?

## Domain-Specific Languages: business rules

- Groovy uses **BigDecimal** by default for floating point
- you can, of course, specify floats or doubles if needed
- important for financial calculations w/ exact arithmetics



**Mutual of Omaha**



# How?

## Domain-Specific Languages: business rules

- Groovy uses **BigDecimal** by default for floating point
- you can, of course, specify floats or doubles if needed
- important for financial calculations w/ exact arithmetics

```
BigDecimal uMinusv = upperBound.subtract(value);  
BigDecimal vMinusl = value.subtract(lowerBound);  
BigDecimal uMinusl = upperBound.subtract(lowerBound);  
return lowerValue.multiply(uMinusv).  
    add(upperVal.multiply(vMinusl)).  
    divide(uMinus1, 10, BigDecimal.ROUND_HALF_UP);
```



**Mutual of Omaha**



# How?

## Domain-Specific Languages: business rules

- Groovy uses **BigDecimal** by default for floating point
- you can, of course, specify floats or doubles if needed
- important for financial calculations w/ exact arithmetics



Mutual of Omaha

```
BigDecimal uMinusv = upperBound.subtract(value);  
BigDecimal vMinusl = value.subtract(lowerBound);  
BigDecimal uMinusl = upperBound.subtract(lowerBound);  
return lowerValue.multiply(uMinusv).  
    add(upperVal.multiply(vMinusl)).  
    divide(uMinusl, 10, BigDecimal.ROUND_HALF_UP);
```



```
(lowerVal * (upperBound - value) + upperVal * (value - lowerBound)) /  
(upperBound - lowerBound)
```



# How?

## Domain-Specific Languages: business rules

- Groovy uses **BigDecimal** by default for floating point
- you can, of course, specify floats or doubles if needed
- important for financial calculations w/ exact arithmetics



Mutual of Omaha

```
BigDecimal uMinusv = upperBound.subtract(value);  
BigDecimal vMinusl = value.subtract(lowerBound);  
BigDecimal uMinusl = upperBound.subtract(lowerBound);  
return lowerValue.multiply(uMinusv).  
    add(upperVal.multiply(vMinusl)).  
    divide(uMinusl, 10, BigDecimal.ROUND_HALF_UP);
```



Which formula would  
you rather maintain?



```
(lowerVal * (upperBound - value) + upperVal * (value - lowerBound)) /  
(upperBound - lowerBound)
```



# How?

## Domain-Specific Languages: business rules

- Groovy uses **BigDecimal** by default for floating point
- you can, of course, specify floats or doubles if needed
- important for financial calculations w/ exact arithmetics



Mutual of Omaha

```
BigDecimal uMinusv = upperBound.subtract(value);  
BigDecimal vMinusl = value.subtract(lowerBound);  
BigDecimal uMinusl = upperBound.subtract(lowerBound);  
return lowerValue.multiply(uMinusv).  
    add(upperVal.multiply(vMinusl)).  
    divide(uMinusl, 10, BigDecimal.ROUND_HALF_UP);
```



Which formula would  
you rather maintain?



```
(lowerVal * (upperBound - value) + upperVal * (value - lowerBound)) /  
(upperBound - lowerBound)
```

Operator overloading



# How?

## Domain-Specific Languages: business rules

- Groovy lets you create your own control structures with closures





# How?

## Domain-Specific Languages: business rules

- Groovy lets you create your own control structures with closures



```
given {  
    [person: new Person(age: 100)]  
} when {  
    person.age > 65  
} then {  
    println "Do not grant loan"  
}
```



# How?

## Domain-Specific Languages: business rules

- Groovy lets you create your own control structures with closures



```
given {  
    [person: new Person(age: 100)]  
} when {  
    person.age > 65  
} then {  
    println "Do not grant loan"  
}
```

```
@groovy.transform.Immutable  
class Person {  
    String name  
    int age  
}
```



# How?

## Domain-Specific Languages: business rules

- Groovy lets you create your own control structures with closures



```
given {  
    [person: new Person(age: 100)]  
} when {  
    person.age > 65  
} then {  
    println "Do not grant loan"  
}
```

```
@groovy.transform.Immutable  
class Person {  
    String name  
    int age  
}
```

```
def given(Closure givenCondition) {  
    def dataSetup = givenCondition()  
    [when: { Closure whenCase ->  
        dataSetup.each { String k, v ->  
            whenCase[k] = v  
        }  
        [then: { Closure thenAction ->  
            if (whenCase()) {  
                thenAction()  
            }  
        }]  
    }]  
}
```



# How?

## Domain-Specific Languages: business rules

- A Groovy-based DSL developed at the Liverpool School of Tropical Medicine to study antimalarial drug resistance



UNIVERSITY OF  
LIVERPOOL

```
DHFR = protein("DHFR")
DHFR.mutatingAmino 51, Asn, Ile
DHFR.mutatingAmino 59, Cys, Arg
DHFR.mutatingAmino 108, Asp, Asn

sulfa = compound(name : "Sulfadoxine",
                  abbreviation : "S",
                  halfLife : 116.h)

effect = effect(name : "General effect",
                formula : { 3.45 * MQ**y / (MQ**y + IC50**y) },
                parameters : [IC50 : 665.4.microg / ml, y : 2.44])

SPregimen = regimen()
SPregimen.take 3.pills, of: SP, at: 0.h
SPregimen.take 3.pills, of: SP, at: 1.d
```



# How?

## Domain-Specific Languages: business rules

- A Groovy-based DSL developed at the Liverpool School of Tropical Medicine to study antimalarial drug resistance



UNIVERSITY OF  
LIVERPOOL

```
DHFR = protein("DHFR")
DHFR.mutatingAmino 51, Asn, Ile
DHFR.mutatingAmino 59, Cys, Arg
DHFR.mutatingAmino 108, Asp, Asn

sulfa = compound(name : "Sulfadoxine",
                  abbreviation : "S",
                  halfLife : 116.h)

effect = effect(name : "General effect",
                formula : { 3.45 * MQ**y / (MQ**y + IC50**y) },
                parameters : [IC50 : 665.4.microg / ml, y : 2.44])

SPregimen = regimen()
SPregimen.take 3.pills, of: SP, at: 0.h
SPregimen.take 3.pills, of: SP, at: 1.d
```

Paper:

<http://bit.ly/ronald-dsl>



```

def chloroquine = "chloroquine"
def water = "water"

Integer.metaClass.getPills = { -> delegate }
Integer.metaClass.getHours = { -> delegate }

def take(numOfPills) {
    [of: { drug ->
        [after: { duration ->
            [with: { liquid ->
                println "take $numOfPills pills " +
                    "of $drug " +
                    "after $duration hours " +
                    "with $liquid"
            }]
        }]
    }]
}

```

```
take 3.pills of chloroquine after 6.hours with water
```

```
// take(3.pills).of(chloroquine).after(6.hours).with(water)
```



```
def chloroquine = "chloroquine"  
def water = "water"
```

```
Integer.metaClass.getPills = { -> delegate }  
Integer.metaClass.getHours = { -> delegate }
```

Adding properties to  
numbers through runtime  
meta-programming

```
def take(numOfPills) {  
  [of: { drug ->  
    [after: { duration ->  
      [with: { liquid ->  
        println "take $numOfPills pills " +  
          "of $drug " +  
          "after $duration hours " +  
          "with $liquid"  
      }]  
    }]  
  }]  
}]  
}
```

```
take 3.pills of chloroquine after 6.hours with water
```

```
// take(3.pills).of(chloroquine).after(6.hours).with(water)
```



```
def chloroquine = "chloroquine"  
def water = "water"
```

```
Integer.metaClass.getPills = { -> delegate }  
Integer.metaClass.getHours = { -> delegate }
```

```
def take(numOfPills) {  
  [of: { drug ->  
    [after: { duration ->  
      [with: { liquid ->  
        println "take $numOfPills pills " +  
          "of $drug " +  
          "after $duration hours " +  
          "with $liquid"  
      }]  
    }]  
  }]  
}]  
}
```

Adding properties to  
numbers through runtime  
meta-programming

Groovy « command chains »: syntax  
convention for dropping parens & dots for  
natural-language-like sentences

```
take 3.pills of chloroquine after 6.hours with water
```

```
// take(3.pills).of(chloroquine).after(6.hours).with(water)
```



Case 5

# Full blown apps

---

Web & reactive apps  
Desktop apps





# Groovy ecosystem

---

- The Groovy ecosystem is rich of tools, frameworks, and libraries
- For web development, you can use
  - the mainstream **Grails** web stack, powered by Spring and Groovy
  - the newcomer **Ratpack**, a lightweight toolkit on top of Netty
  - or **Spring Boot**, an opinionated & Groovy-friendly take on Spring
- For desktop applications, you can use **Griffon**
  - which supports different view toolkits such as GroovyFX, Swing, Pivot...



- Grails is a full web stack for the JVM



- based on **Groovy** and **Spring**
- can interact with **relational databases** through **Hibernate** or through any **NoSQL** datastore with **Spring Data**, with the **GORM** mapping layer
- follows the **Convention over Configuration** paradigm
- lets you see changes live with **hot reloading**
- advanced **REST** and **Async** support
- built-in view technology (**GSPs**) with **taglibs**



**Who?**

**Full blown apps:** the Grails web framework

**NETFLIX**

**Linked in**

**sky**

**WIRED**



# Who?

## Full blown apps: the Grails web framework

The Netflix logo, featuring the word "NETFLIX" in white, bold, sans-serif capital letters on a red rectangular background.

Created the Open Source Asgard web interface for application deployments and cloud management in Amazon Web Services

The LinkedIn logo, consisting of the word "Linked" in black sans-serif font followed by a blue square containing the white lowercase letters "in".The Sky logo, featuring the word "sky" in a stylized, lowercase font with a rainbow gradient and a 3D effect.The Wired logo, featuring the word "WIRED" in white, bold, sans-serif capital letters, with each letter contained within its own black square.



# Who?

## Full blown apps: the Grails web framework



NETFLIX

Created the Open Source Asgard web interface for application deployments and cloud management in Amazon Web Services



Linked in

LinkedIn uses Grails for their commercial portals, for recruiters, companies searching for profiles...



sky



WIRED



# Who?

## Full blown apps: the Grails web framework



NETFLIX

Created the Open Source Asgard web interface for application deployments and cloud management in Amazon Web Services



Linked in

LinkedIn uses Grails for their commercial portals, for recruiters, companies searching for profiles...



sky

Many web frontends from the Sky network serving a few hundred millions page views a month



WIRED



# Who?

## Full blown apps: the Grails web framework



NETFLIX

Created the Open Source Asgard web interface for application deployments and cloud management in Amazon Web Services



LinkedIn

LinkedIn uses Grails for their commercial portals, for recruiters, companies searching for profiles...



sky

Many web frontends from the Sky network serving a few hundred millions page views a month



WIRED

Wired's product review section is powered by Grails



# How?

## Full blown apps: the Grails web framework





# How?

## Full blown apps: the Grails web framework

Too hard to cover Grails in a couple slides, just go download it now! :-)





# Summary

---



Groovy's cool :-)



# Java's best friend

---

- Java derived syntax
  - **Flat learning curve**
  - Easy to learn
- But goes beyond Java
  - Concise, expressive, readable
  - Fit for **Domain-Specific Languages**
- **Seamless & transparent Java integration**
  - Mix & match Groovy & Java classes (joint compil.)
  - No language barrier to cross



# Groovy's nature

---

- Object oriented dynamic language...
- But...
  - as **type safe** as you want it — *static type checking*
  - as **fast** as you need it — *static compilation*
  - as **functional** as you make it — *closures...*



# Groovy use cases

---

- Scripting tasks, build automation, DevOps
- Extension points for customizing / configuring apps
- More readable and expressive tests
- Business languages & Domain-Specific Languages
- Full blown apps
  - for the web with Grails, Ratpack, Gaelyk
  - for web reactive programming with Reactor
  - for desktop with Griffon



# Thanks

---

...for your  
attention!





# Q & A

---

Slides will be available at:  
[speakerdeck.com/  
glaforge](https://speakerdeck.com/glaforge)

