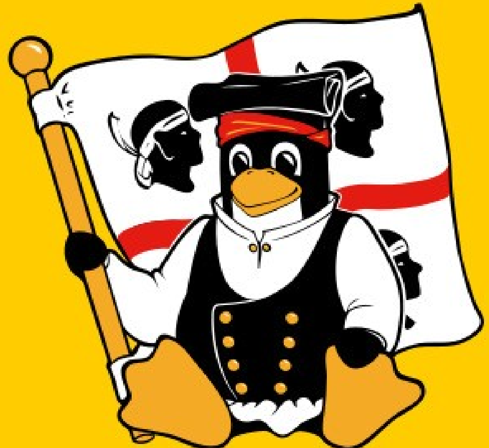
Hacking Maven
how to add steroids on Maven

di Massimiliano Dessì

2017
LINUX DAY
ITALIA



GULCh
Gruppo Utenti Linux Cagliari h...?

Abstract

30 minutes to illustrate from 30000 ft
the ideas applied to turn Maven from a “static” producer
to a “rich” compiler enabled to reads objects
usually unacessible

Speaker @desmax74

Massimiliano Dessì has more than 17 years of experience in programming.

Manager of GDG Sardegna,

co-founder of Jug Sardegna.

Author of Spring 2.5 AOP.

He works as a Senior Software Engineer for

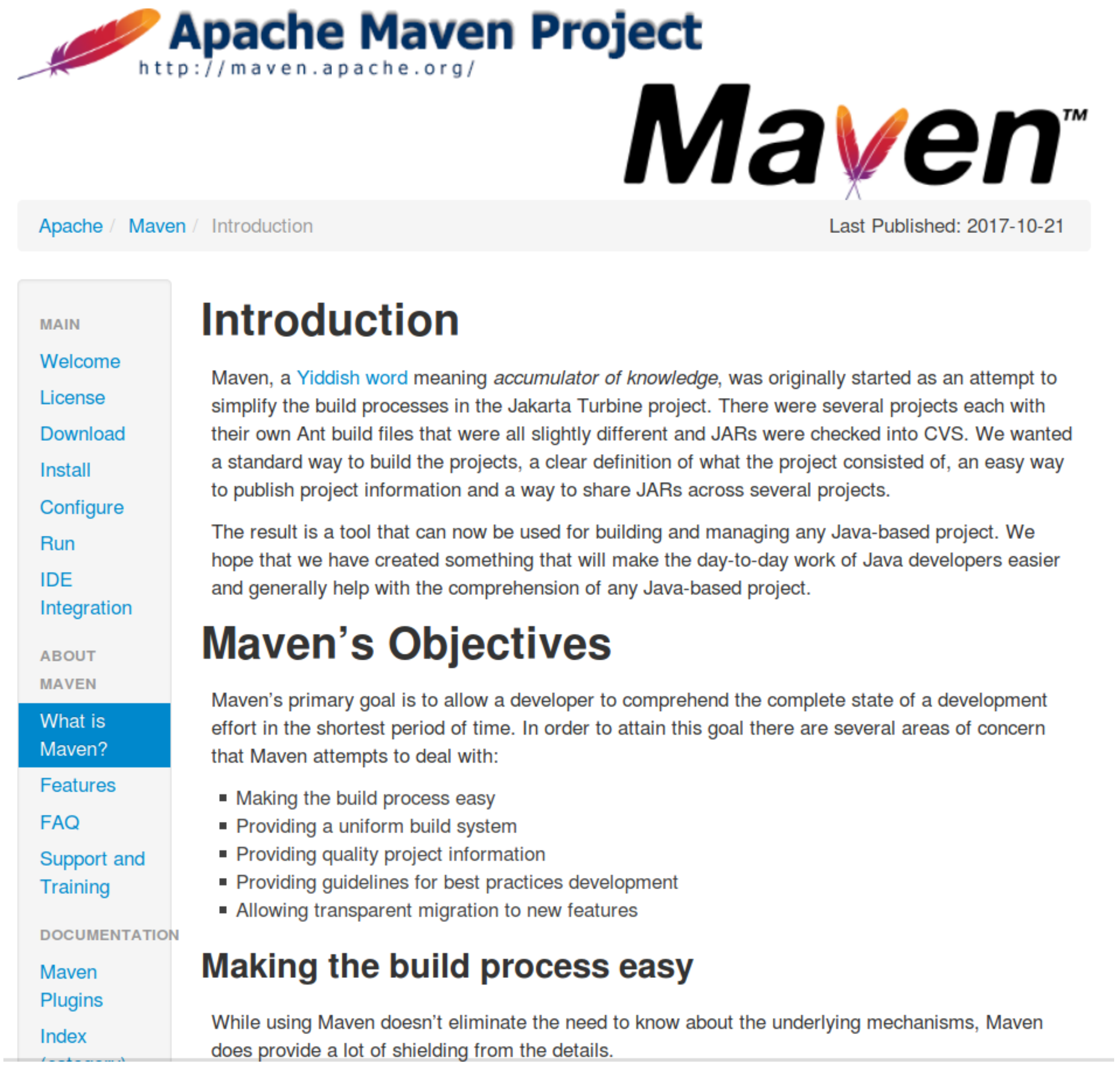
Red Hat in the

Business Systems and Intelligence Group

(aka KIE - Knowledge Is Everything),

he lives in Cagliari, Sardinia, Italy.

Starting point: Maven's Objectives


Apache Maven Project
http://maven.apache.org/

Maven™

Apache / Maven / Introduction   Last Published: 2017-10-21

MAIN
Welcome
License
Download
Install
Configure
Run
IDE Integration

ABOUT MAVEN
What is Maven?
Features
FAQ
Support and Training

DOCUMENTATION
Maven Plugins
Index

Introduction

Maven, a Yiddish word meaning *accumulator of knowledge*, was originally started as an attempt to simplify the build processes in the Jakarta Turbine project. There were several projects each with their own Ant build files that were all slightly different and JARs were checked into CVS. We wanted a standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information and a way to share JARs across several projects.

The result is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

Maven's Objectives

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

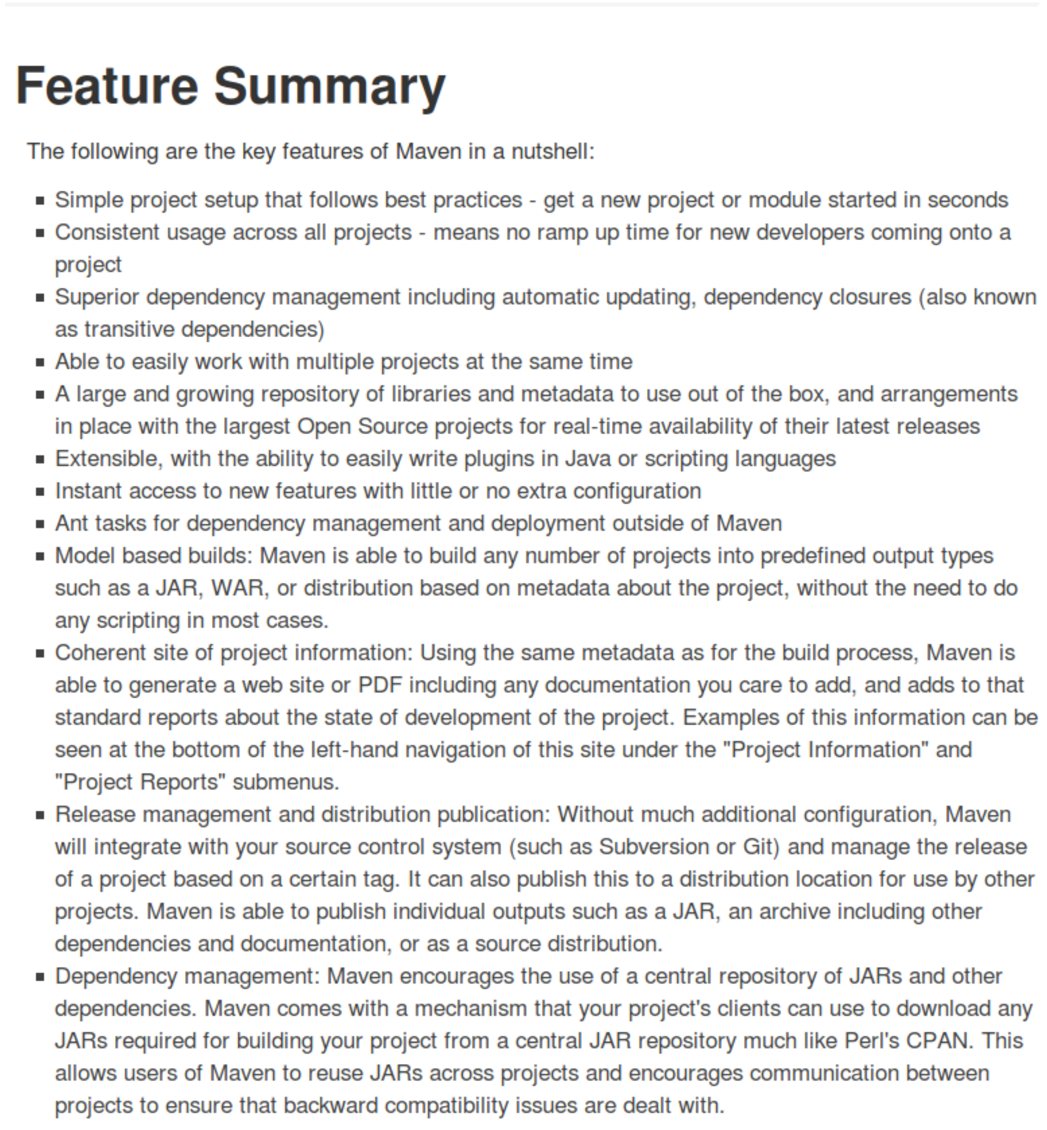
- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best practices development
- Allowing transparent migration to new features

Making the build process easy

While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does provide a lot of shielding from the details.

https://maven.apache.org/what-is-maven.html

Maven's features



Feature Summary

The following are the key features of Maven in a nutshell:

- Simple project setup that follows best practices - get a new project or module started in seconds
- Consistent usage across all projects - means no ramp up time for new developers coming onto a project
- Superior dependency management including automatic updating, dependency closures (also known as transitive dependencies)
- Able to easily work with multiple projects at the same time
- A large and growing repository of libraries and metadata to use out of the box, and arrangements in place with the largest Open Source projects for real-time availability of their latest releases
- Extensible, with the ability to easily write plugins in Java or scripting languages
- Instant access to new features with little or no extra configuration
- Ant tasks for dependency management and deployment outside of Maven
- Model based builds: Maven is able to build any number of projects into predefined output types such as a JAR, WAR, or distribution based on metadata about the project, without the need to do any scripting in most cases.
- Coherent site of project information: Using the same metadata as for the build process, Maven is able to generate a web site or PDF including any documentation you care to add, and adds to that standard reports about the state of development of the project. Examples of this information can be seen at the bottom of the left-hand navigation of this site under the "Project Information" and "Project Reports" submenus.
- Release management and distribution publication: Without much additional configuration, Maven will integrate with your source control system (such as Subversion or Git) and manage the release of a project based on a certain tag. It can also publish this to a distribution location for use by other projects. Maven is able to publish individual outputs such as a JAR, an archive including other dependencies and documentation, or as a source distribution.
- Dependency management: Maven encourages the use of a central repository of JARs and other dependencies. Maven comes with a mechanism that your project's clients can use to download any JARs required for building your project from a central JAR repository much like Perl's CPAN. This allows users of Maven to reuse JARs across projects and encourages communication between projects to ensure that backward compatibility issues are dealt with.

Pretty boring isn't it ?

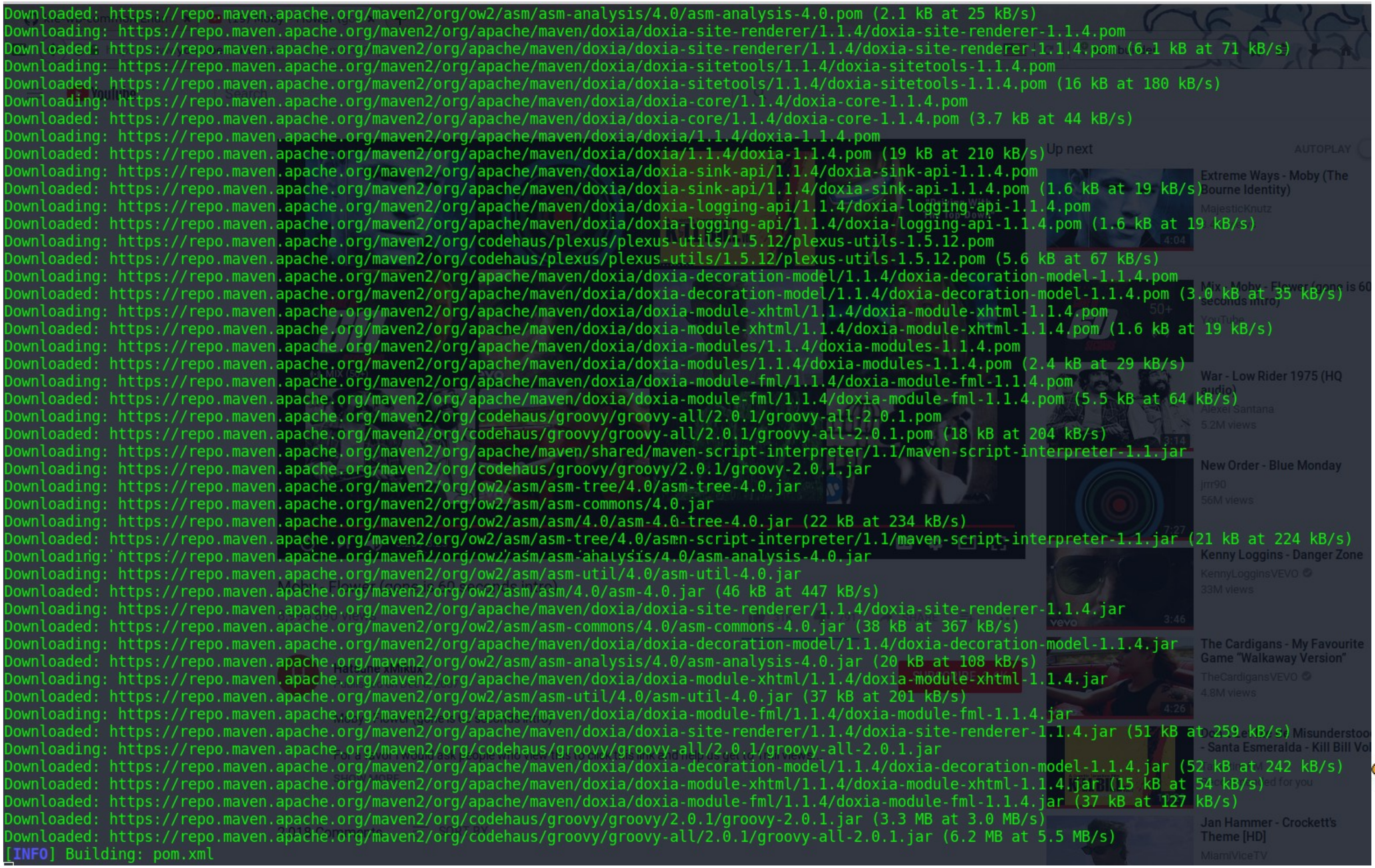
https://maven.apache.org/maven-features.html

Context

Basically Maven produces artifacts on filesystem using plugins,
Jar,War, Ear, documentation, but basically is a “box” to product other files,
from Java multimodule projects.

In our group we using a plugin and a pipeline to process rules and other
projects managed by Drools/Optaplanner/JBPM to produce files ,
but are “dead” files, not a complete representation of a runtime objects.

Maven, the visible (and hated) part



```
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-analysis/4.0/asm-analysis-4.0.pom (2.1 kB at 25 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-site-renderer/1.1.4/doxia-site-renderer-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-site-renderer/1.1.4/doxia-site-renderer-1.1.4.pom (6.1 kB at 71 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sitetools/1.1.4/doxia-sitetools-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sitetools/1.1.4/doxia-sitetools-1.1.4.pom (16 kB at 180 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-core/1.1.4/doxia-core-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-core/1.1.4/doxia-core-1.1.4.pom (3.7 kB at 44 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia/1.1.4/doxia-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia/1.1.4/doxia-1.1.4.pom (19 kB at 210 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.1.4/doxia-sink-api-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.1.4/doxia-sink-api-1.1.4.pom (1.6 kB at 19 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1.4/doxia-logging-api-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1.4/doxia-logging-api-1.1.4.pom (1.6 kB at 19 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/1.5.12/plexus-utils-1.5.12.pom
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/1.5.12/plexus-utils-1.5.12.pom (5.6 kB at 67 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.4/doxia-decoration-model-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.4/doxia-decoration-model-1.1.4.pom (3.0 kB at 35 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.4/doxia-module-xhtml-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.4/doxia-module-xhtml-1.1.4.pom (1.6 kB at 19 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-modules/1.1.4/doxia-modules-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-modules/1.1.4/doxia-modules-1.1.4.pom (2.4 kB at 29 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-fml/1.1.4/doxia-module-fml-1.1.4.pom
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-fml/1.1.4/doxia-module-fml-1.1.4.pom (5.5 kB at 64 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-all/2.0.1/groovy-all-2.0.1.pom
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-all/2.0.1/groovy-all-2.0.1.pom (18 kB at 204 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-script-interpreter/1.1/maven-script-interpreter-1.1.jar
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy/2.0.1/groovy-2.0.1.jar
Downloading: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-tree/4.0/asm-tree-4.0.jar
Downloading: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-commons/4.0.jar
Downloading: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/4.0/asm-4.0-tree-4.0.jar (22 kB at 234 kB/s)
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-tree/4.0/asmn-script-interpreter/1.1/maven-script-interpreter-1.1.jar (21 kB at 224 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-analysis/4.0/asm-analysis-4.0.jar
Downloading: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-util/4.0/asm-util-4.0.jar
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/4.0/asm-4.0.jar (46 kB at 447 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-site-renderer/1.1.4/doxia-site-renderer-1.1.4.jar
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-commons/4.0/asm-commons-4.0.jar (38 kB at 367 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.4/doxia-decoration-model-1.1.4.jar
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-analysis/4.0/asm-analysis-4.0.jar (20 kB at 108 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.4/doxia-module-xhtml-1.1.4.jar
Downloaded: https://repo.maven.apache.org/maven2/org/ow2/asm/asm-util/4.0/asm-util-4.0.jar (37 kB at 201 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-fml/1.1.4/doxia-module-fml-1.1.4.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-site-renderer/1.1.4/doxia-site-renderer-1.1.4.jar (51 kB at 259 kB/s)
Downloading: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-all/2.0.1/groovy-all-2.0.1.jar
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.4/doxia-decoration-model-1.1.4.jar (52 kB at 242 kB/s)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.4/doxia-module-xhtml-1.1.4.jar (15 kB at 54 kB/s)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-fml/1.1.4/doxia-module-fml-1.1.4.jar (37 kB at 127 kB/s)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy/2.0.1/groovy-2.0.1.jar (3.3 MB at 3.0 MB/s)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-all/2.0.1/groovy-all-2.0.1.jar (6.2 MB at 5.5 MB/s)
[INFO] Building: pom.xml
```

Embedding

Our first goal is to use Maven like a normal API,
Maven could be embedded in two ways using two libraries:

Maven Invoker, open a new process separated from the caller
https://maven.apache.org/plugins/maven-invoker-plugin/index.html

Maven Embedder, Works in the same process of the caller
https://maven.apache.org/ref/3.5.2/maven-embedder/index.html

this was our starting point

Our New Requirements

Cloud

Openshift/Kubernetes

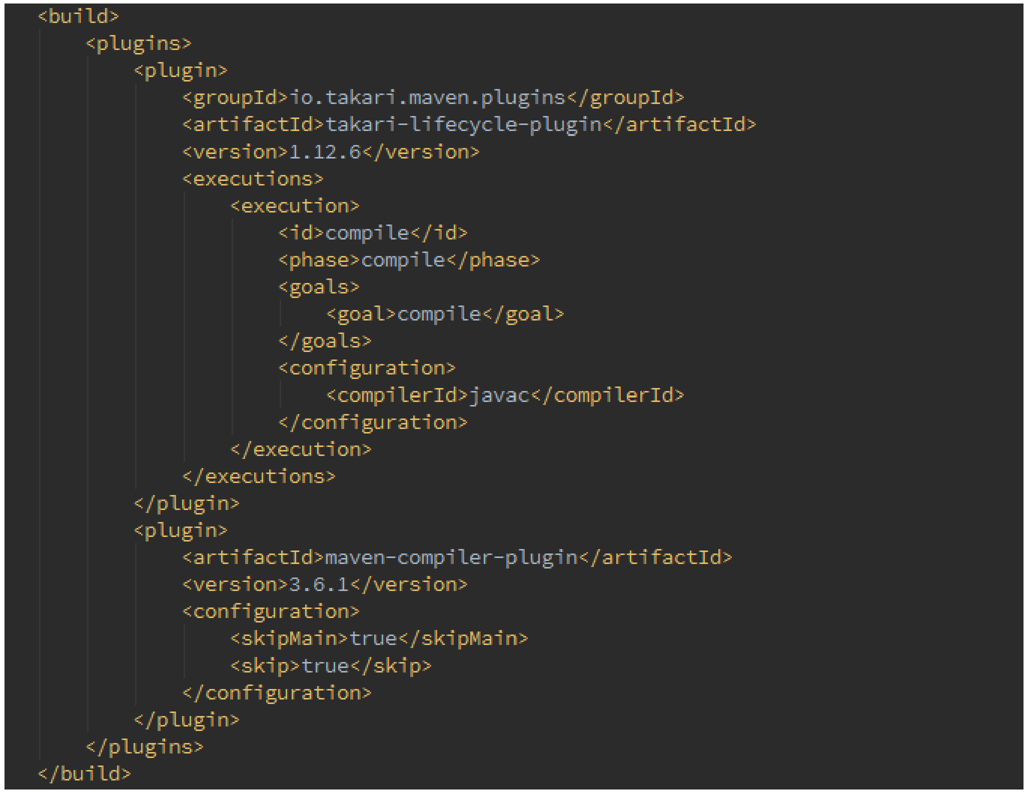
Containers

Fast as possible between builds

Live objects :)

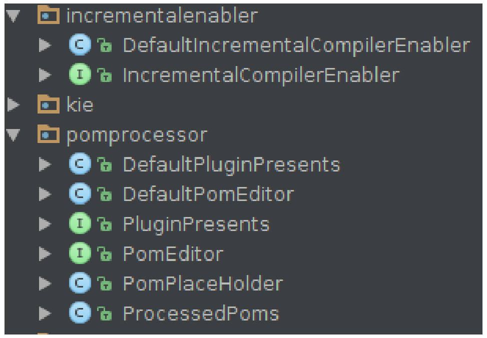
InseHacking Maven

Let’s go to do something challenging
but first we want to optimize the time in front of Maven
enabling the incremental compiler called Takari



```
<build>
    <plugins>
        <plugin>
            <groupId>io.takari.maven.plugins</groupId>
            <artifactId>takari-lifecycle-plugin</artifactId>
            <version>1.12.6</version>
            <executions>
                <execution>
                    <id>compile</id>
                    <phase>compile</phase>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                    <configuration>
                        <compilerId>javac</compilerId>
                    </configuration>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.6.1</version>
            <configuration>
                <skipMain>true</skipMain>
                <skip>true</skip>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Every time a new project is discovered



```
▼ incrementalenabler
    ▶ DefaultIncrementalCompilerEnabler
    ▶ IncrementalCompilerEnabler
▶ kie
▼ pomprocessor
    ▶ DefaultPluginPresents
    ▶ DefaultPomEditor
    ▶ PluginPresents
    ▶ PomEditor
    ▶ PomPlaceHolder
    ▶ ProcessedPoms
```

http://takari.io/ https://github.com/takari

Request-Response behaviour

In our initial design we want to ask to Maven:

Result of the build

Output Log

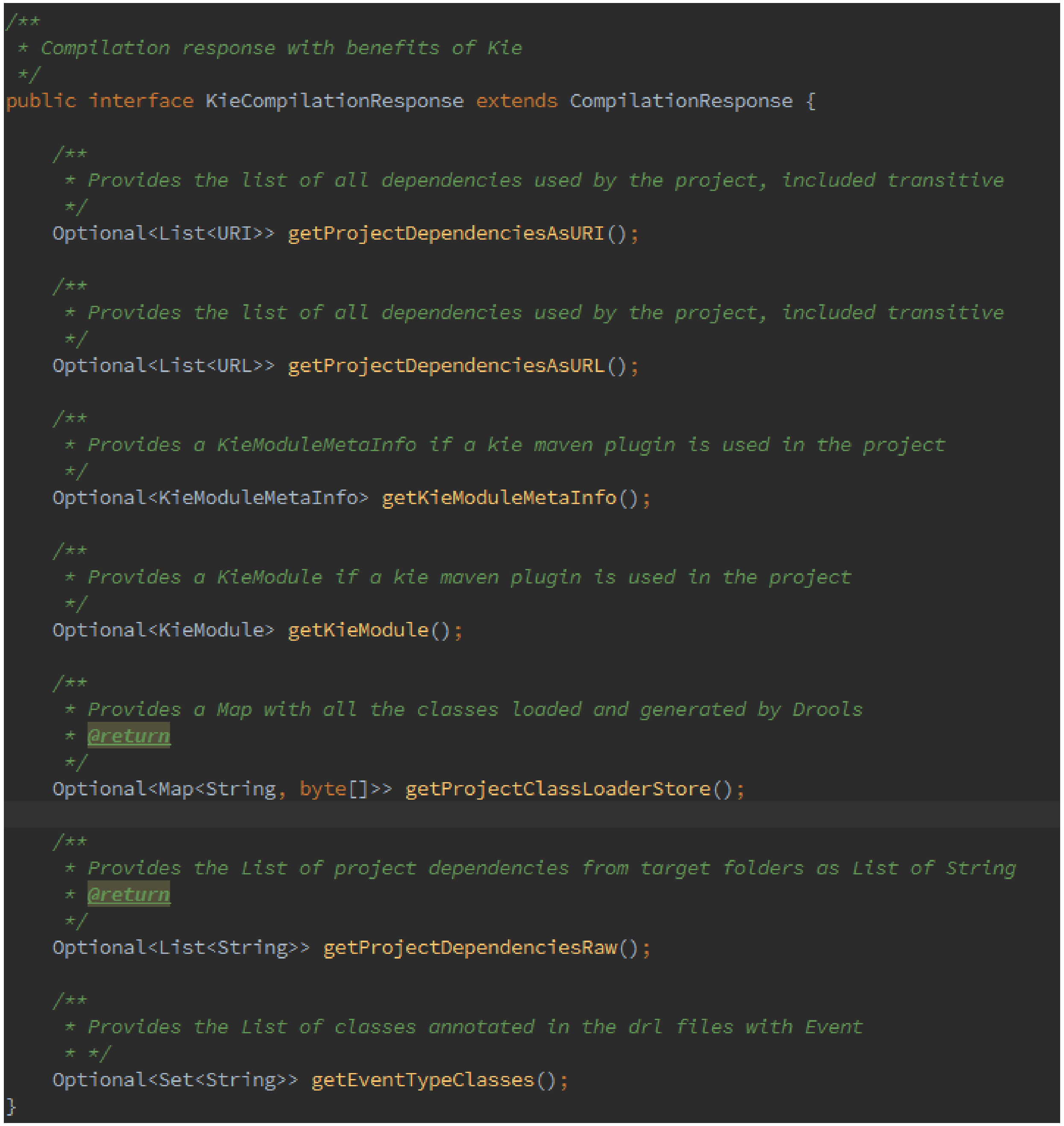
Classloaders

Live objects

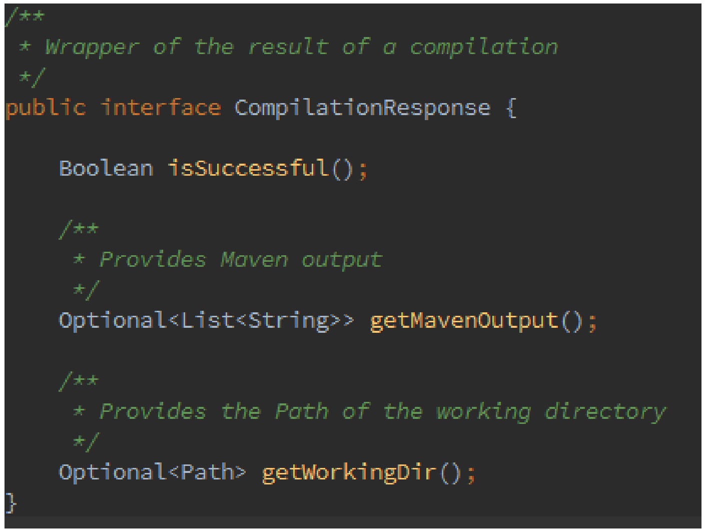
Live objects generated on the fly (no .class file)

then we will add other useful features

Response contracts



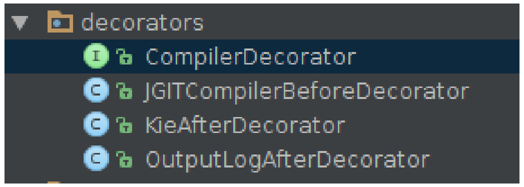
```
/**
 * Compilation response with benefits of Kie
 */
public interface KieCompilationResponse extends CompilationResponse {

    /**
     * Provides the list of all dependencies used by the project, included transitive
     */
    Optional<List<URI>> getProjectDependenciesAsURI();

    /**
     * Provides the list of all dependencies used by the project, included transitive
     */
    Optional<List<URL>> getProjectDependenciesAsURL();

    /**
     * Provides a KieModuleMetaInfo if a kie maven plugin is used in the project
     */
    Optional<KieModuleMetaInfo> getKieModuleMetaInfo();

    /**
     * Provides a KieModule if a kie maven plugin is used in the project
     */
    Optional<KieModule> getKieModule();

    /**
     * Provides a Map with all the classes loaded and generated by Drools
     * @return
     */
    Optional<Map<String, byte[]>> getProjectClassLoaderStore();

    /**
     * Provides the List of project dependencies from target folders as List of String
     * @return
     */
    Optional<List<String>> getProjectDependenciesRaw();

    /**
     * Provides the List of classes annotated in the drl files with Event
     * */
    Optional<Set<String>> getEventTypeClasses();
}
```



```
/**
 * Wrapper of the result of a compilation
 */
public interface CompilationResponse {

    Boolean isSuccessful();

    /**
     * Provides Maven output
     */
    Optional<List<String>> getMavenOutput();

    /**
     * Provides the Path of the working directory
     */
    Optional<Path> getWorkingDir();
}
```

How to reach some of our goals ?

Since we want to add objects to the Maven “embedded” result
(a int to signal success or failure)
we use a pipeline of decorators to add behaviour before and after compilation
(we could add additional decorators to add behaviours)



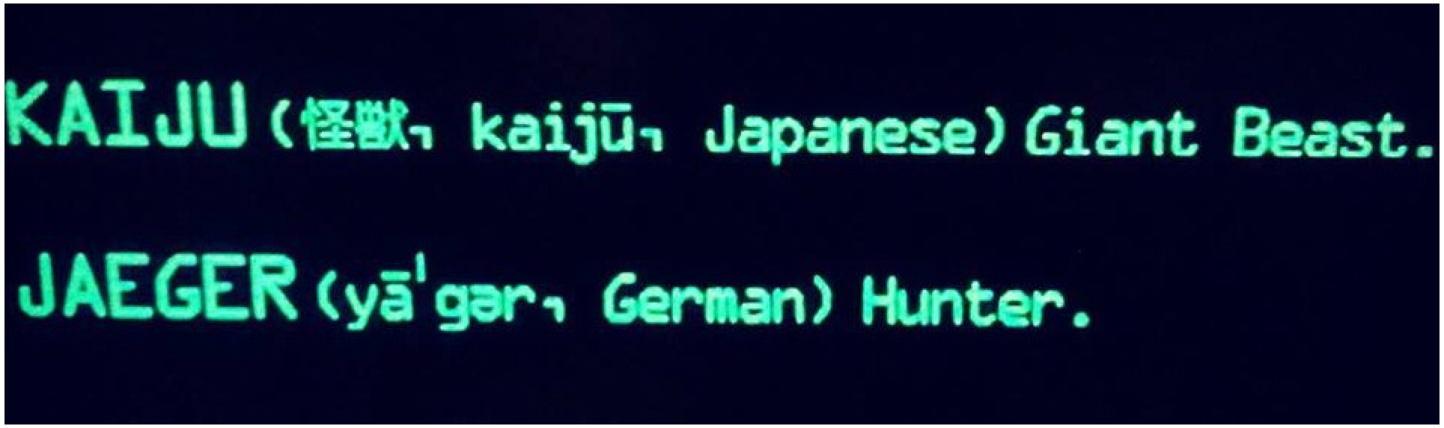
```
decorators
    CompilerDecorator
    JGITCompilerBeforeDecorator
    KieAfterDecorator
    OutputLogAfterDecorator
```

Before to sync the project with the git origin repo because our project coming from git and the changes are commited to be visible to other users

After to add the KieObjects readed from inside Maven, or to add the Maven output if requested in the CompileRequest

The hardest part

Maven is designed
to be extensible with plugins
for the processing
not for changes in its internals


KAIJU (怪獣, kaijū, Japanese) Giant Beast.

JAEGER (yā'gər, German) Hunter.

But we are highly motivated :)

Maven Internal component

Maven for its job use

Plexus/Eclipse Sisu and ClassWorlds

Plexus/Eclipse Sisu is a IoC container

Classworlds is used to manage the different

classloaders required

Classloading

Classworlds is used to create this hierarchy of classloader

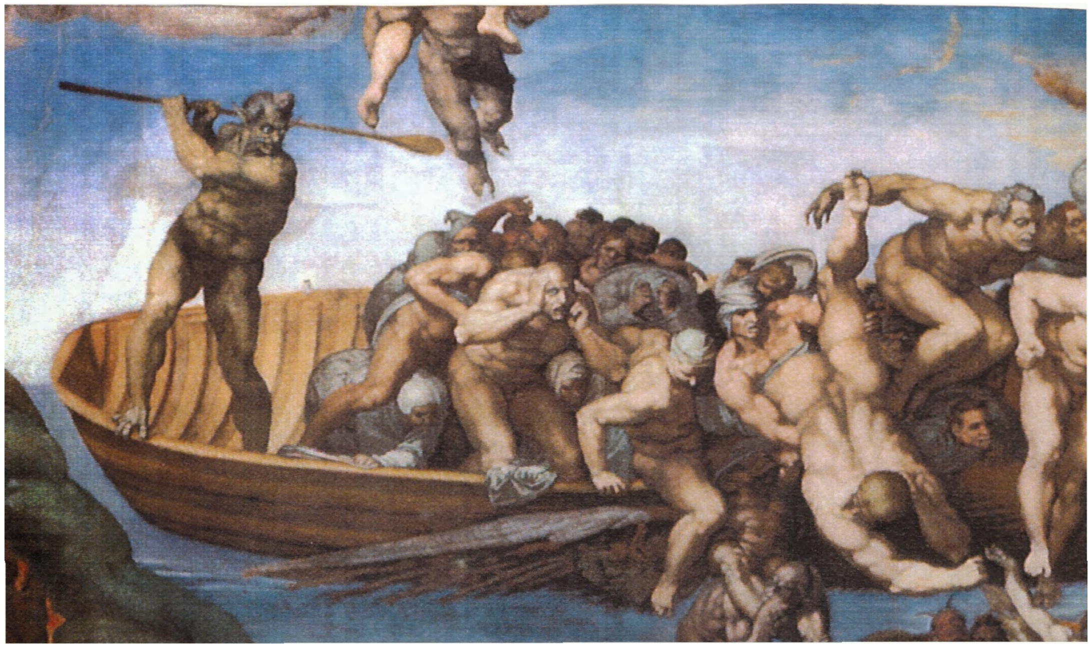
- System Classloader
- Core Classloader
- Plugin Classloaders
- Custom Classloaders

https://maven.apache.org/guides/mini/guide-maven-classloading.html

Classloading

This means that we need to find a way to move
the objects from an unaccessible (from outside) plugin classloader

We need a Charon/Caronte,
something able to move from one “world” to “another”



https://en.wikipedia.org/wiki/Charon_(mythology)

InseHacking Maven

We use a “Charon” object to enable the connection between outside (our API) and inside Maven (the executed plugin), in this way we are capable to read as a []bytes the live obejcts and move from the internal classloader to our external classloader and use for our needs.

In our implementation “Charon” works as a part of Maven

Trojans Horse https://it.wikipedia.org/wiki/Giandomenico_Tiepolo

InseHacking Maven

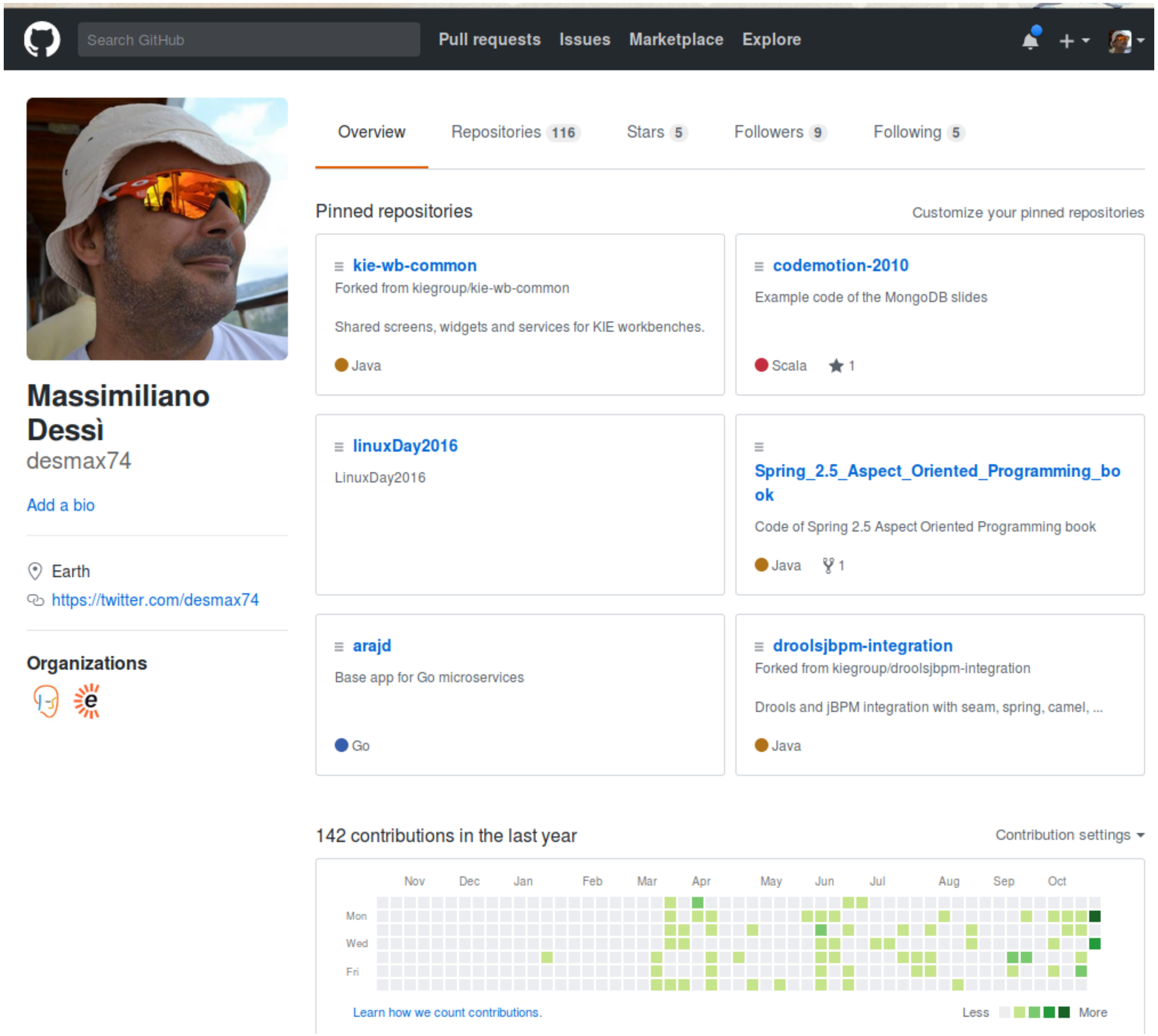
Computers are incredibly fast, accurate and stupid;
Humans are incredibly slow, inaccurate and brilliant;
Together they are powerful beyond imagination. -Albert Einstein

The implementation at the end is

- Flexible
- Extensible
- Open to the changes
- Could be changed on every request
- Stateless
- Ready for update Maven versions

the only informations retained are the links between builders and projects associated

Resources


Search GitHub | Pull requests | Issues | Marketplace | Explore

Overview | Repositories 116 | Stars 5 | Followers 9 | Following 5

Massimiliano Dessì
desmax74

Add a bio

Earth
https://twitter.com/desmax74

Organizations

Pinned repositories — Customize your pinned repositories

- **kie-wb-common** — Forked from kiegroup/kie-wb-common — Shared screens, widgets and services for KIE workbenches. — Java
- **codemotion-2010** — Example code of the MongoDB slides — Scala ★ 1
- **linuxDay2016** — LinuxDay2016
- **Spring_2.5_Aspect_Oriented_Programming_book** — Code of Spring 2.5 Aspect Oriented Programming book — Java, 1 fork
- **arajd** — Base app for Go microservices — Go
- **droolsjbpm-integration** — Forked from kiegroup/droolsjbpm-integration — Drools and jBPM integration with seam, spring, camel, ... — Java

142 contributions in the last year — Contribution settings

Learn how we count contributions. Less 🟩 More

https://github.com/desmax74

Q & A

Happy hacking

Have fun !

and thanks for your attention !