Plugging the users in

extend your application with pluggable Groovy DSL

Who's talking?

github.com/jbaruch

@jbaruch

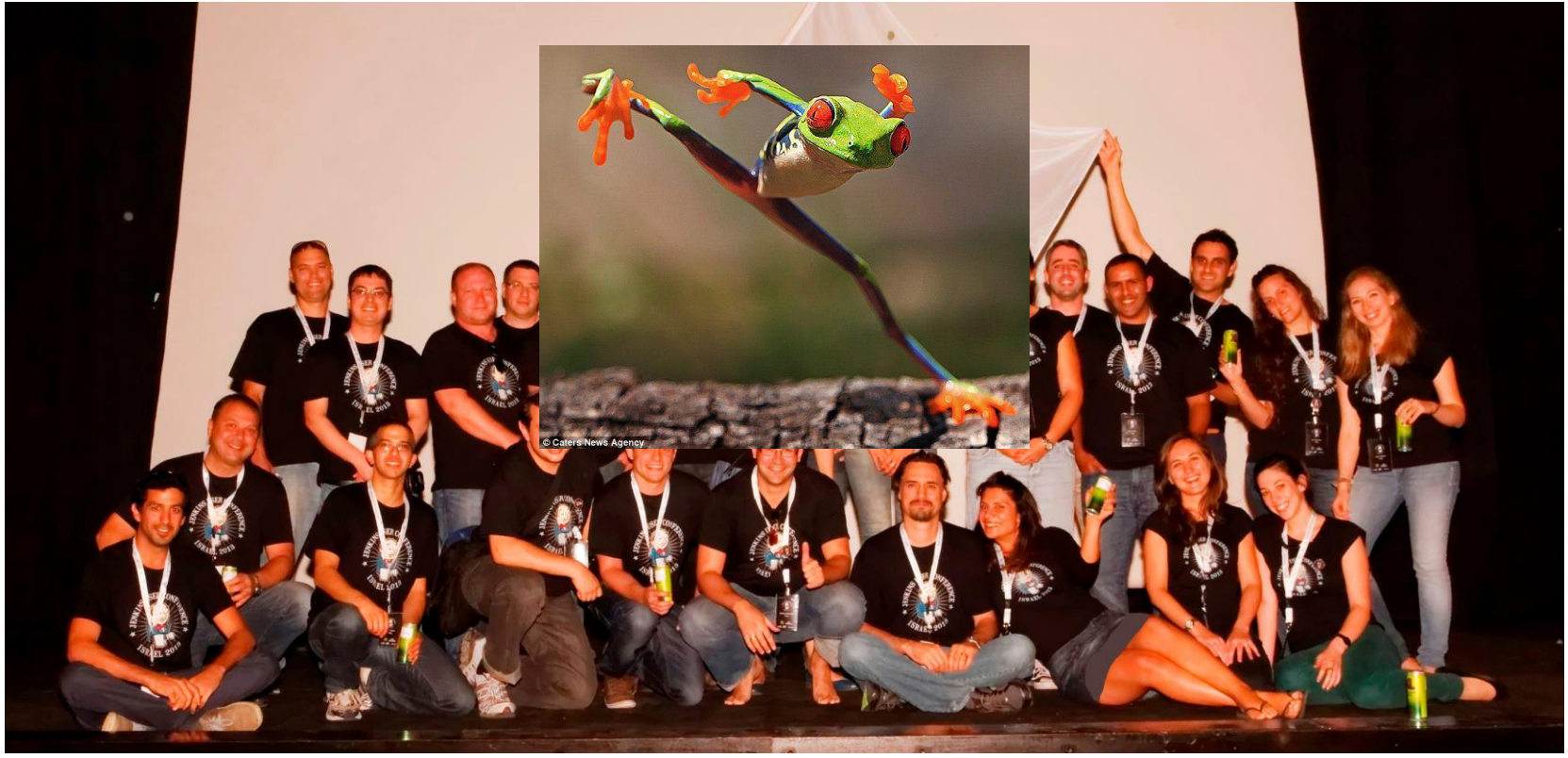
Who's talking?

github.com/freddy33

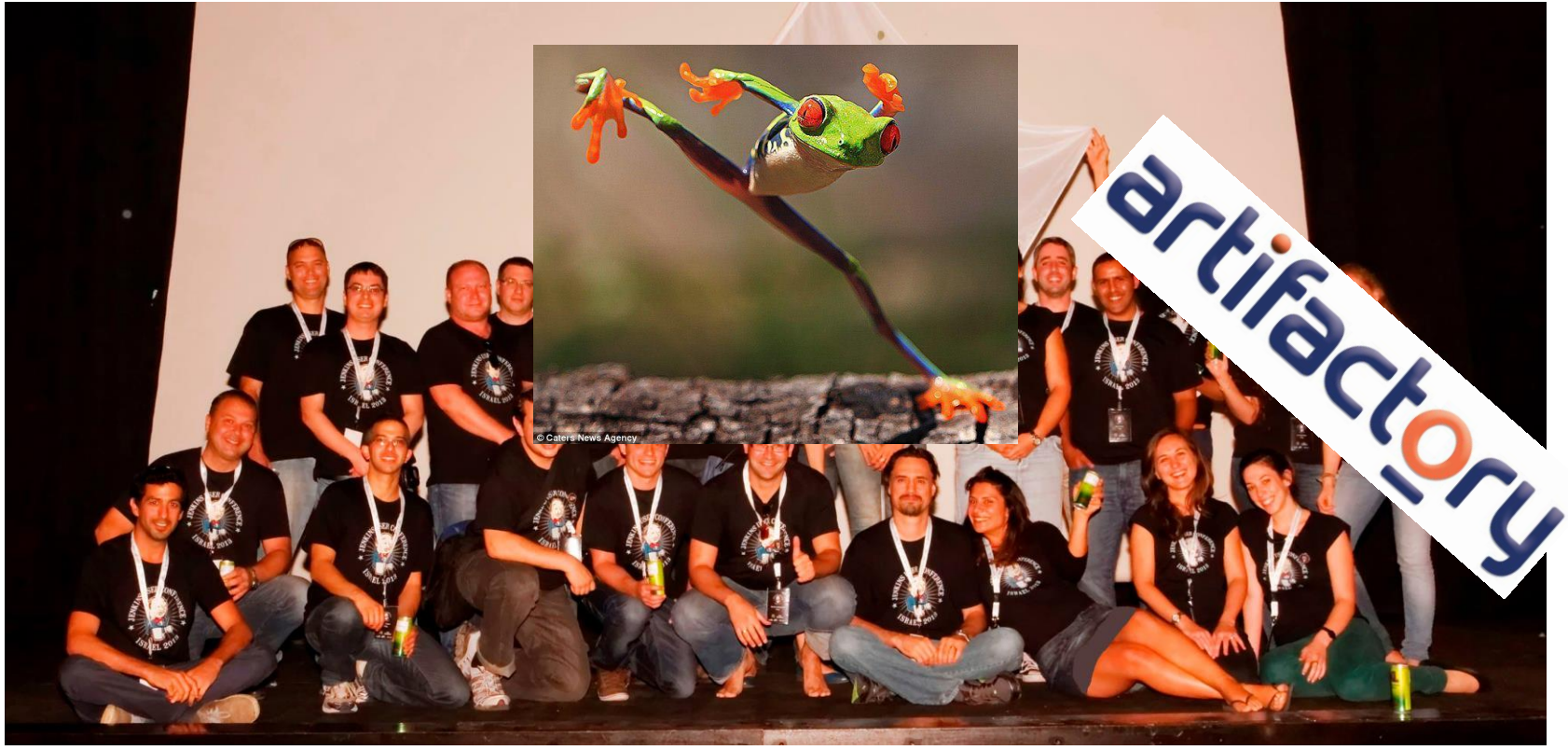
@freddy33

What Frog?

What Frog?



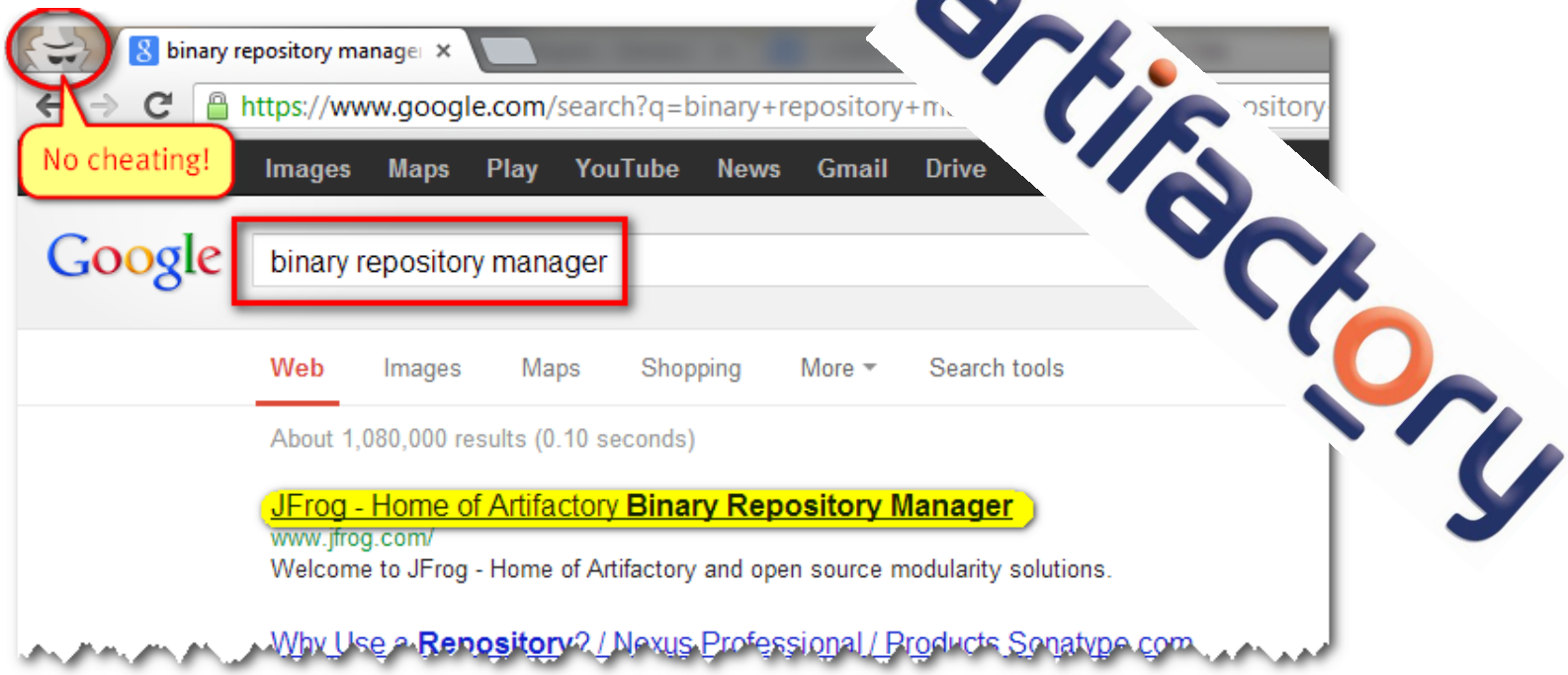
What Frog?



What Frog?


Bintray
once in a 100 years
12/12/12
© Caters News Agency
artifactory

Artifactory what?



Our Imagination is Limited

Pages / ... / Using Artifactory

Artifactory REST API

Added by Yoav Landman, last edited by Shay Yaakov on Aug 29, 2013 (view change) show comment

TABLE OF CONTENTS

- Item Last Modified
- Item Properties
- Set Item Properties
- Delete Item Properties
- Retrieve Artifact
- Retrieve Latest Artifact
- Retrieve Build Artifacts Archive
- Trace Artifact Retrieval
- Archive Entry Download
- Create Directory
- Deploy Artifact
- Deploy Artifact by Checksum
- Deploy Artifacts from Archive
- File Compliance Info
- Delete Item
- Copy Item
- Move Item
- Scheduled Replication Status
- Pull/Push Replication
- Artifact Sync Download (Deprecated)
- Folder Sync (Deprecated)
- File List

- SEARCHES
  - Artifact Search (Quick Search)
  - Archive Entry Search (Class Search)
  - GAVC Search
  - Property Search

- Checksum Search
- Bad Checksum Search
- Artifacts Not Downloaded Since
- Artifacts Created in Date Range
- Pattern Search
- Builds for Dependency
- License Search
- Artifact Version Search
- Artifact Latest Version Search
- Build Artifacts Search

- SECURITY
  - Get Users
  - Get User Details
  - Create or Replace User
  - Update User
  - Delete User
  - Get Groups
  - Get Group Details
  - Create or Replace Group
  - Update Group
  - Delete Group
  - Get Permission Targets
  - Get Permission Target Details
  - Create or Replace Permission Target
  - Delete Permission Target
  - Effective Item Permissions
  - Security Configuration

- Save Security Configuration (Deprecated)
- REPOSITORIES
  - Get Repositories
  - Repository Configuration
  - Create or Replace Repository Configuration
  - Update Repository Configuration
  - Delete Repository
  - Remote Repository Configuration
  - Calculate YUM Repository Metadata
  - Calculate NuGet Repository Metadata
  - Calculate Maven Index
  - Calculate Maven Metadata
- SYSTEM & CONFIGURATION
  - System Info
  - System Health Ping
  - General Configuration
  - Save General Configuration
  - Version and Add-ons information
- PLUGINS
  - Execute Plugin Code
  - Retrieve All Available Plugin Info
  - Retrieve Plugin Info Of A Certain Type
  - Retrieve Build Staging Strategy
  - Execute Build Promotion
- IMPORT & EXPORT
  - Import Repository Content
  - Import System Settings Example

- Full System Import
- Export System Settings Example
- Export System

- Full System Import
- Export System Settings Example
- Export System

This is just a start

Message from our support team:

- SCHEDULED TASKS

- SCHEDULED TASKS
- CUSTOM SECURITY REALMS

- SCHEDULED TASKS
- CUSTOM SECURITY REALMS
- CHANGE RESOLUTION RULES

- SCHEDULED TASKS
- CUSTOM SECURITY REALMS
- CHANGE RESOLUTION RULES
- MANIPULATE DOWNLOADED CONTENT

- SCHEDULED TASKS
- CUSTOM SECURITY REALMS
- CHANGE RESOLUTION RULES
- MANIPULATE DOWNLOADED CONTENT
- LISTENERS ON STORAGE EVENTS

- CUSTOM SECURITY REALMS
- CHANGE RESOLUTION RULES
- MANIPULATE DOWNLOADED CONTENT
- LISTENERS ON STORAGE EVENTS
- SEARCHES

- CHANGE RESOLUTION RULES
- MANIPULATE DOWNLOADED CONTENT
- LISTENERS ON STORAGE EVENTS
- SEARCHES
- NEW REST COMMANDS

- MANIPULATE DOWNLOADED CONTENT
- LISTENERS ON STORAGE EVENTS
- SEARCHES
- NEW REST COMMANDS
- CUSTOM PROMOTIONS

- LISTENERS ON STORAGE EVENTS
- SEARCHES
- NEW REST COMMANDS
- CUSTOM PROMOTIONS
- DEPLOY AND QUERY ARTIFACTS AND METADATA

- ETC.


I WANT YOU
TO CODE YOUR OWN THING

REQUIREMENTS

REQUIREMENTS

- FAMILIAR TO JAVA USERS

REQUIREMENTS

- FAMILIAR TO JAVA USERS
- SIMPLE DSL

REQUIREMENTS

- FAMILIAR TO JAVA USERS
- SIMPLE DSL
- SIMPLE DEPLOYMENT:

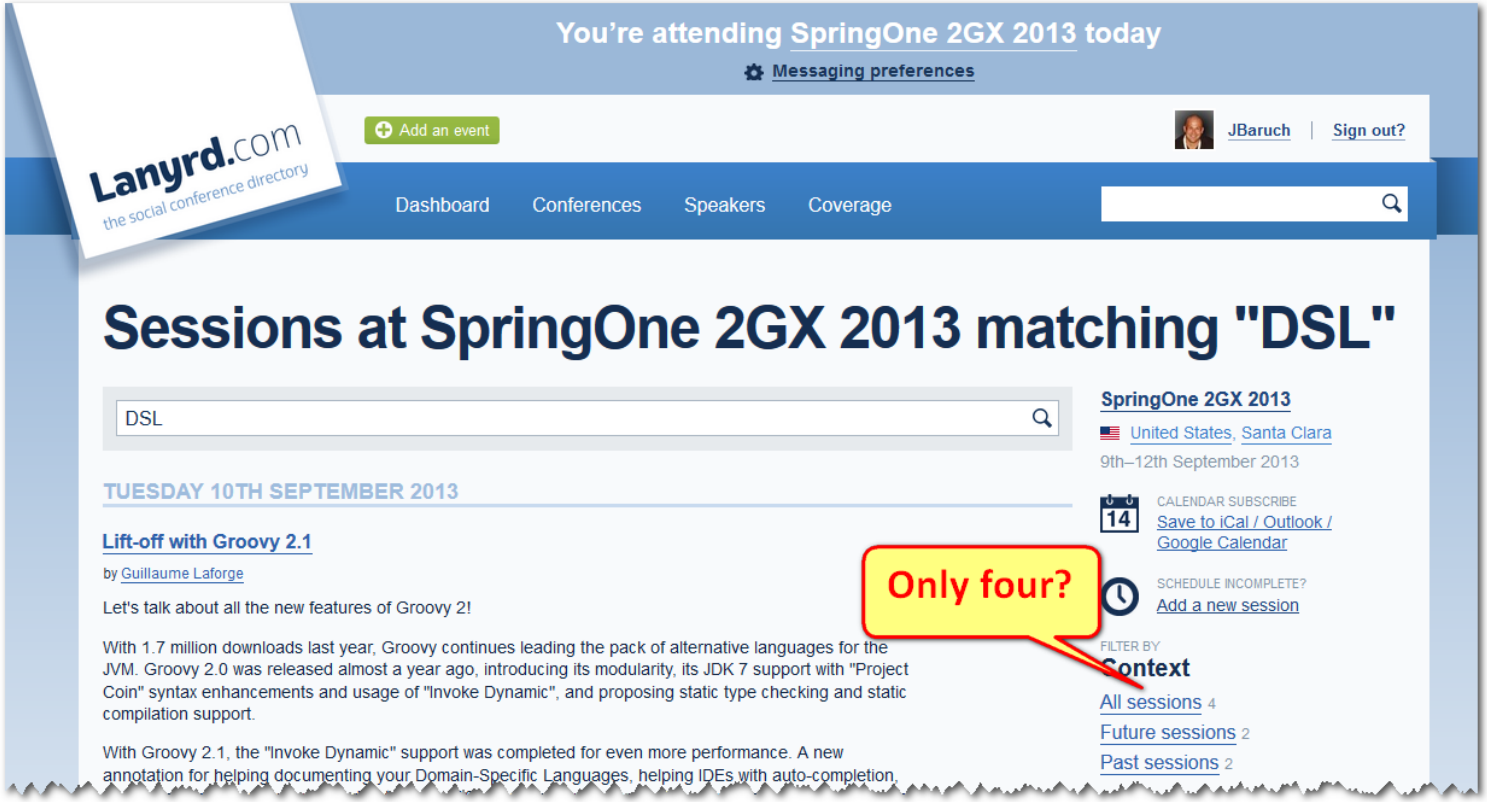
REQUIREMENTS

- FAMILIAR TO JAVA USERS
- SIMPLE DSL
- SIMPLE DEPLOYMENT:
  - NO PACKAGING

REQUIREMENTS

- FAMILIAR TO JAVA USERS
- SIMPLE DSL
- SIMPLE DEPLOYMENT:
  - NO PACKAGING
  - NO RESTART

Of course you know DSL



But anyway...

But anyway...

```
import static mars.Direction.*;
import mars.Robot;
public class Command{
  public static void main(String[] args){
    Robot robot= new Robot();
    robot.move(left);
  }
}
```

But anyway...

```
import static mars.Direction.*;
import mars.Robot;
public class Command{
  public static void main(String[] args){
    Robot robot= new Robot();
    robot.move(left);
  }
}
```

```
move left
```

Groovy Domain-Specific Languages

Paul King
Groovy Core Developer
ASERT
@paulk_asert

Guillaume Laforge
Groovy Project Manager
SpringSource / VMware
@glaforge

19

Domain-Specific Languages

Throughout this chapter, we'll cover:

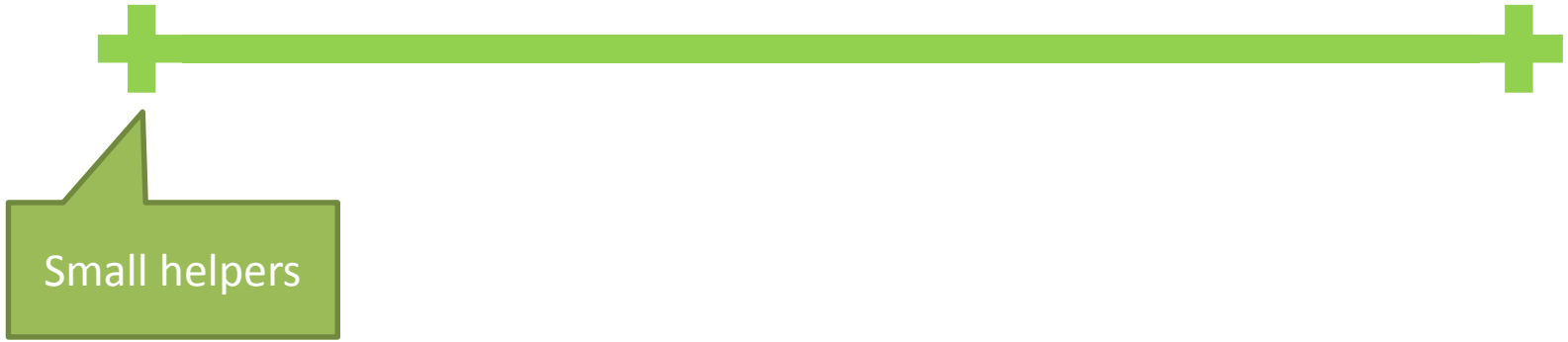
- How Groovy allows you to write Domain-Specific Languages (nicknamed *DSLs*), i.e. languages tailored towards representing a particular domain of knowledge,
- How to concretely integrate DSLs in your applications,
- What the various techniques are to create readable and expressive languages,
- And how to test, secure and provide good error reporting.

Various papers and studies will give you statistics about project successes and failures. On the other end of the spectrum, proponents of development methodologies will tell you about various techniques and approaches you should adopt to ensure that your projects have better chances to succeed. You can read a lot of interesting literature on those topics and I would argue that oftentimes, a common denominator in those sources about what leads a project on its path to success or failure is the quality of communication between the different parties involved, how the various stakeholders exchange information and cooperate together towards a common goal of producing quality software that delivers on their promises at solving a particular domain problem.

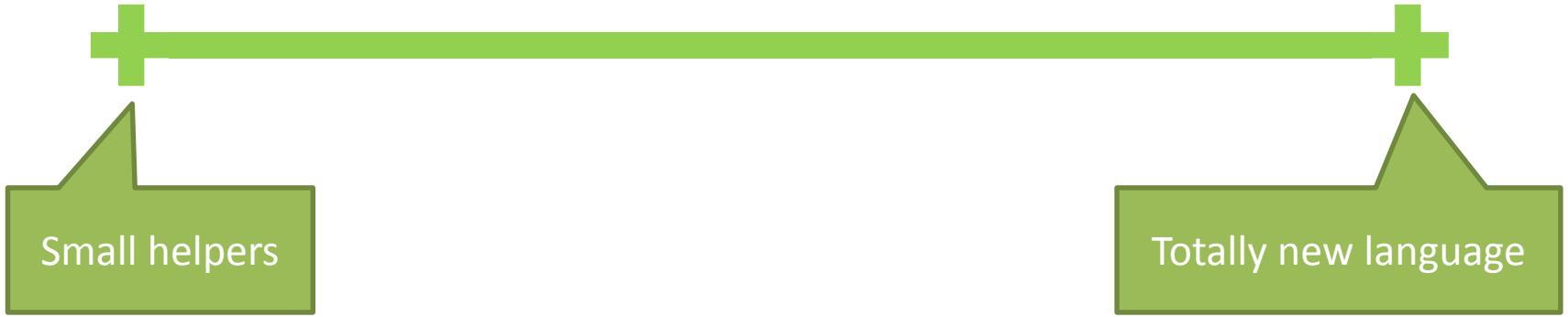
Languages are at the root of any kind of communication and involve two interlocutors. A *Subject Matter Expert* (often reduced to the SME acronym) can write specifications in his mother tongue, say, English, with tons of very domain specific words and concept names that will be read by software developers. A developer can also somehow speak with a computer with different languages to tell it about the business rules of the application the SME is longing to play with. The former will use a natural language while the latter will use one or several

It's a Range

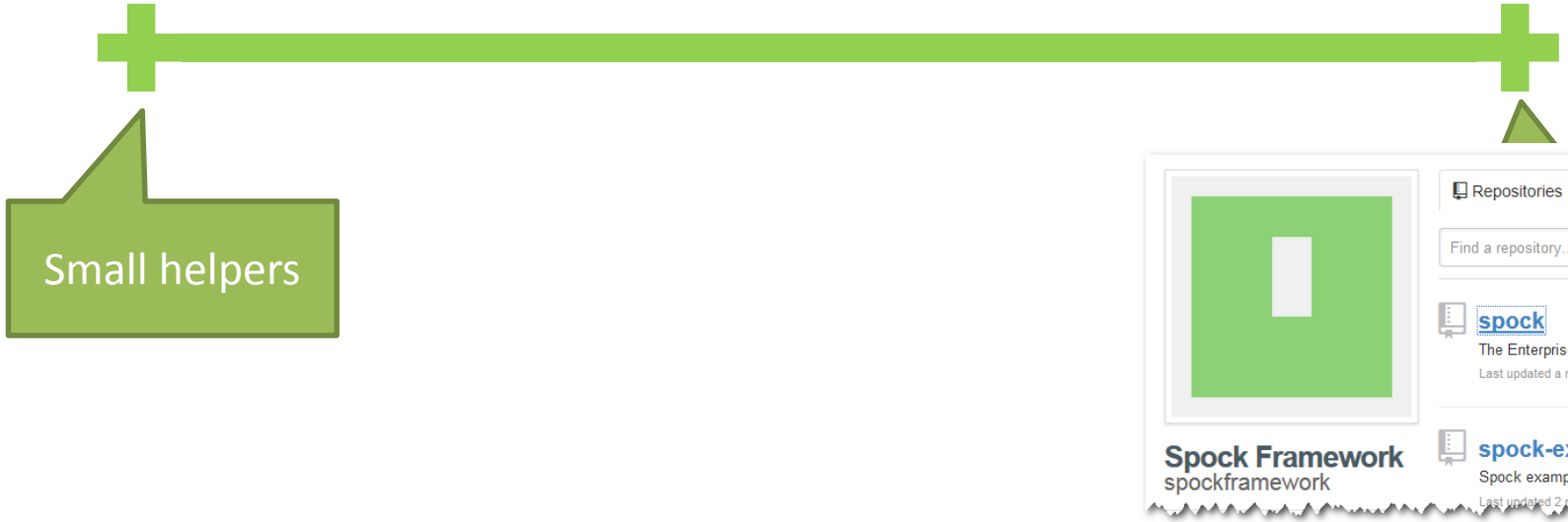
It's a Range

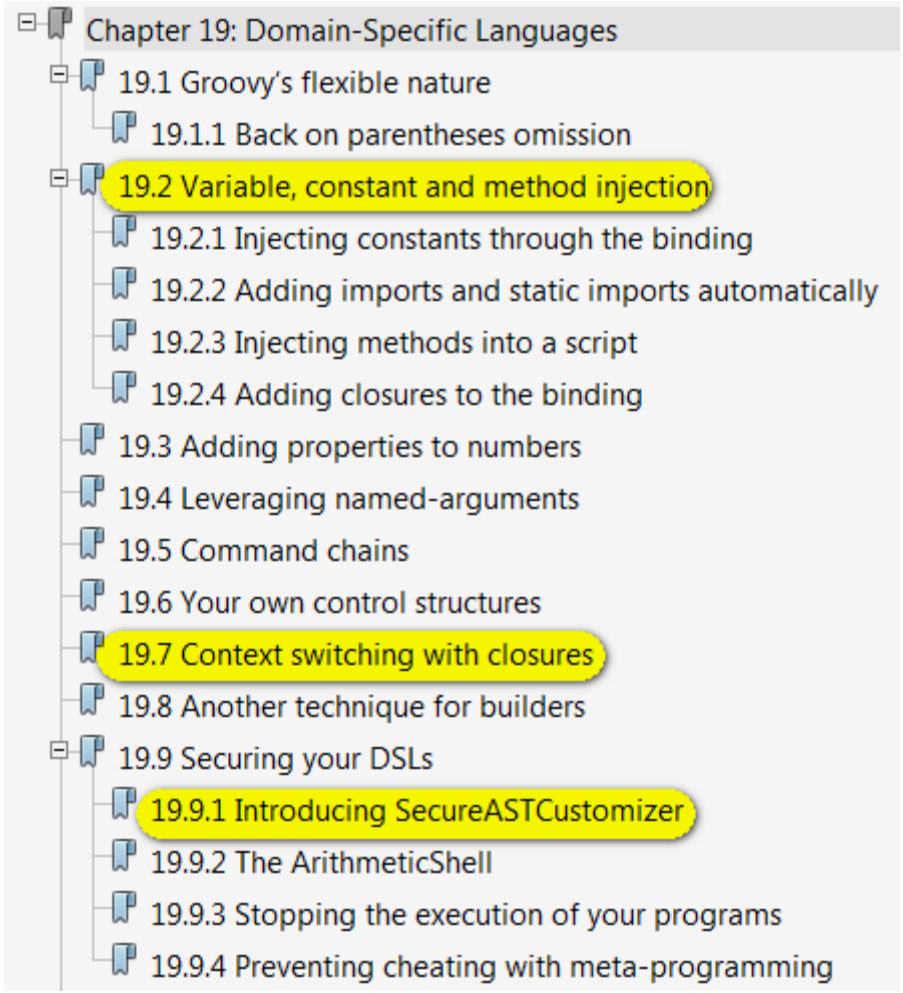


It's a Range

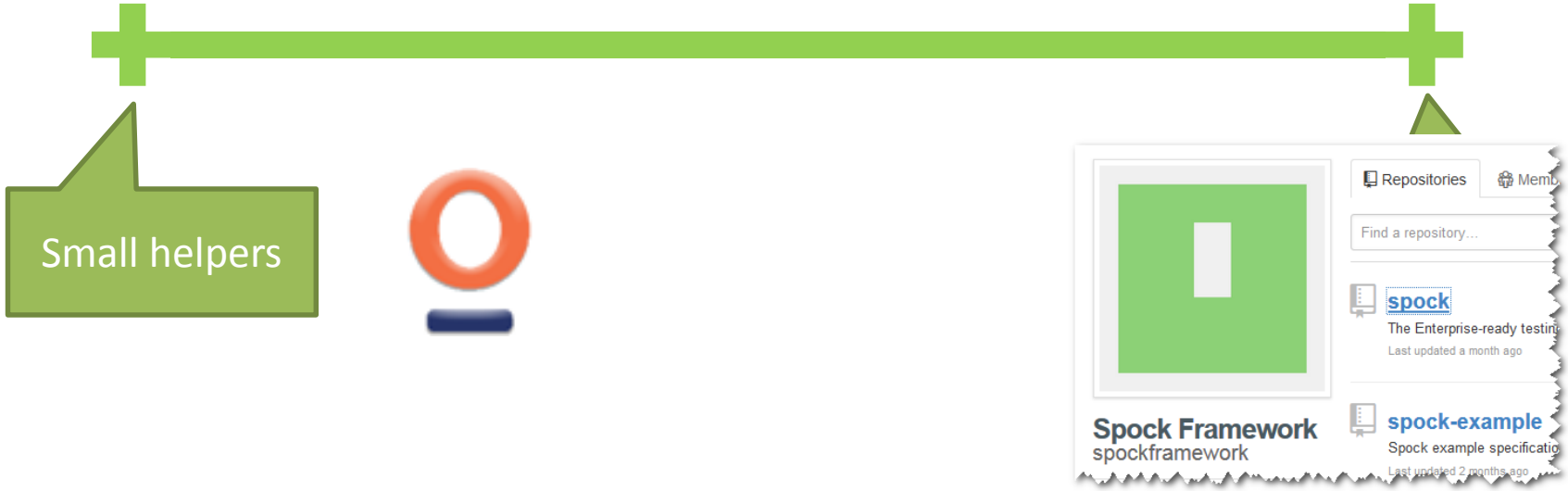


It's a Range


Small helpers
Repositories
Find a repository...
spock
The Enterprise-ready testin
Last updated a month ago
spock-example
Spock example specificatio
Last updated 2 months ago
Spock Framework
spockframework


Chapter 19: Domain-Specific Languages
19.1 Groovy's flexible nature
19.1.1 Back on parentheses omission
19.2 Variable, constant and method injection
19.2.1 Injecting constants through the binding
19.2.2 Adding imports and static imports automatically
19.2.3 Injecting methods into a script
19.2.4 Adding closures to the binding
19.3 Adding properties to numbers
19.4 Leveraging named-arguments
19.5 Command chains
19.6 Your own control structures
19.7 Context switching with closures
19.8 Another technique for builders
19.9 Securing your DSLs
19.9.1 Introducing SecureASTCustomizer
19.9.2 The ArithmeticShell
19.9.3 Stopping the execution of your programs
19.9.4 Preventing cheating with meta-programming

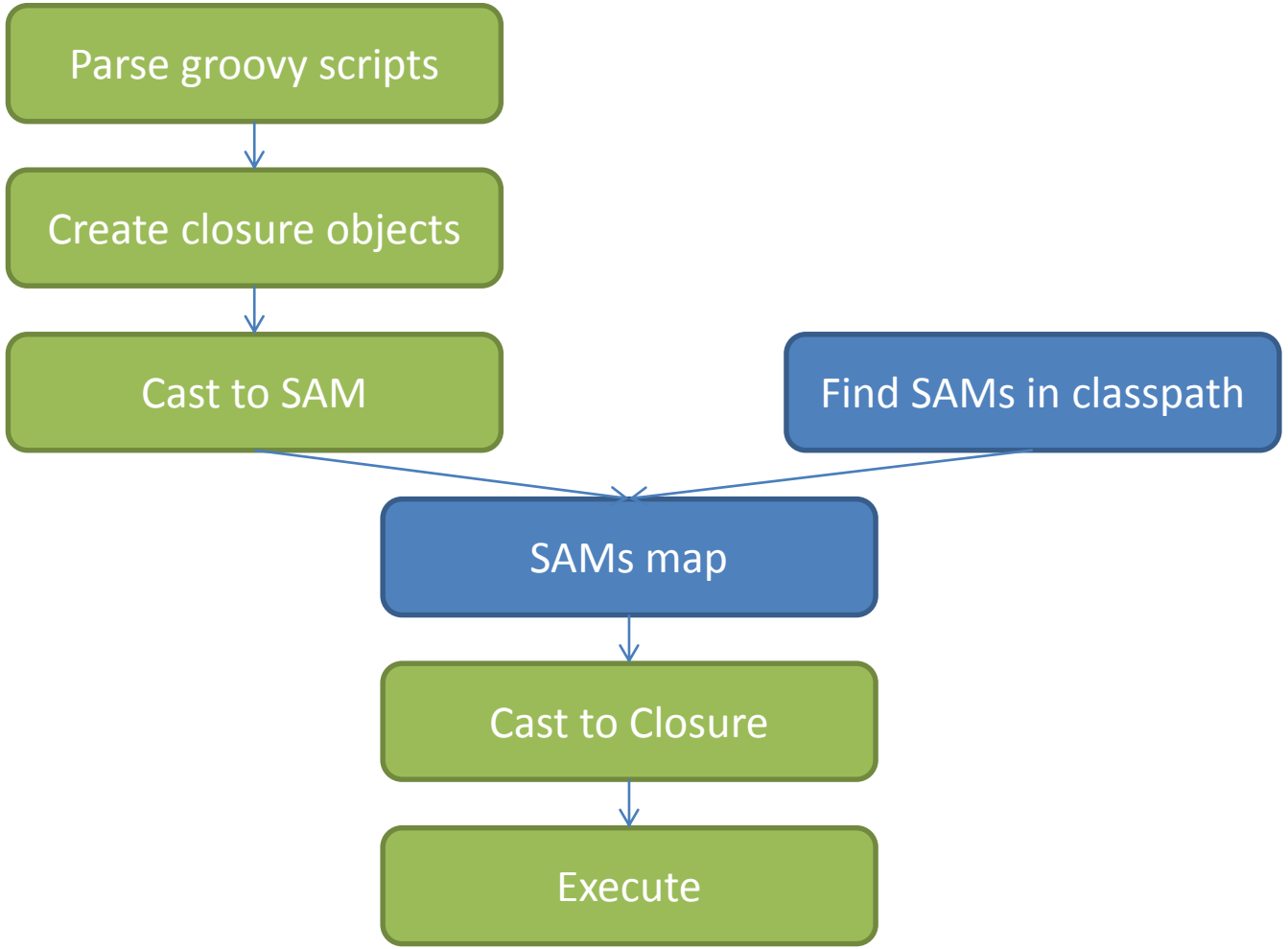
It's a Range


Small helpers
Repositories
Find a repository...
spock
The Enterprise-ready testin
Last updated a month ago
Spock Framework
spockframework
spock-example
Spock example specificatio
Last updated 2 months ago

ImportCustomizer

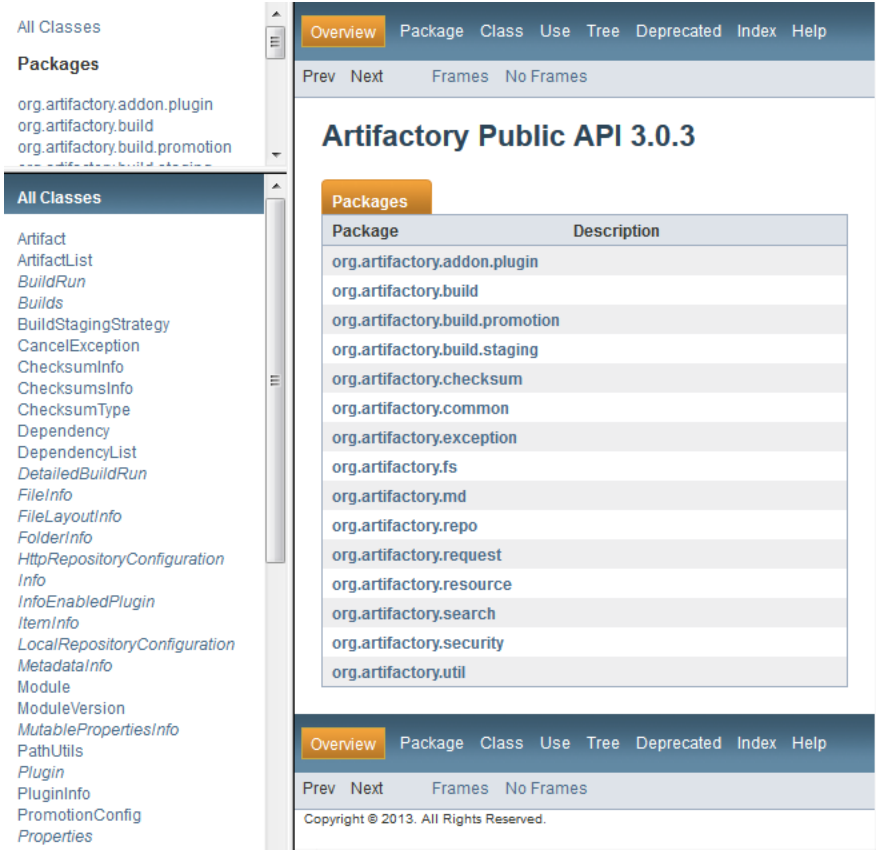
IDEA-113547 **Support DSLs with ImportCustomizer in GDSL**

If ImportCustomizer is added to CompilerConfiguration of the DSL script, all the classes imported by the customerizer should be resolvable without additional imports.


Parse groovy scripts
Create closure objects
Cast to SAM
Find SAMs in classpath
SAMs map
Cast to Closure
Execute

Implementation

Public API


All Classes
Packages
org.artifactory.addon.plugin
org.artifactory.build
org.artifactory.build.promotion
All Classes
Artifact
ArtifactList
BuildRun
Builds
BuildStagingStrategy
CancelException
ChecksumInfo
ChecksumsInfo
ChecksumType
Dependency
DependencyList
DetailedBuildRun
FileInfo
FileLayoutInfo
FolderInfo
HttpRepositoryConfiguration
Info
InfoEnabledPlugin
ItemInfo
LocalRepositoryConfiguration
MetadataInfo
Module
ModuleVersion
MutablePropertiesInfo
PathUtils
Plugin
PluginInfo
PromotionConfig
Properties
Overview Package Class Use Tree Deprecated Index Help
Prev Next Frames No Frames
Artifactory Public API 3.0.3
Packages
Package Description
org.artifactory.addon.plugin
org.artifactory.build
org.artifactory.build.promotion
org.artifactory.build.staging
org.artifactory.checksum
org.artifactory.common
org.artifactory.exception
org.artifactory.fs
org.artifactory.md
org.artifactory.repo
org.artifactory.request
org.artifactory.resource
org.artifactory.search
org.artifactory.security
org.artifactory.util
Overview Package Class Use Tree Deprecated Index Help
Prev Next Frames No Frames
Copyright © 2013. All Rights Reserved.

PUBLIC API DESIGN

PUBLIC API DESIGN

- PAPI IS CONTRACT

PUBLIC API DESIGN

- PAPI IS CONTRACT
- MAINTAIN BACKWARDS COMPATIBILITY AT ALL COSTS

PUBLIC API DESIGN

- PAPI IS CONTRACT
- MAINTAIN BACKWARDS COMPATIBILITY AT ALL COSTS
  - BREAKING API MIGHT CRASH THE SERVER!

PUBLIC API DESIGN

- PAPI IS CONTRACT
- MAINTAIN BACKWARDS COMPATIBILITY AT ALL COSTS
  - BREAKING API MIGHT CRASH THE SERVER!
- START SMALL

PUBLIC API DESIGN

- MAINTAIN BACKWARDS COMPATIBILITY AT ALL COSTS
  - BREAKING API MIGHT CRASH THE SERVER!
- START SMALL
- LISTEN TO USERS

PUBLIC API DESIGN

COMPATIBILITY AT ALL COSTS

- BREAKING API MIGHT CRASH THE SERVER!

- START SMALL
- LISTEN TO USERS
- FLUENT API IS YOUR FRIEND

PUBLIC API DESIGN

- BREAKING API MIGHT CRASH THE SERVER!
- START SMALL
- LISTEN TO USERS
- FLUENT API IS YOUR FRIEND
- STRONG JAVA TYPES, COMPATIBILITY

Implementation

springone 2GX

September 9-12, 2013
Santa Clara, CA, USA

AGENDA | SPEAKERS | TRAVEL | CONTACT | MEMBERS | REGISTER

Creating Groovy DSLs that Developers can Actually Use

In this presentation, Guillaume, Paul, and Andrew will show you how to leverage Groovy to build a Domain-Specific Language (DSL) used to control a rover on Mars! Various metaprogramming techniques and integration mechanisms will be demonstrated. But the language itself is only the first part of the story. Developers cannot be expected to properly use a DSL without first-class IDE support and documentation.

The presentation will start by building the DSL from scratch, using the power of Groovy to create a concise and readable mini-language, and showing how to secure its integration. The second part of the presentation will demonstrate how to integrate the DSL into Groovy-Eclipse with custom content assist, navigation, searching, and inline documentation.

About the speaker

Guillaume LaForge

As Head of Groovy Development for SpringSource, Guillaume Laforge is the official Groovy Project Manager. He initiated the creation of the Grails web framework, and created the Gaelyk lightweight toolkit for Google App Engine. He is also a frequent conference speaker presenting Groovy and Grails at JavaOne, SpringOne, QCon, the Sun TechDays, and JavaPolis. Guillaume also co-authored Groovy in Action. Before founding G2One, which was acquired by SpringSource in late 2008, and taking the role of VP Technology, Guillaume worked for OCTO Technology, a consultancy focusing on architecture and agile methodologies. While at OCTO, Guillaume developed new offerings around Groovy and Grails for its customers.

More About Guillaume »

Paul King

Paul King leads ASERT, an organization based in Brisbane, Australia which provides software development, training and mentoring services to customers wanting to embrace new technologies, harness best practices and innovate. He has been contributing to open source projects for nearly 20 years and is an active committer on numerous projects including Groovy. Paul speaks at international conferences, publishes in software magazines and journals, and is a co-author of Manning's best-seller: Groovy in Action.

More About Paul »


What about security and safety?
RANGER
HAM

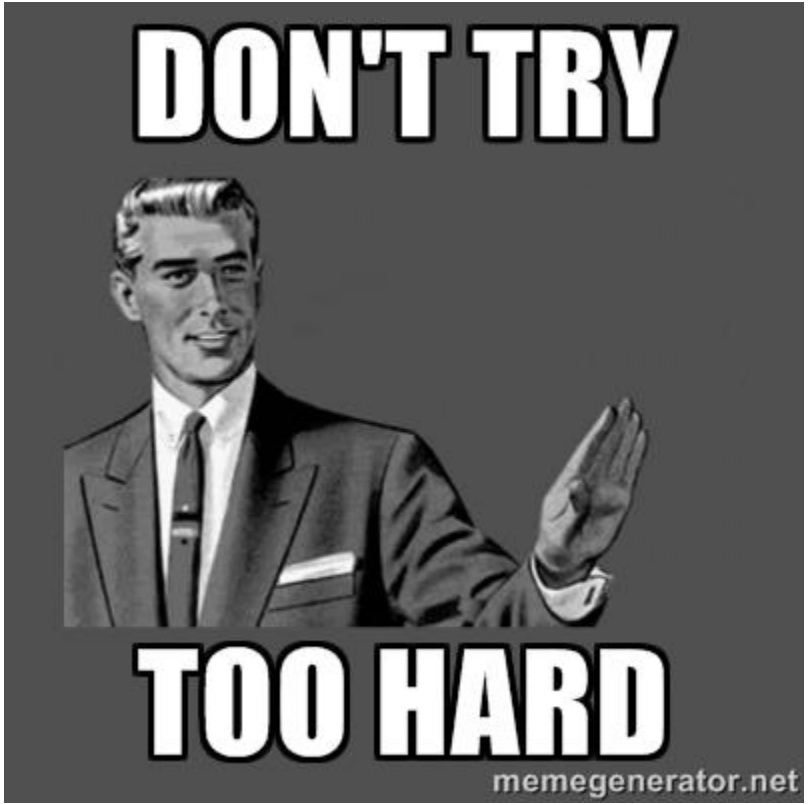
Security and Safety

JVM Security Managers

SecureASTCustomizer

Sandboxing

Controlling script execution



It's their server!

Plugin Dependencies

Plugin Dependencies

THE CONFLICT

THE CONFLICT

- NEED TO ACCESS ARTIFACTORY JARS (PAPI)

THE CONFLICT

- NEED TO ACCESS ARTIFACTORY JARS (PAPI)
- NEED TO BE ISOLATED FROM ARTIFACTORY JARS (3RD PARTY)


Gerd Wütherich · Nils Hartmann
Bernd Kolb · Matthias Lübken
Die OSGi
Service Platform
Eine Einführung mit Eclipse Equinox
dpunkt.verlag

JBoss modules

David M. Lloyd
@dmlloyd0

@jbaruch the manual at docs.jboss.org /author/display... is fairly complete to get you started, also check out the #wildfly distro.

JBoss modules

David M. Lloyd
@dmlloyd0

@jbaruch the manual at docs.jboss.org /author/display... is fairly complete to get you started, also check out the #wildfly distro.

Introduction

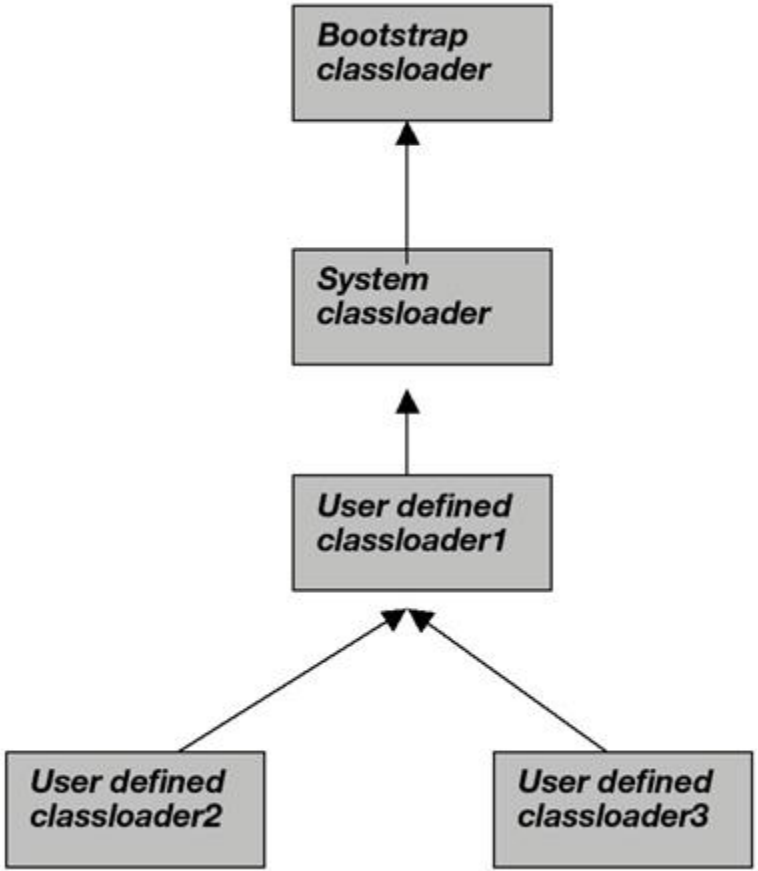
Tools

Added by David Lloyd, last edited by David Lloyd on Jul 13, 2011 (view change)

JBoss Modules is a standalone implementation of a *modular* (non-hierarchical) class loading and execution environment for Java. In other words, rather than a single class loader which loads all JARs into a flat class path, each library becomes a *module* which only links against the exact modules it depends on, and nothing more. It implements a thread-safe, fast, and highly concurrent delegating class loader model, coupled to an extensible module resolution system, which combine to form a unique, simple and powerful system for application execution and distribution.

JBoss Modules is designed to work with any existing library or application without changes, and its simple naming and resolution strategy is what makes that possible. Unlike OSGi, JBoss Modules does not implement a container; rather, it is a thin bootstrap wrapper for executing an application in a modular environment. The moment your application takes control, the modular environment is ready to load and link modules as needed. Furthermore, modules are never loaded (not even for resolution purposes) until required by a dependency, meaning that the performance of a modular application depends only on the number of modules actually used (and when they are used), rather than the total number of modules in the system. And, they may be unloaded by the user at any time.

WE'LL DO IT OURSELVES!


Bootstrap classloader
System classloader
User defined classloader1
User defined classloader2
User defined classloader3

WE'LL DO IT OURSELVES!

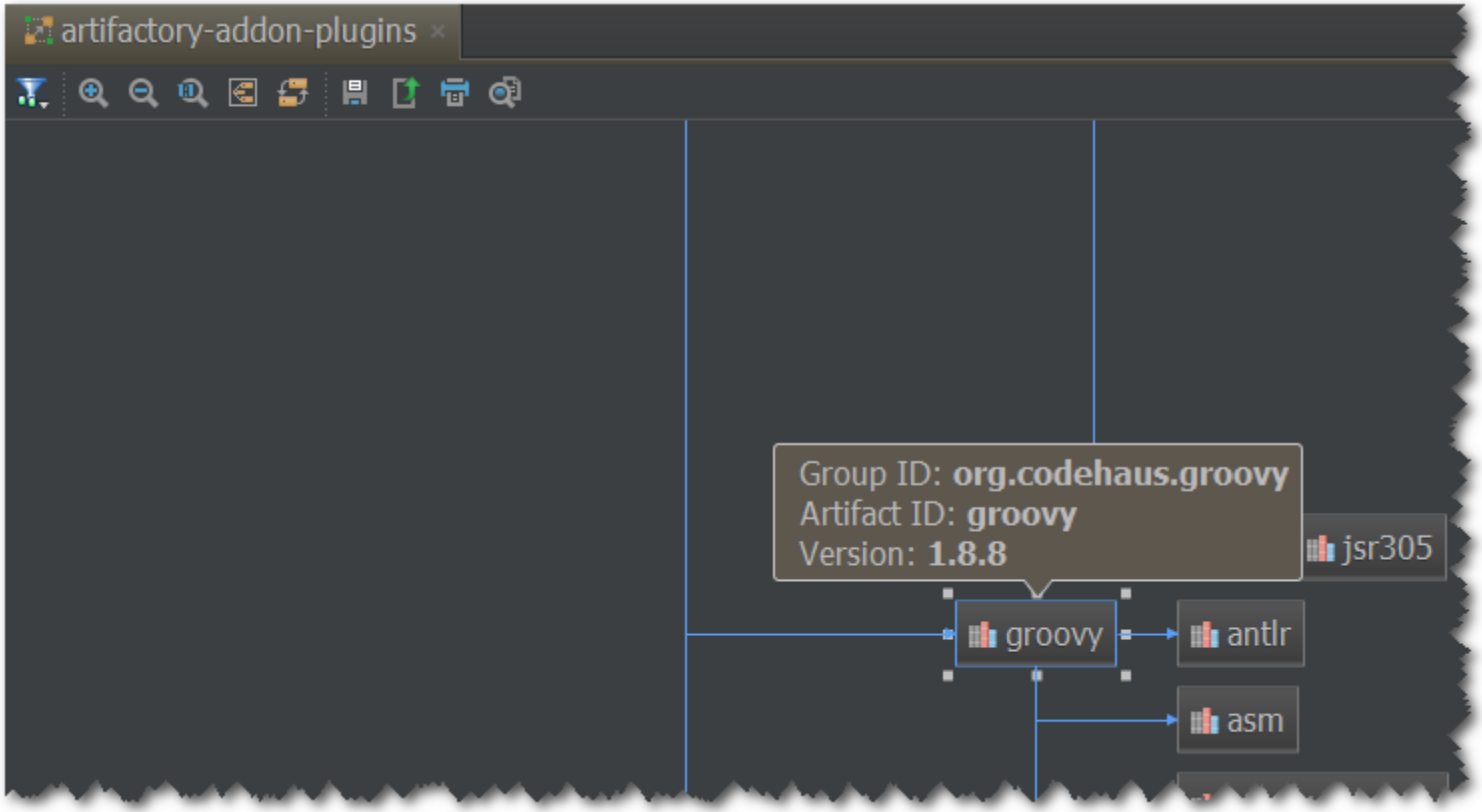
1. PULL UP WHAT'S COMMON

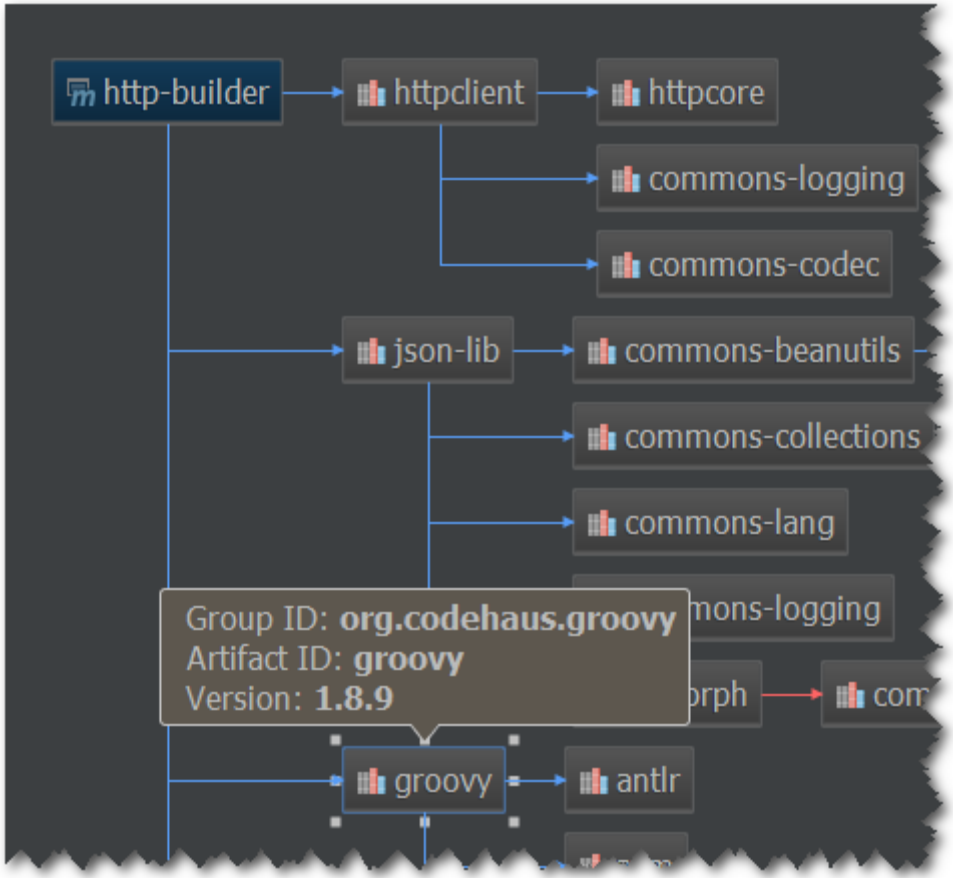
WE'LL DO IT OURSELVES!

1. PULL UP WHAT'S COMMON
2. ISOLATE THE REST

WE'LL DO IT OURSELVES!

1. F[illegible]MMON
2.


artifactory-addon-plugins
Group ID: org.codehaus.groovy
Artifact ID: groovy
Version: 1.8.8
jsr305
groovy
antlr
asm


http-builder
httpclient
httpcore
commons-logging
commons-codec
json-lib
commons-beanutils
commons-collections
commons-lang
Group ID: org.codehaus.groovy
Artifact ID: groovy
Version: 1.8.9
groovy
antlr

No, thank you!