When Old Meets New: Turning Maven into a High Scalable, Resource Efficient, Cloud Ready Microservice.

Massimiliano Dessì
@desmax74

Speaker

Massimiliano Dessì has more than 18 years of experience in programming.

Co-founder of Java User Group Sardegna,

Manager of Google Developer Group Sardegna,

Author of Spring 2.5 AOP (Packt)

He works as a Senior Software Engineer for Red Hat in the Business Systems and Intelligence Group (BSIG), on the Knowledge Is Everything (KIE) projects,

[illegible]iari, Sardinia, Europe.

Starting point

Maven’s features

Maven produces artifacts
on file system using plugins,
basically is a producer of static
file, from Java
(and other languages)
projects.

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.

Out goals, Maven like never seen before

- From static file producer to in memory producer and exporter
- Incremental builds
- Reusable internal components to optimize memory footprint and time of execution
- Maven like a builder Daemon
- Configurable behaviour using a pipeline of decorators
- Stateless (when possible)
- Concurrent builds
- Cloud enabled
- Local and remote executions
- Plugins turned from FileSystem based to in memory based
- Async API to consume build result

Current requirements

In our Red Hat group we using a Maven plugin and

a build pipeline to

process rules and other projects managed by

Drools/Optaplanner/JBPM/Workbench

to produce files,

but are “dead” files on FS,

not an in memory

runtime objects.

JBCN Conf
BARCELONA JAVA USERS GROUP

New Requirements

Reads runtime objects inside Maven Plugins

and make they available to the “client” code

Maven API compatibility for easy upgrades of Maven

Maven Output in memory

even with concurrent builds

Memory efficient
(reuse the Plexus/Sisu container between build invocations on the same project)

Fast as possible between builds

“Live” objects

JBCN Conf
BARCELONA JAVA USERS GROUP

New Requirements

Use different maven repos

Global/User repo

Override of files compile and

then rollback to test changes

without affects the codebase

Cloud/Openshift/Kubernetes/Containers

JBCN Conf
BARCELONA JAVA USERS GROUP

Maven compatibility

Our first goal is to use Maven like a normal

API without braking changes to

enable easy updated of Maven.

Maven could be embedded/invoken as API using two libraries:

Maven Invoker open a new process separated from the caller

Maven Embedder works in the same process of the caller

the latter was our starting point

JBCN Conf
BARCELONA JAVA USERS GROUP

Our idea

Provide a “enhanced compiler”

with request-response

behaviour

simple as possible

in its use and configuration

and rich in terms of

objects in the response

compared to “plain” Maven

Request-Response

- Maven repo per request
- Project Path
- Maven cli args
- Settings file
- Unique compilation ID

```java
package org.kie.workbench.common.services.backend.compiler;

import ...

/**
 * Wrap a compilation request
 */
public interface CompilationRequest {

    AFCliRequest getKieCliRequest();

    WorkspaceCompilationInfo getInfo();

    String getMavenRepo();

    String[] getOriginalArgs();

    Map<String, Object> getMap();

    String getRequestUUID();

    Boolean skipAutoSourceUpdate();

    Boolean skipPrjDependenciesCreationList();

    Boolean getRestoreOverride();
}
```

Request-Response

Response:

Result of the build

In memory log messages

Dependencies list

Target folder elements

```java
package org.kie.workbench.common.services.backend.compiler;

import ...

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

    Boolean isSuccessful();

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

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

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

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

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

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

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

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

Request-Response behaviour

KieResponse:

Drools live objects extracted from the Maven plugin

Drools live objects generated on the fly (no .class file)

Classloaders contents

```java
package org.kie.workbench.common.services.backend.compiler.impl.kie;

import ...

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

    /**
     * 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
     */
    Map<String, byte[]> getProjectClassLoaderStore();

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

AFCompiler

The base of our compiler

```java
package org.kie.workbench.common.services.backend.compiler;

import ...

/**
 * Define the behaviour of a Compiler
 */
public interface AFCompiler<T extends CompilationResponse> {

    /**
     * Compile a project starting from the main POM
     */
    T compile(final CompilationRequest req);

    /**
     * Compile a project overriding or creating the elements in the Map and then revert this changes
     */
    T compile(final CompilationRequest req,
              final Map<Path, InputStream> override);

    Boolean cleanInternalCache();
}
```

Request-Response

We add objects to the

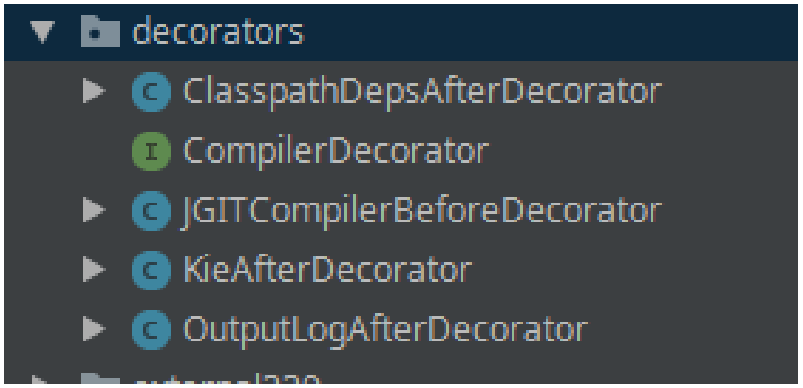
Maven result (a simple int)

and we use a pipeline of decorators

to add behaviours before and after compilation

Even Better, extensible

Chain of decorators
A decorator use the same contract of the AFCompiler



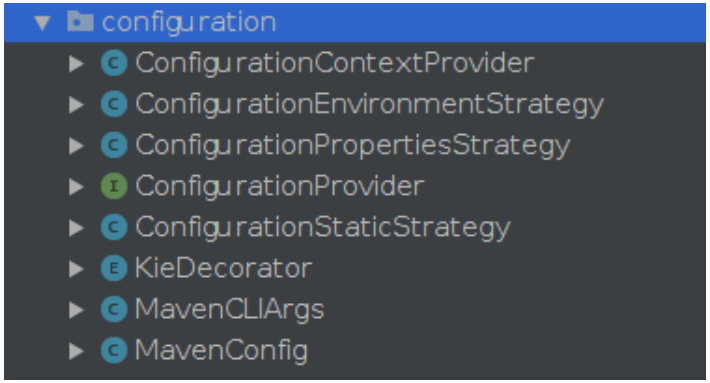
- Static factory
- Configurable per project

You could write your own before or after decorator and add to the pipeline

Configurable

Configuration strategies

- From environment (containers/pods)
- From files
- From classes



How to change the Maven internals ?

The first problem is how to open

the “sealed” Maven API,

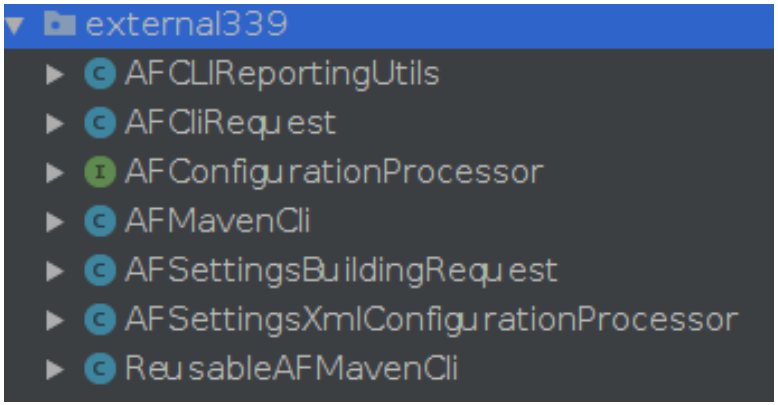
we change the use of Plexus/Sisu

we can reuse the same IoC container

per project saving a huge amount of time,

because the major part of the time in a Maven startup is the

creation of this container



Classloaders

Maven use classworlds to manage the various classloaders used
for its tasks

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

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

Classloaders

Classloaders

If we want read an object inside the plugin and export it

to the client code

we have to cross the barriers of those classworlds classloaders

Let’s see the trick

in our codebase

<code>

In memory Plugins

A cool side effects

when you cross the barriers

of the classloaders

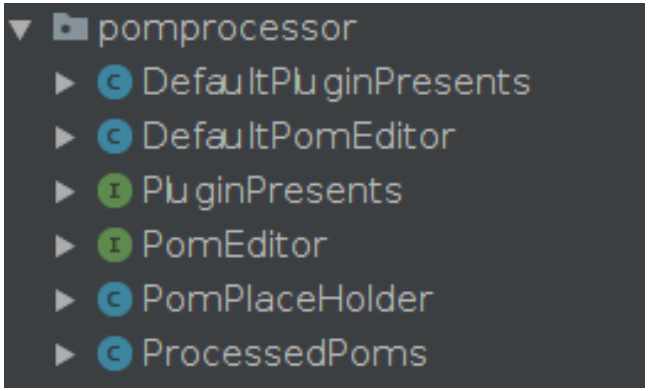
is the option

to turn the plugins impl

from FS to in memory.

Incremental compiler

To spent less time in front of a Maven build
we change on the fly the pom to add the Takari
Compiler (*) in every module of the project
tied with the compiler



* Our changes are already contributed in the Takari prjs since the 1.13.5

Pom Processor

The Compiler is aware
of the plugins in the POM
It changes on the fly
the Pom turning off the Default Compiler
and adds Takari and the Kie Plugin

Compiler service async

A lot of times
we could prefer
ask an async execution of a build
to avoid a blocking call,
especially if a build is long

Let’s go ahead : compiler service

We could use the compiler core
to enable a compiler service
enabling local Maven executors
or remote Maven executors
on other nodes/pods/containers/VMs

Local and remote Remote executors

We could demand a build to a remote

node/pod/container/vm

and reads a remote File System

because

the core compiler

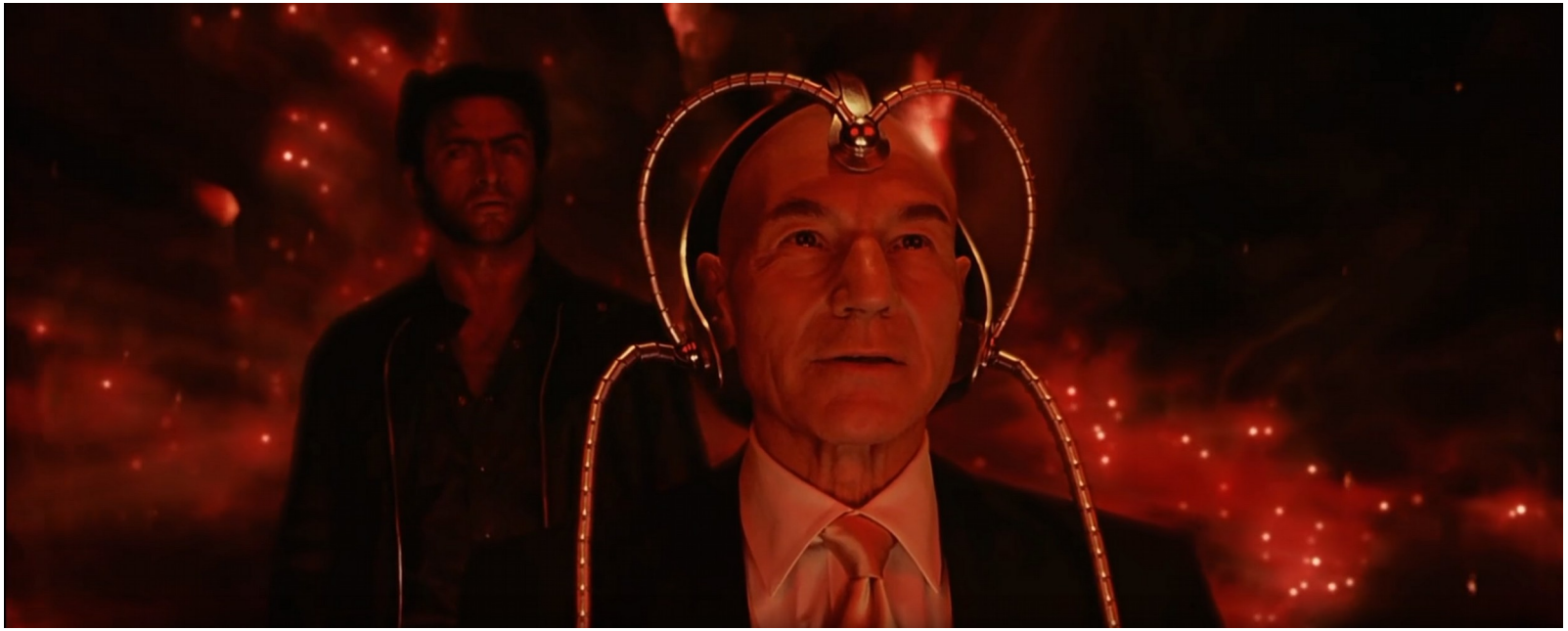
can use use a

JGITCompilerBeforeDecorator

to sync different “nodes”

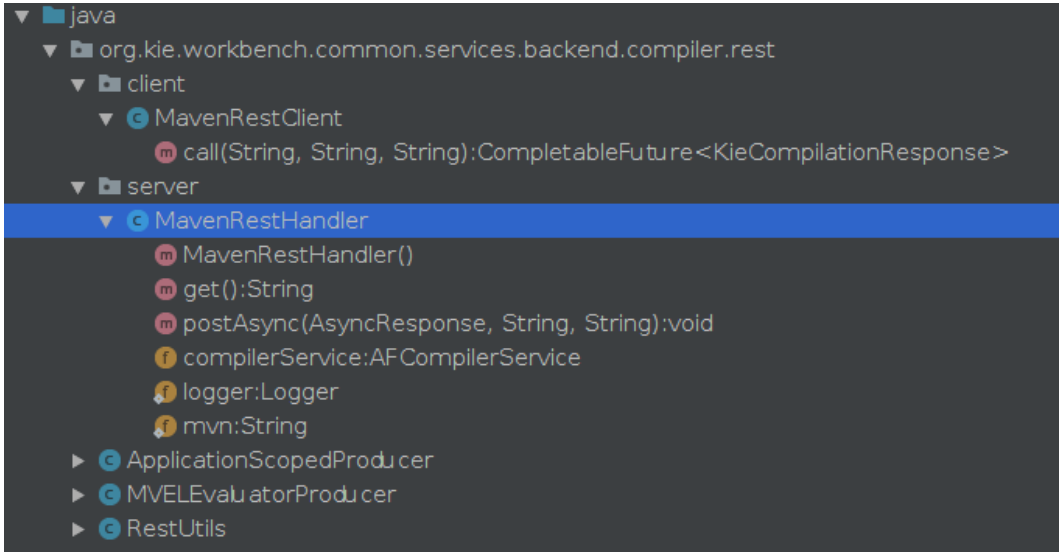
Local and remote Remote executors

The compiler service need to know if the current “machine” owns enough resources to run a local executors
or if is better to ask to a remote executor to satisfy the build.
To invoke a remote executors
It needs to know the address to reach the choosed instance



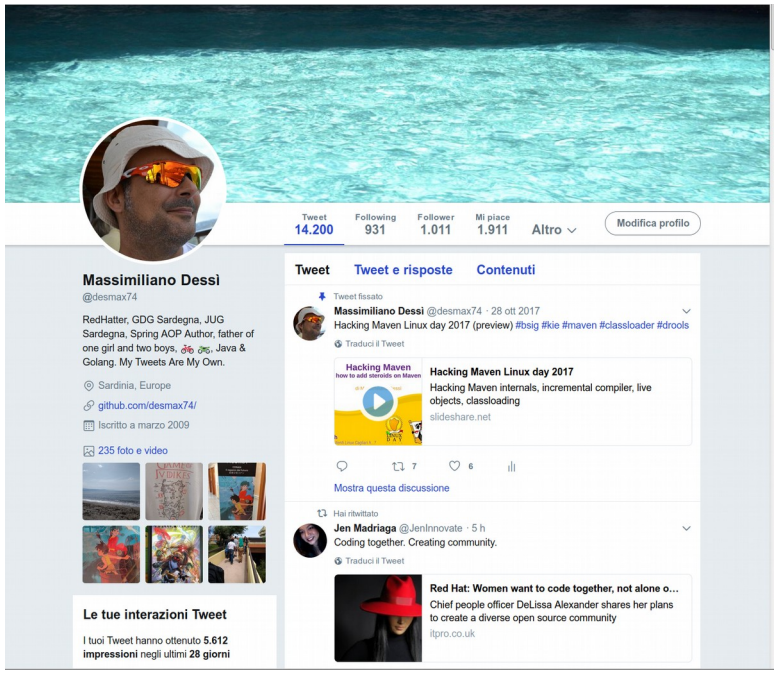
Rest Enpoint

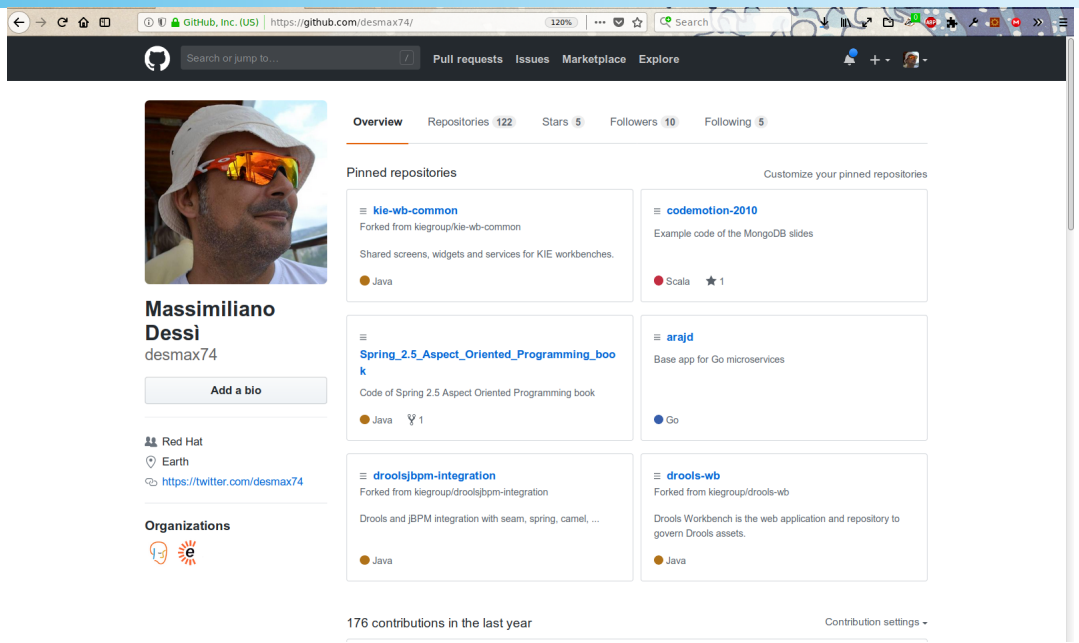
In some context is more easy
use an async Rest call
to execute a build on a remote
container/pod/machine.



Q & A

Resources


Massimiliano Dessì
@desmax74
RedHatter, GDG Sardegna, JUG Sardegna, Spring AOP Author, father of one girl and two boys, Java & Golang. My Tweets Are My Own.
Sardinia, Europe
github.com/desmax74/
Iscritto a marzo 2009
235 foto e video
Tweet 14.200
Following 931
Follower 1.011
Mi piace 1.911
Altro
Modifica profilo
Tweet
Tweet e risposte
Contenuti
Tweet fissato
Massimiliano Dessì @desmax74 · 28 ott 2017
Hacking Maven Linux day 2017 (preview) #bsig #kie #maven #classloader #drools
Traduci il Tweet
Hacking Maven Linux day 2017
Hacking Maven internals, incremental compiler, live objects, classloading
slideshare.net
Mostra questa discussione
Hai ritwittato
Jen Madriaga @JenInnovate · 5 h
Coding together. Creating community.
Traduci il Tweet
Red Hat: Women want to code together, not alone o…
Chief people officer DeLissa Alexander shares her plans to create a diverse open source community
itpro.co.uk
Le tue interazioni Tweet
I tuoi Tweet hanno ottenuto 5.612 impressioni negli ultimi 28 giorni


GitHub, Inc. (US) https://github.com/desmax74/
Search or jump to…
Pull requests
Issues
Marketplace
Explore
Overview
Repositories 122
Stars 5
Followers 10
Following 5
Massimiliano Dessì
desmax74
Add a bio
Red Hat
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
Spring_2.5_Aspect_Oriented_Programming_book
Code of Spring 2.5 Aspect Oriented Programming book
Java 1
arajd
Base app for Go microservices
Go
droolsjbpm-integration
Forked from kiegroup/droolsjbpm-integration
Drools and jBPM integration with seam, spring, camel, ...
Java
drools-wb
Forked from kiegroup/drools-wb
Drools Workbench is the web application and repository to govern Drools assets.
Java
176 contributions in the last year
Contribution settings


https://www.slideshare.net/desmax74
SlideShare
Search
Upload
Login
Signup
Home
Explore
Presentation Courses
PowerPoint Courses
by LinkedIn Learning
Massimiliano Dessì
Follow
37 SlideShares
112 Followers
0 Clipboards
Other
Software Senior Engineer
https://twitter.com/desmax74
Red Hatter, Author of book "Spring 2.5 Aspect Oriented Programming", published by Packt Publishing (2009)
Twitter
LinkedIn
Followers (112)
Massimiliano Dessì
Embed
Hacking Maven
how to add steroids on Maven
di Massimiliano Dessì
GULCh
Gruppo Utenti Linux Cagliari h...?
LINUX DAY
1 of 22
Presentations 37
Documents 0
Infographics 0
Videos 0
Hacking Maven Linux day 2017 507

Links

https://twitter.com/desmax74

https://github.com/desmax74/

https://www.slideshare.net/desmax74

Thanks for your attention !

And HAVE FUN !