

A Journey in Software Development

An overview of methods and tools

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

Material

<http://mathieuacher.com/teaching/PDL/>

Agenda

Point projet

- Compréhension (domaine, technique, besoins) et Cahier des charges (EX)
- Implémentation (SP); c'est le moment de commencer le travail technique pour:
 1. Mieux comprendre le projet
 2. Prototyper des solutions (et raffiner EX)
 3. Organiser le travail de groupe

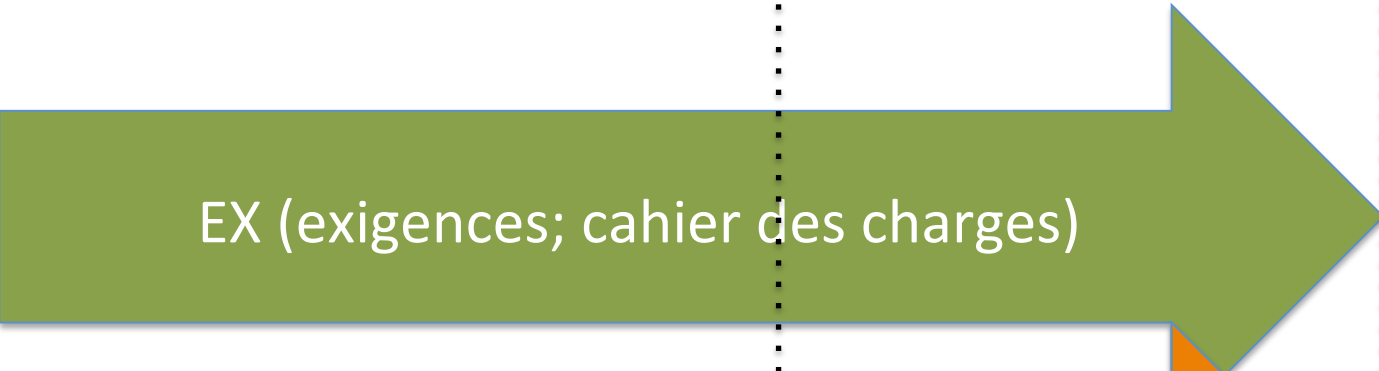
(Rappel de) quelques outils et techniques

10 novembre

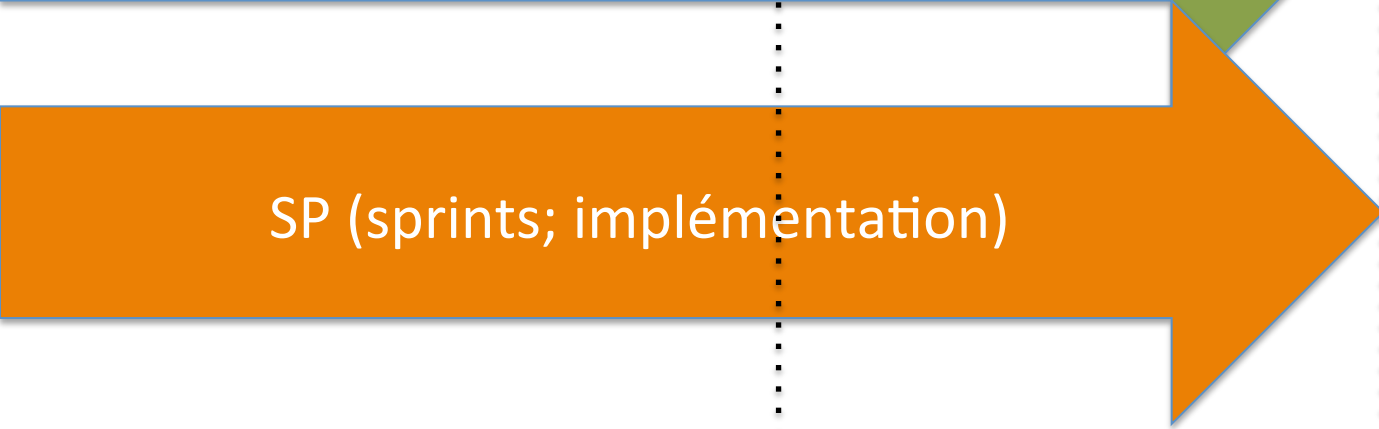
18 décembre



EX (exigences; cahier des charges)



SP (sprints; implémentation)



PR
(présentation)



Multi-Tools and Languages



Visual Basic



Code::Blocks Studio

eclipse



Microsoft Visual Studio



maven

mongoDB



SUBVERSION

git

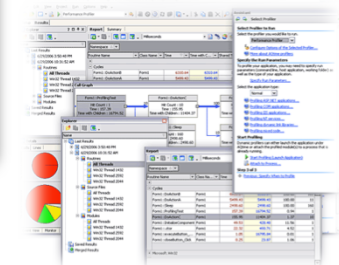


Logging Service



<APACHE ANT>

JUnit.org



Today

- Build System (maven)
- Manage your source code
- Refactoring
- Logging, Debugging and Test
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

Today

- Tools and techniques are useful for your project and have to be used
 - Logging, Test, Build system, IDE
- Objectives:
 - Why these tools/techniques exist?
 - Presentation
 - First: independent of any technology
 - and then describe specific solutions

Today

- **Build System (maven)**
- Manage your source code
- Refactoring
- Logging, Debugging and Test
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

Maven

```
PACKAGE      = package
VERSION      = `date "+%Y.%m%d%"`
RELEASE_DIR  = ..
RELEASE_FILE = $(PACKAGE)-$(VERSION)

# Notice that the variable LOGNAME comes from the environment in
# POSIX shells.
#
# target: all - Default target. Does nothing.
all:
    echo "Hello $(LOGNAME), nothing to do by default"
    # sometimes: echo "Hello ${LOGNAME}, nothing to do by default"
    echo "Try 'make help'"

# target: help - Display callable targets.
help:
    egrep "^# target:" [Mm]akefile

# target: list - List source files
list:
    # Won't work. Each command is in separate shell
    cd src
    ls

    # Correct, continuation of the same shell
    cd src; \
    ls
```

Make/Makefile

Original problem: compiling your source code files can be tedious!

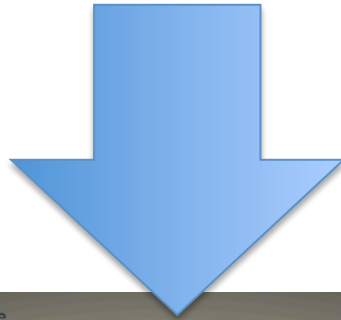
- several source files

- type the compiling commands **s** everytime

Make for increasing automation, avoiding accidental complexity, and have more flexibility when « compiling » projects

(initial release in 1977)

Make



```
PACKAGE = package
VERSION = `date +%Y.%m%d%` `
RELEASE_DIR = ..
RELEASE_FILE = ${PACKAGE}-${VERSION}

# Notice that the variable LOGNAME comes from the environment in
# POSIX shells.
#
# target: all - Default target. Does nothing.
all:
    echo "Hello $(LOGNAME), nothing to do by default"
    # sometimes: echo "Hello ${LOGNAME}, nothing to do by default"
    echo "Try 'make help'"

# target: help - Display callable targets.
help:
    egrep "^# target:" [Mm]akefile

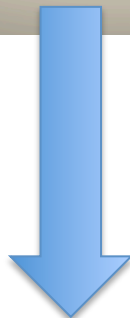
# target: list - List source files
list:
    # Won't work. Each command is in separate shell
    cd src
    ls

    # Correct, continuation of the same shell
    cd src; \
    ls
```

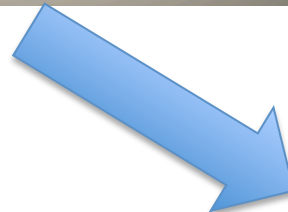
Makefile



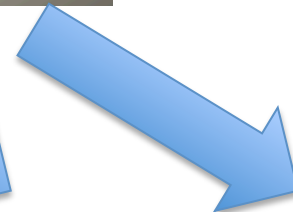
help



compile



gendoc



list

.....

Compile chain

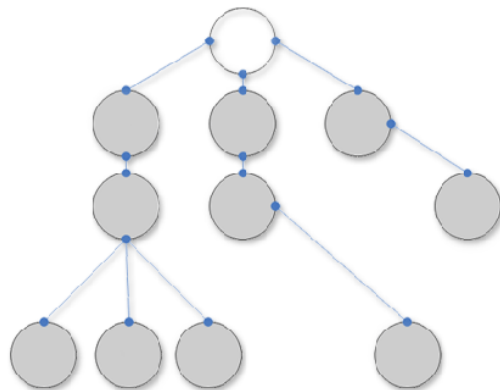
- Sometimes hidden in the IDE
 - But generally speaking, you need to master your “compile” chain
- Tools
 - make, gmake, nmake (Win),
 - Apache ANT, Apache **MAVEN**, Freshmeat 7Bee ...
- To **automate**:
 - pre-compilation, obfuscation, verification
 - generation of .class and .jar
 - normal, tracing, debug, ...
 - documentation generation
 - « stubs » generation (rmic, idl2java, javacard ...)
 - test
 - 3rd party libraries/dependencies
 - ... And a **combination** of all these tasks

What is Maven?

A build tool

```
C:\WINDOWS\system32\cmd.exe
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-alpha-4/wagon-1.0-alpha-4.pom
3K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider-api/1.0-alpha-4/wagon-provider-api-1.0-alpha-4.jar
45K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-artifact-manager/2.0-alpha-3/maven-artifact-manager-2.0-alpha-3.jar
32K downloaded
[INFO] Installing: install
[INFO] Installing C:\my-app\target\my-app-1.0-SNAPSHOT.jar to C:\Documents and Settings\Administrator\TOSHIBA\.m2\repository\com\mycompany\app\my-app\1.0-SNAPSHOT\my-app-1.0-SNAPSHOT.jar
[INFO]
-----
[INFO] BUILD SUCCESSFUL
[INFO]
[INFO] Total time: 47 seconds
[INFO] Finished at: Fri Jun 24 16:24:10 PDT 2005
[INFO] Final Memory: 2M/5M
[INFO]
C:\my-app>
```

A dependency management tool



A documentation tool



Apply patterns to project build infrastructure

Maven is really a process of applying **patterns** to a build infrastructure in order to provide a coherent view of software projects.

Provides a way to help with managing:

- Builds
- Documentation
- Reporting
- Dependencies
- Software Configuration Management
- Releases

Objectives

- Make the development process visible or transparent
- Provide an easy way to see the health and status of a project
- Decreasing training time for new developers
- Bringing together the tools required in a uniform way
- Preventing inconsistent setups
- Providing a standard development infrastructure across projects
- Focus energy on writing applications

Benefits

- Standardization
- Fast and easy to set up a powerful build process
- Greater momentum vs. Ant - it is now becoming legacy and not moving fast ahead.
- Dependency management (automatic downloads)
- Project website generation, Javadoc
- Repository management
- Extensible architecture

Maven and POM

aka project's configurations

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Kind of packaging



Maven facilities and lifecycle

validate: validate the project is correct and all necessary information is available

compile: compile the source code of the project

test: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

package: take the compiled code and package it in its distributable format, such as a JAR.

integration-test: process and deploy the package if necessary into an environment where integration tests can be run

verify: run any checks to verify the package is valid and meets quality criteria

install: install the package into the local repository, for use as a dependency in other projects locally

deploy: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

clean: cleans up artifacts created by prior builds

site: generates site documentation for this project

Build the Project

```
mvn package
```

Generating the Site

```
mvn site
```

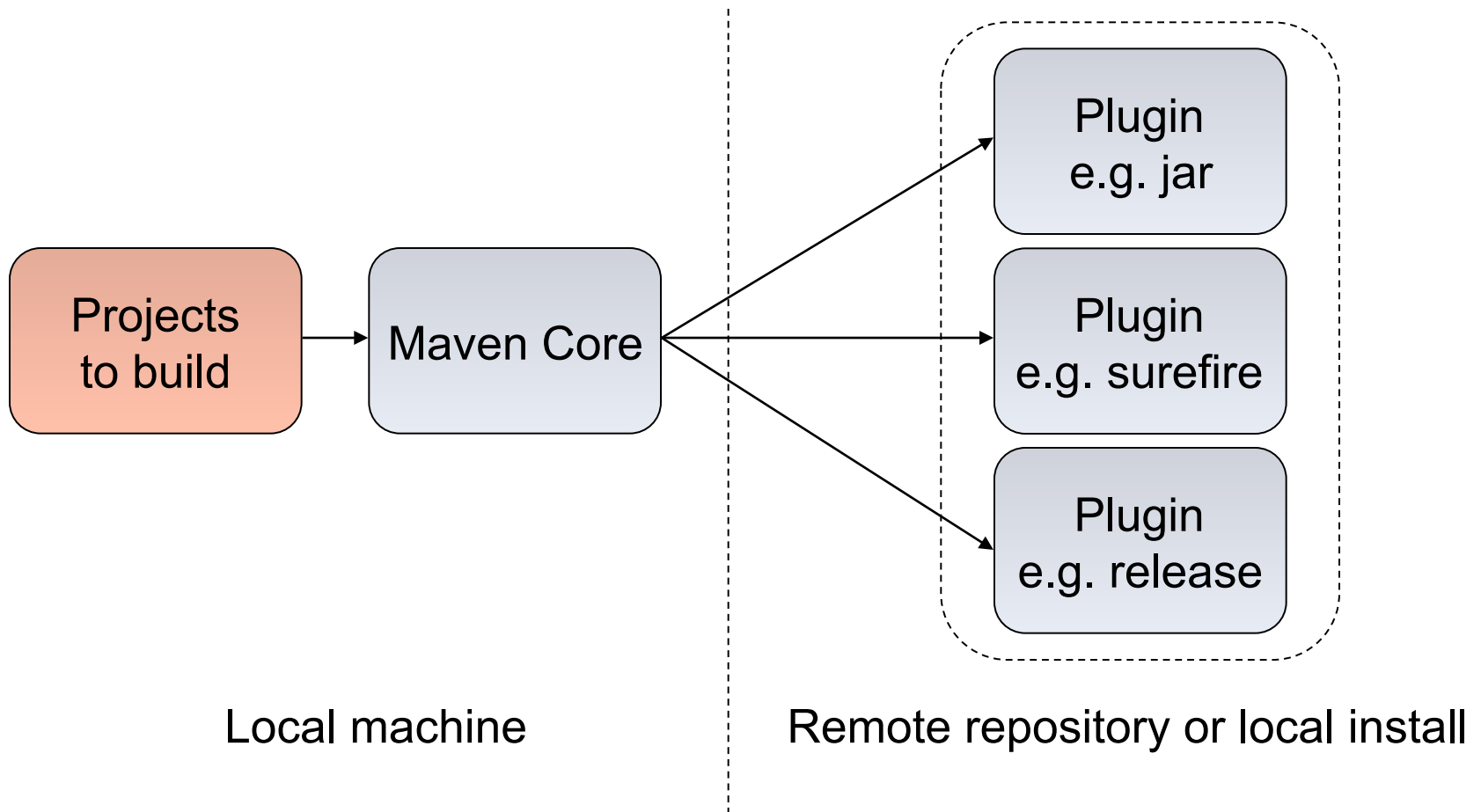
Maven

- Abstract project model (POM)
 - Object oriented, inheritance
 - Separation of concerns
- Default lifecycle
 - Default state (goals) sequence
 - plugins depend on states
- Give a project « standard » structure
 - Standard naming conventions
 - Standard lifecycle
- Automatic handling of dependencies between projects
 - Including updates
- Project repositories
 - public or private, local or remotes
 - caching and proxy
- Extensible via external plugins

Maven plugins

- Core
 - clean, compiler, deploy, install, resources, site, surefire, verifier
- Packaging
 - ear, ejb, jar, rar, war, bundle (OSGi)
- Reporting
 - changelog, changes, checkstyle, clover, doap, docck, javadoc, jxr, pmd, project-info-reports, surefire-report
- Tools
 - ant, antrun, archetype, assembly, dependency, enforcer, gpg, help, invoker, one (interop Maven 1), patch, plugin, release, remote-resource, repository, scm
- IDEs
 - eclipse, netbeans, idea
- Others
 - exec, jdepend, castor, cargo, jetty, native, sql, taglist, javacc, obr
...

Maven Architecture



Common project metadata format

- POM = Project Object Model = pom.xml
- Contains metadata about the project
 - Location of directories, Developers/Contributors, Issue tracking system, Dependencies, Repositories to use, etc
- Example:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.codehaus.cargo</groupId>
  <artifactId>cargo-core-api-container</artifactId>
  <name>Cargo Core Container API</name>
  <version>0.7-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies/>
  <build/>
  [...]

```

Minimal POM

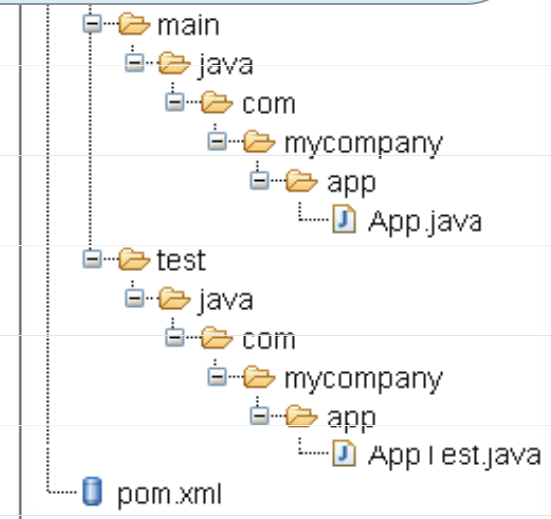
Use Inheritance

Standard directory organization

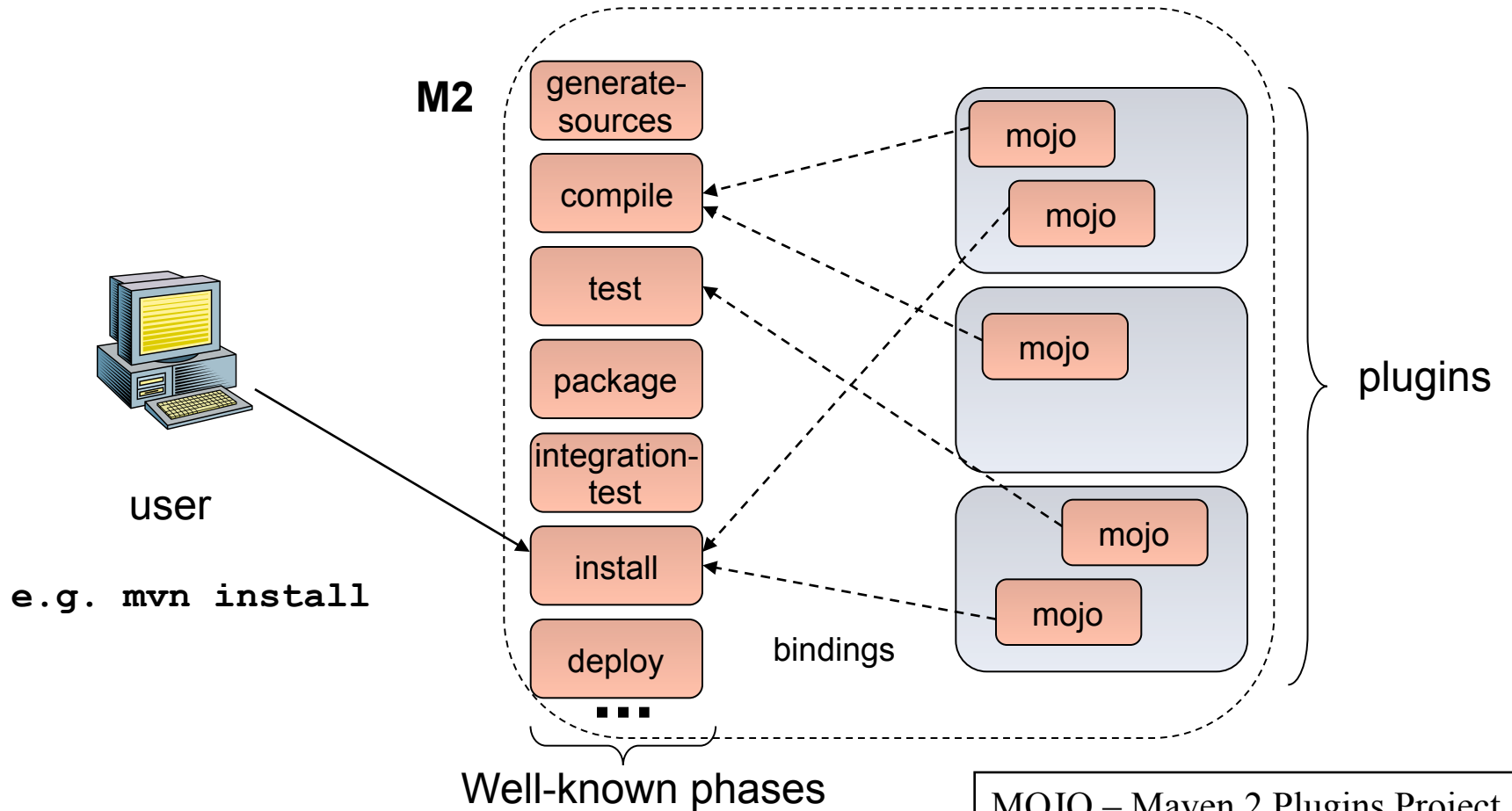
- Having a common directory layout would allow for users familiar with one Maven project to work with other Maven projects

Convention over configuration

| | |
|--------------------|----------------------------|
| src/main/java | |
| src/main/resources | |
| src/main/filters | |
| src/main/assembly | Assembly descriptors |
| src/main/config | Configuration files |
| src/main/webapp | Web application resources |
| src/test/java | Test source code |
| src/test/resources | Test resources |
| src/test/filters | Test resource filter files |
| src/site | Site |
| LICENSE.txt | Project's license |
| README.txt | Project's readme |

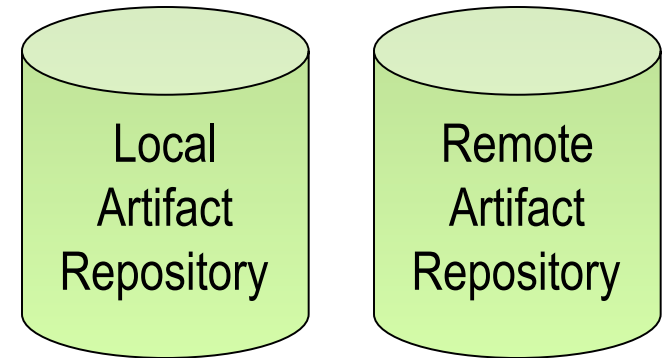


Common way to build applications



Artifact repositories (1/3)

- Used to store all kind of artifacts
 - JARs, EARs, WARs, NBMs, EJBs, ZIPs, plugins, ...
- All project interactions go through the repository
 - No more relative paths!
 - Easy to share between team

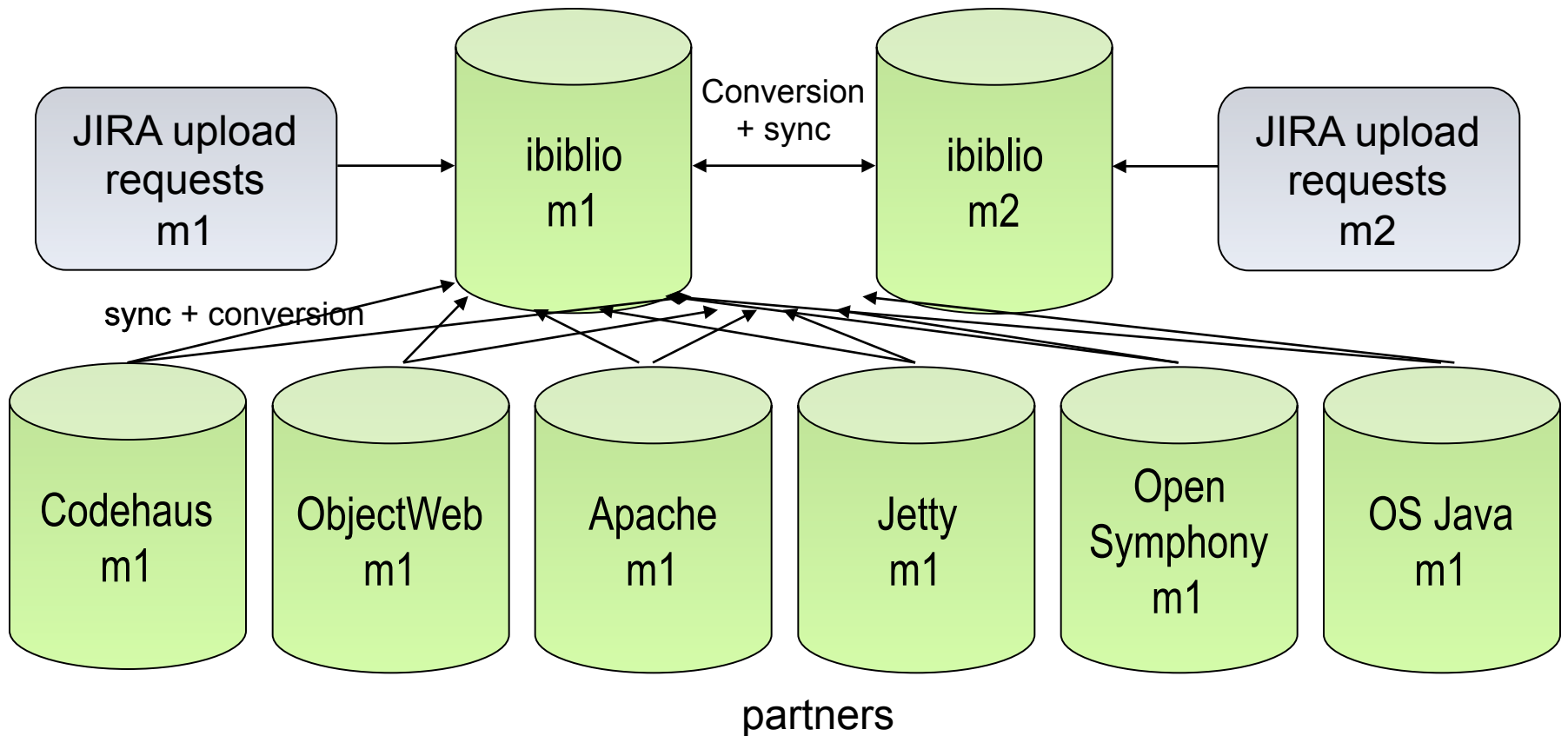


e.g. <http://ibiblio.org/maven2>

```
<repositories>
  <repository>
    <id>maven2-snapshot</id>
    <releases>
      <enabled>>true</enabled>
    </releases>
    <name>Maven Central Development Repository</name>
    <url>http://snapshots.maven.codehaus.org/maven2</url>
    <layout>legacy|default</layout>
  </repository>
</repositories>
```

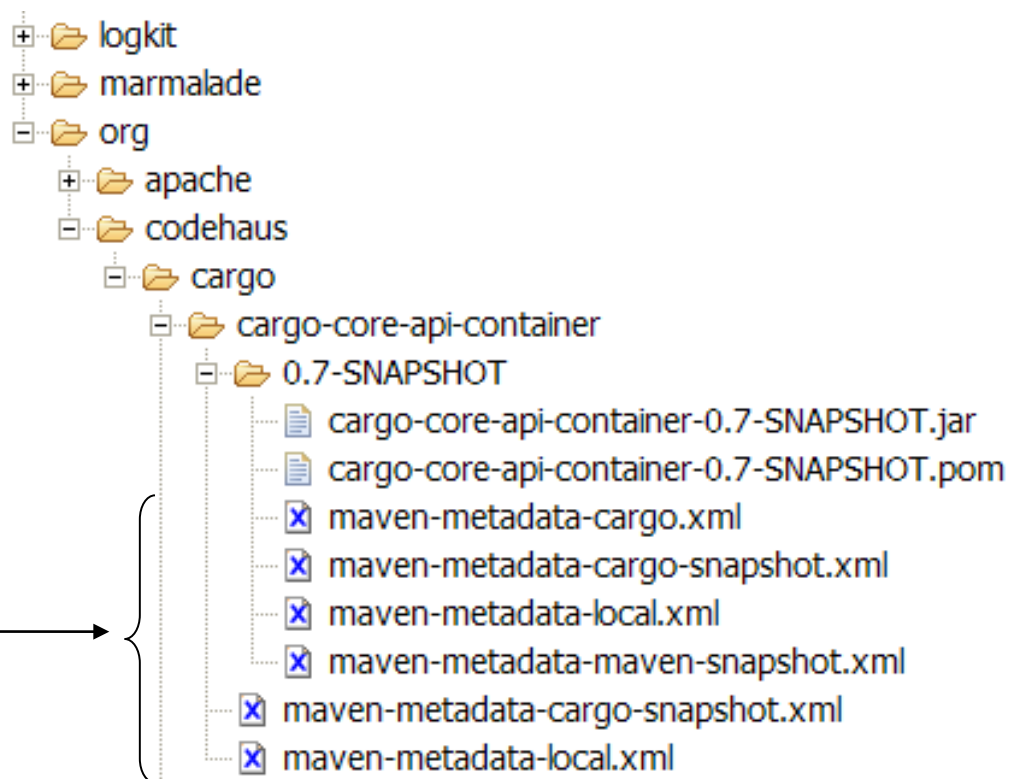
Artifact repositories (2/3)

- Some public remote repositories



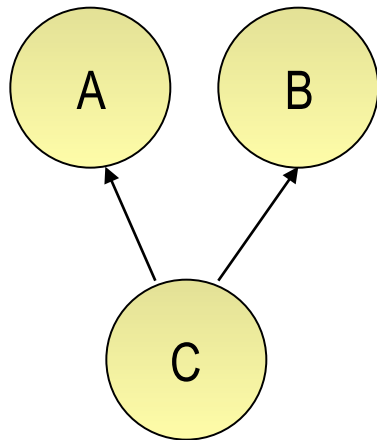
Artifact repositories (3/3)

- Hierarchical structure
- Automatic plugin download
- Plugins are read directly from the repository
- Configurable strategies for checking the remote repositories for updates
 - Daily check by default for plugin and ranges updates
- Remote repositories contain Metadata information
 - Releases, latest, and more to come



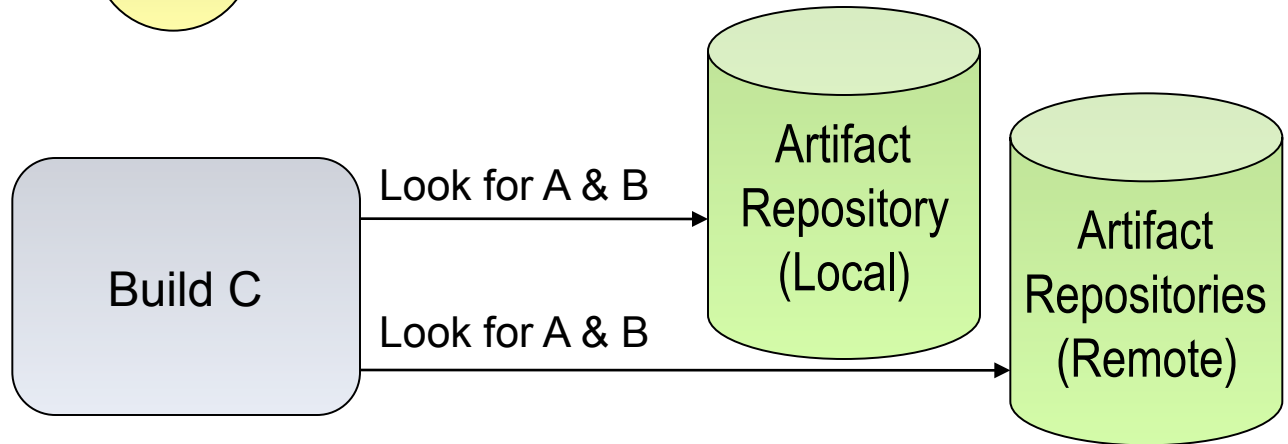
Dependency management (1/2)

- Maven uses binary dependencies



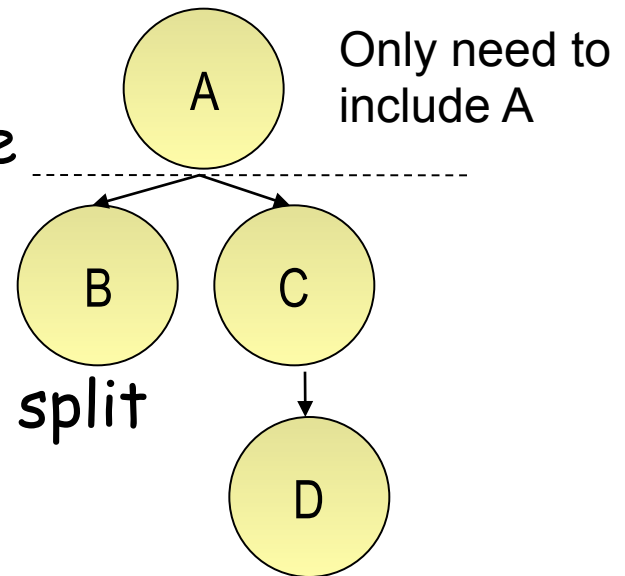
```
<dependencies>
  <dependency>
    <groupId>com.acme</groupId>
    <artifactId>B</artifactId>
    <version>[1.0,)</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

« any version after 1.0 »



Dependency management (2/2)

- Transitive dependencies
 - Possibility to exclude some dependencies
 - Need good metadata
 - Ideally projects should be split
- SNAPSHOT handling
 - Always get latest
- Automatic dependency updates
 - By default every day



Installation and Setup

- <http://maven.apache.org/>
- Add Maven's bin directory to PATH
- Ensure JAVA_HOME is set to SDK
- Run `mvn -version` to test install

```
C:\Documents and Settings\alina>mvn -version
Maven version: 3.0.4
Java version: 1.6.0_30
```


Maven plugin for JAVA IDE

- Maven plugins exists for
 - Eclipse
 - IntelliJ
 - NetBeans
 - ...

Installing JARs to local repository

- Sometimes you need to put some specific JARs in your local repository for use in your builds
- The JARs must be placed in the correct place in order for it to be correctly picked up by Maven
- To install a JAR in the local repository use the following command:

```
mvn install:install-file -Dfile=<path-to-file> -DgroupId=<group-id> \
-DartifactId=<artifact-id> -Dversion=<version> -Dpackaging=jar
```

- Now can include dependency in pom.xml:

```
<dependency>
  <groupId><group-id></groupId>
  <artifactId><artifact-id></artifactId>
  <version><version></version>
</dependency>
```

Overview of common Goals

- **clean** - clean the current project
- **validate** - validate the project is correct and all necessary information is available
- **compile** - compile the source code of the project
- **test** - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package** - take the compiled code and package it in its distributable format, such as a JAR
- **integration-test** - process and deploy the package if necessary into an environment where integration tests can be run
- **install** - install the package into the local repository, for use as a dependency in other projects locally
- **deploy** - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects

More stuff

- Automatically generate reports, diagrams, and so on through Maven / the project site
- Internationalization - create different language project websites
- Create projects within projects (more pom.xml files inside sub dirs \jars), with different build stats and so on
- Maven can make .war files, EJBs, etc.

Using Maven Plugins

- Whenever you want to customise the build for a Maven project, this is done by adding or reconfiguring plugins
- For example, configure the Java compiler to allow JDK 5.0 sources

- Plugins in Maven 3.0 look much like a dependency

```
...
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
  </plugins>
</build>
...
```

Maven Plugins

- **AlmostPlainText**
- **Maven Axis**
- **Maven Cobertura**
- **Maven DB2**
- **Dbunit**
- **Debian Package**
- **Maven DotUml**
- **Doxygen**
- **Maven Files**
- **FindBugs**
- **Maven flash**
- **Help**
- **Maven IzPack**
- **Java Application**
- **Maven JAVANCSS**
- **Maven JAXB**
- **JUNITPP**
- **Kodo**
- **Maven Macker**
- **SDocBook**
- **Sourceforge**
- **Maven SpringGraph**
- **RPM Plugin**
- **Runtime Builder**
- **Strutsdoc**
- **Tasks**
- **Maven Transform**
- **Maven UberDist**
- **Maven Vignette**
- **WebSphere 4.0**
- **WebSphere 5 (5.0/5.1)**
- **Maven WebLogic**
- **Canoo WebTest**
- **Wiki**
- **Word to HTML**
- **XML Resume**
- **Maven DotUml**
- **Middlegen**
- **Maven News**

Archetypes

- For reuse, create archetypes that work as project templates with build settings, etc
- An archetype is a project, with its own pom.xml
- An archetype has a descriptor called archetype.xml
- Allows easy generation of Maven projects

Good things about Maven

- Standardization
- Reuse
- Dependency management
- Build lifecycle management
- Large existing repository
- IDE aware
- One directory layout
- A single way to define dependencies
- Setting up a project is really fast
- Transitive dependencies
- Common build structure
- Use of remote repository
- Web site generation
- Build best practices enforcement
- Automated build of application
- Works well with distributed teams
- All artifacts are versioned and are stored in a repository
- Build process is standardized for all projects
- A lot of goals are available
- It provides quality project information with generated site
- Easy to learn and use
- Makes the build process much easier at the project level
- Promotes modular design of code

References

- **Maven Home**
<http://maven.apache.org/>
- **Maven Getting Started Guide**
<http://maven.apache.org/guides/getting-started/index.html>
- **Steps for creating a Maven-based Website**
http://www.javaworld.com/javaworld/jw-02-2006/jw-0227-maven_p.html
/
- **Maven Integration for Eclipse**
<http://m2eclipse.codehaus.org/>

Example and Demonstration

An example: getting started with OpenCompare

<https://github.com/OpenCompare/getting-started>

The screenshot shows a GitHub repository page for 'OpenCompare / getting-started'. At the top, there are navigation links for 'Unwatch' (3), 'Star' (0), and 'Fork' (0). Below this, the repository name and a link to 'Edit' are shown. A summary bar indicates '3 commits', '1 branch', '0 releases', and '1 contributor'. The current branch is 'master', and the repository name 'getting-started' is highlighted. A commit history table follows, listing files like '.gitignore', '.travis.yml', 'LICENSE', 'README.md', and 'pom.xml' with their commit messages and dates. Below the table is a preview of the 'README.md' file, which contains the title 'getting-started' and the subtitle 'Examples for using OpenCompare API and services'. On the right side, there is a sidebar with navigation links for 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. At the bottom of the sidebar, there are options to clone the repository via HTTPS, SSH, or Subversion, and buttons for 'Clone in Desktop' and 'Download ZIP'.

OpenCompare / [getting-started](#) Unwatch 3 Star 0 Fork 0

Examples for using OpenCompare API and services — Edit

3 commits 1 branch 0 releases 1 contributor

Branch: **master** [getting-started](#) / +

add .gitignore and .travis.yml file
gbecan authored 9 days ago latest commit 3107ed8f57

| File | Commit Message | Time |
|-------------|-------------------------------------|------------|
| pcms | create getting-started project | 9 days ago |
| src | create getting-started project | 9 days ago |
| .gitignore | add .gitignore and .travis.yml file | 9 days ago |
| .travis.yml | add .gitignore and .travis.yml file | 9 days ago |
| LICENSE | Initial commit | 9 days ago |
| README.md | Initial commit | 9 days ago |
| pom.xml | create getting-started project | 9 days ago |

README.md

getting-started

Examples for using OpenCompare API and services

Code

- Issues 0
- Pull requests 0
- Wiki
- Pulse
- Graphs
- Settings

HTTPS clone URL
`https://github.com/`

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP

Source Git Repository

Enter the location of the source repository.



Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

Store in Secure Store



< Back

Next >

Cancel

Finish

Open File...
Close ⌘W
Close All ⇧⌘W
Save ⌘S
Save As...
Save All ⇧⌘S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To ▶
Print... ⌘P
Switch Workspace ▶
Restart
Import...
Export...
Properties ⌘I

1 pom.xml [getting-started]
2 MyPCMPrinterTest.java [getting-started]
3 MyPCMPrinter.java [getting-started/...]
4 VideoGen.xtext [org.xtext.example.y...

Import

Select Existing Maven Projects

Select an import source:

type filter text

- General
- CVS
- Git
- Install
- Maven
 - Check out Maven Projects from SCM
 - Existing Maven Projects**
 - Install or deploy an artifact to a Maven repository
 - Materialize Maven Projects from SCM
- Oomph
- Plug-in Development
- Run/Debug
- Team
- XML

< Back Next > Cancel Finish

Import Maven Projects

Maven Projects

Select Maven projects

Root Directory: /Users/macher1/git/getting-started Browse...

Projects:

| | |
|-------------------------------------|---|
| <input checked="" type="checkbox"/> | /pom.xml org.opencompare:getting-started:1.0-SNAPSHOT:jar |
|-------------------------------------|---|

Select All
Deselect All
Select Tree
Deselect Tree












Add project(s) to working set
getting-started

Advanced

? < Back Next > Cancel Finish

Package Explorer

JUnit

- ▼  > getting-started [getting-started master]
 - ▶  src/main/java
 - ▶  src/test/java
 - ▶  JRE System Library [JavaSE-1.8]
 - ▶  Maven Dependencies
 - ▶  pcms
 - ▶  src
 -  target
 -  LICENSE
 -  pom.xml
 -  README.md

```
getting-started/pom.xml 83
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>org.opencompare</groupId>
8      <artifactId>getting-started</artifactId>
9      <version>1.0-SNAPSHOT</version>
10
11     <dependencies>
12
13         <!-- Dependencies to Java PCM API and its implementation -->
14         <dependency>
15             <groupId>org.opencompare</groupId>
16             <artifactId>api-java</artifactId>
17             <version>0.5</version>
18         </dependency>
19
20         <dependency>
21             <groupId>org.opencompare</groupId>
22             <artifactId>api-java-impl</artifactId>
23             <version>0.5</version>
24         </dependency>
25
26         <!-- Dependency to Junit for testing our project -->
27         <dependency>
28             <groupId>junit</groupId>
29             <artifactId>junit</artifactId>
30             <version>4.11</version>
31             <scope>test</scope>
32         </dependency>
33
34     </dependencies>
35
36     <build>
37         <pluginManagement>
38             <plugins>
39                 <plugin>
40                     <groupId>org.apache.maven.plugins</groupId>
41                     <artifactId>maven-compiler-plugin</artifactId>
42                     <version>2.1</version>
43                     <configuration>
44                         <source>1.8</source>
45                         <target>1.8</target>
46                     </configuration>
47                 </plugin>
48             </plugins>
49         </pluginManagement>
50     </build>
51
52 </project>
```

pom.xml

getting-started/pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>org.opencompare</groupId>
8     <artifactId>getting-started</artifactId>
9     <version>1.0-SNAPSHOT</version>
```

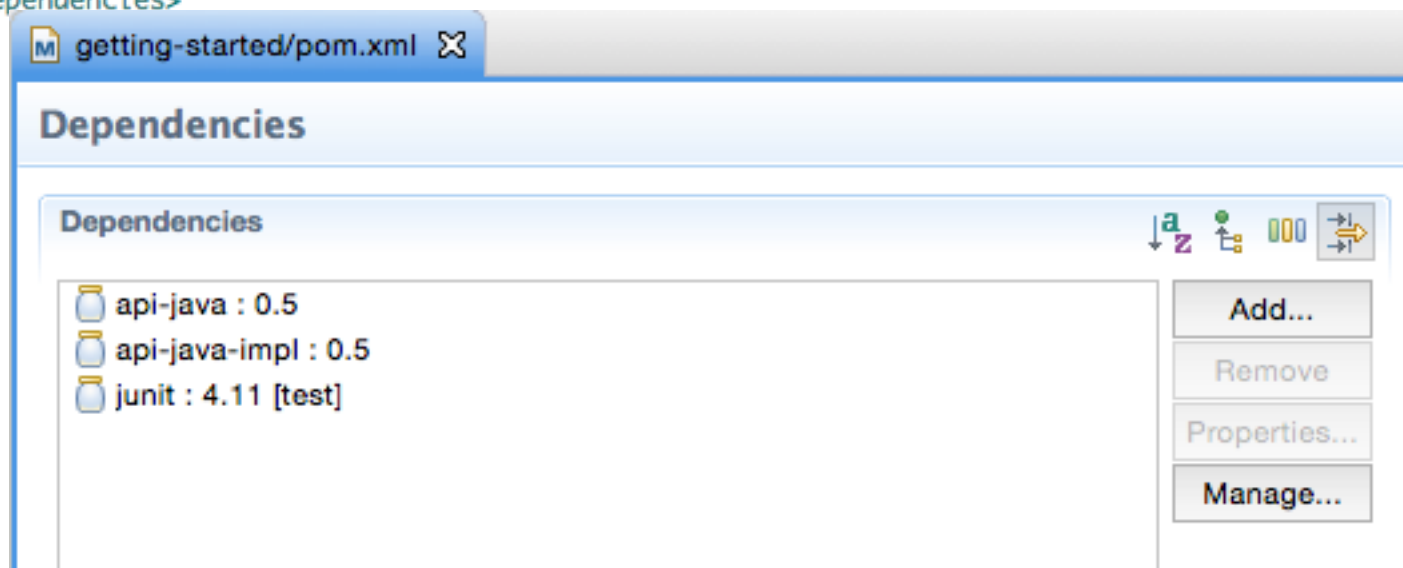
pom.xml


```
11 <dependencies>
12
13 <!-- Dependencies to Java PCM API and its implementation -->
14 <dependency>
15     <groupId>org.opencompare</groupId>
16     <artifactId>api-java</artifactId>
17     <version>0.5</version>
18 </dependency>
19
20 <dependency>
21     <groupId>org.opencompare</groupId>
22     <artifactId>api-java-impl</artifactId>
23     <version>0.5</version>
24 </dependency>
25
26 <!-- Dependency to Junit for testing our project -->
27 <dependency>
28     <groupId>junit</groupId>
29     <artifactId>junit</artifactId>
30     <version>4.11</version>
31     <scope>test</scope>
32 </dependency>
33
34 </dependencies>
```












pom.xml

```
11 <dependencies>
12
13 <!-- Dependencies to Java PCM API and its implementation -->
14 <dependency>
15     <groupId>org.opencompare</groupId>
16     <artifactId>api-java</artifactId>
17     <version>0.5</version>
18 </dependency>
19
20 <dependency>
21     <groupId>org.opencompare</groupId>
22     <artifactId>api-java-impl</artifactId>
23     <version>0.5</version>
24 </dependency>
25
26 <!-- Dependency to Junit for testing our project -->
27 <dependency>
28     <groupId>junit</groupId>
29     <artifactId>junit</artifactId>
30     <version>4.11</version>
31     <scope>test</scope>
32 </dependency>
33
34 </dependencies>
```

pom.xml














▼ Maven Dependencies

- ▶  api-java-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/api-java/0.5
- ▶  opencsv-3.3.jar - /Users/macher1/.m2/repository/com/opencsv/opencsv/3.3
- ▶  commons-lang3-3.3.2.jar - /Users/macher1/.m2/repository/org/apache/commons/commons-lang3/3.3.2
- ▶  jsoup-1.8.2.jar - /Users/macher1/.m2/repository/org/jsoup/jsoup/1.8.2
- ▶  api-java-impl-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/api-java-impl/0.5
- ▶  model-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/model/0.5
- ▶  org.kevoree.modeling.microframework-3.5.12.jar - /Users/macher1/.m2/repository/org/kevoree/modeling
- ▶  kotlin-stdlib-0.8.11.jar - /Users/macher1/.m2/repository/org/jetbrains/kotlin/kotlin-stdlib/0.8.11
- ▶  kotlin-runtime-0.8.11.jar - /Users/macher1/.m2/repository/org/jetbrains/kotlin/kotlin-runtime/0.8.11
- ▶  junit-4.11.jar - /Users/macher1/.m2/repository/junit/junit/4.11
- ▶  hamcrest-core-1.3.jar - /Users/macher1/.m2/repository/org/hamcrest/hamcrest-core/1.3

Why opencsv, jsoup, etc?

Maven Dependencies

- ▶  api-java-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/api-java/0.5
- ▶  opencsv-3.3.jar - /Users/macher1/.m2/repository/com/opencsv/opencsv/3.3
- ▶  commons-lang3-3.3.2.jar - /Users/macher1/.m2/repository/org/apache/commons/commons-lang3/3.3.2
- ▶  jsoup-1.8.2.jar - /Users/macher1/.m2/repository/org/jsoup/jsoup/1.8.2
- ▶  api-java-impl-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/api-java-impl/0.5
- ▶  model-0.5.jar - /Users/macher1/.m2/repository/org/opencompare/model/0.5
- ▶  org.kevoree.modeling.microframework-3.5.12.jar - /Users/macher1/.m2/repository/org/kevoree/modeling
- ▶  kotlin-stdlib-0.8.11.jar - /Users/macher1/.m2/repository/org/jetbrains/kotlin/kotlin-stdlib/0.8.11
- ▶  kotlin-runtime-0.8.11.jar - /Users/macher1/.m2/repository/org/jetbrains/kotlin/kotlin-runtime/0.8.11
- ▶  junit-4.11.jar - /Users/macher1/.m2/repository/junit/junit/4.11
- ▶  hamcrest-core-1.3.jar - /Users/macher1/.m2/repository/org/hamcrest/hamcrest-core/1.3

```
11<img alt="minus icon" data-bbox="260 525 275 540"/> <dependencies>
12
13 <img alt="minus icon" data-bbox="260 565 275 580"/> <!-- Dependencies to Java PCM API and its implementation -->
14<img alt="minus icon" data-bbox="260 585 275 600"/> <dependency>
15     <groupId>org.opencompare</groupId>
16     <artifactId>api-java</artifactId>
17     <version>0.5</version>
18 </dependency>
19
20<img alt="minus icon" data-bbox="260 705 275 720"/> <dependency>
21     <groupId>org.opencompare</groupId>
22     <artifactId>api-java-impl</artifactId>
23     <version>0.5</version>
24 </dependency>
25
26 <!-- Dependency to Junit for testing our project -->
27<img alt="minus icon" data-bbox="260 845 275 860"/> <dependency>
28     <groupId>junit</groupId>
29     <artifactId>junit</artifactId>
30     <version>4.11</version>
31     <scope>test</scope>
32 </dependency>
33
34 </dependencies>
```

Dependency Hierarchy [test]

Filter:

Dependency Hierarchy

- api-java : 0.5 [compile]
 - opencsv : 3.3 [compile]
 - commons-lang3 : 3.3.2 [compile]
 - jsoup : 1.8.2 [compile]
- api-java-impl : 0.5 [compile]
 - model : 0.5 [compile]
 - org.kevoree.modeling.microframework : 3.5.12 (omitted for
 - api-java : 0.5 (omitted for conflict with 0.5) [compile]
 - org.kevoree.modeling.microframework : 3.5.12 [compile]
 - kotlin-stdlib : 0.8.11 [compile]
 - kotlin-runtime : 0.8.11 [compile]
- junit : 4.11 [test]
 - hamcrest-core : 1.3 [test]

Resolved Dependencies

- api-java : 0.5 [compile]
- api-java-impl : 0.5 [compile]
- commons-lang3 : 3.3.2 [compile]
- hamcrest-core : 1.3 [test]
- jsoup : 1.8.2 [compile]
- junit : 4.11 [test]
- kotlin-runtime : 0.8.11 [compile]
- kotlin-stdlib : 0.8.11 [compile]
- model : 0.5 [compile]
- opencsv : 3.3 [compile]
- org.kevoree.modeling.microframework : 3.5.12 [compile]

Overview

Dependencies

Dependency Hierarchy

Effective POM

pom.xml

This XML file does not appear to have any style information associated with it. The document tree is

```
▼ <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  ▼ <parent>
    <artifactId>opencompare</artifactId>
    <groupId>org.opencompare</groupId>
    <version>0.5</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>api-java</artifactId>
  ▼ <dependencies>
    ▼ <dependency>
      <groupId>com.opencsv</groupId>
      <artifactId>opencsv</artifactId>
      <version>3.3</version>
    </dependency>
    ▼ <dependency>
      <groupId>org.scala-lang</groupId>
      <artifactId>scala-library</artifactId>
      <version>${scala.version}</version>
      <scope>test</scope>
    </dependency>
    ▼ <dependency>
      <groupId>org.scalatest</groupId>
      <artifactId>scalatest_${scala.version.minor}</artifactId>
      <version>${scalatest.version}</version>
      <scope>test</scope>
    </dependency>
    ▼ <dependency>
      <!-- jsoup HTML parser library @ http://jsoup.org/ -->
      <groupId>org.jsoup</groupId>
      <artifactId>jsoup</artifactId>
      <version>1.8.2</version>
    </dependency>
  </dependencies>
</project>
```

http://search.maven.org/

 The Central Repository

[SEARCH](#) | [ADVANCED SEARCH](#) | [B](#)

The Search Engine for The Central Repository

SEARCH

[New: About Central](#)

[Advanced Search](#) | [API Guide](#) | [Help](#)

SEARCH

[New: About Central](#)

[Advanced Search](#) | [API Guide](#) | [Help](#)

Search Results

< 1 > displaying 1 t

| GroupId | ArtifactId | Latest Version | Updated | Download |
|---|------------------------------------|---|-------------|---|
| org.opencompare | opencompare | 0.5 all (2) | 21-Aug-2015 | pom |
| org.opencompare.io | io | 0.5 all (2) | 21-Aug-2015 | pom |
| org.opencompare.dataset | dataset | 0.5 all (2) | 21-Aug-2015 | pom |
| org.opencompare | api-js | 0.5 all (2) | 21-Aug-2015 | pom jar |
| org.opencompare | api-js-impl | 0.5 all (2) | 21-Aug-2015 | pom jar |
| org.opencompare.dataset | dataset-wikipedia | 0.5 all (2) | 21-Aug-2015 | pom jar |
| org.opencompare | formalizer | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare | model | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare.dataset | dataset-best-buy | 0.5 all (2) | 21-Aug-2015 | pom jar |
| org.opencompare | hac | 1.0.0 | 21-Jul-2015 | pom jar javadoc.jar sources.jar |
| org.webjars.bower | opencompare-editor | 0.1.1 | 14-Sep-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare.io | io-wikipedia | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare.io | io-best-buy | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare | play-app | 0.5 all (2) | 21-Aug-2015 | pom jar zip war javadoc.jar sources.jar |
| org.opencompare | api-java-impl | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar |
| org.opencompare | api-java | 0.5 all (2) | 21-Aug-2015 | pom jar javadoc.jar sources.jar tests.jar |

pom.xml

```
36<build>
37  <pluginManagement>
38    <plugins>
39      <plugin>
40        <groupId>org.apache.maven.plugins</groupId>
41        <artifactId>maven-compiler-plugin</artifactId>
42        <version>2.1</version>
43        <configuration>
44          <source>1.8</source>
45          <target>1.8</target>
46        </configuration>
47      </plugin>
48    </plugins>
49  </pluginManagement>
```

```
macher-wifi:getting-started macher1$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building getting-started 1.0-SNAPSHOT
[INFO] -----
Downloading: https://repo.maven.apache.org/maven2/org/opencompare/api-java/0.5/api-java-0.5.pom
Downloaded: https://repo.maven.apache.org/maven2/org/opencompare/api-java/0.5/api-java-0.5.pom (4 KB at 3.6 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/opencompare/api-java-impl/0.5/api-java-impl-0.5.pom
Downloaded: https://repo.maven.apache.org/maven2/org/opencompare/api-java-impl/0.5/api-java-impl-0.5.pom (4 KB at 29.1 KB/sec)
Downloading: https://repo.maven.apache.org/maven2/org/opencompare/api-java/0.5/api-java-0.5.jar
Downloading: https://repo.maven.apache.org/maven2/org/opencompare/api-java-impl/0.5/api-java-impl-0.5.jar
Downloaded: https://repo.maven.apache.org/maven2/org/opencompare/api-java/0.5/api-java-0.5.jar (51 KB at 254.1 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/opencompare/api-java-impl/0.5/api-java-impl-0.5.jar (38 KB at 130.5 KB/sec)
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ getting-started ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/macher1/git/getting-started/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.1:compile (default-compile) @ getting-started ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.183 s
[INFO] Finished at: 2015-09-30T14:56:35+02:00
[INFO] Final Memory: 10M/167M
[INFO] -----
```

```
36< build>
37<   <pluginManagement>
38<     <plugins>
39<       <plugin>
40<         <groupId>org.apache.maven.plugins</groupId>
41<         <artifactId>maven-compiler-plugin</artifactId>
42<         <version>2.1</version>
43<         <configuration>
44<           <source>1.8</source>
45<           <target>1.8</target>
46<         </configuration>
47<       </plugin>
48<     </plugins>
49<   </pluginManagement>
```

```
.....
macher-wifi:getting-started macher1$ mvn
[INFO] Scanning for projects...
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.099 s
[INFO] Finished at: 2015-09-30T16:41:50+02:00
[INFO] Final Memory: 5M/123M
[INFO] -----
[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoGoalSpecifiedException
macher-wifi:getting-started macher1$
```

<http://maven.apache.org/plugins/maven-compiler-plugin/usage.html>

```
macher-wifi:getting-started macher1$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building getting-started 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ getting-started ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/macher1/git/getting-started/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.1:compile (default-compile) @ getting-started ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ getting-started ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/macher1/git/getting-started/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.1:testCompile (default-testCompile) @ getting-started ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ getting-started ---
[INFO] Surefire report directory: /Users/macher1/git/getting-started/target/surefire-reports
```

```
-----
T E S T S
-----
```

```
Running org.opencompare.MyPCMPrinterTest
```

```
--- Products ---
```

```
n1
```


- > pom.xml
- README.md
- org.xtext.example.questionnaire
- org.xtext.example.questionnaire.sdk
- org.xtext.example.questionnaire.tests
- org.xtext.example.questionnaire.ui
- org.xtext.example.videogenerator
- org.xtext.example.videogenerator.sdk
- org.xtext.example.videogenerator.tests
- org.xtext.example.videogenerator.ui
- videogen

- New
- Open F3
- Open With
- Show In ⌘ ⌘ W
- Copy ⌘ C
- Copy Qualified Name
- Paste ⌘ V
- Delete ⌘ X
- Build Path
- Refactor ⌘ ⌘ T
- Import...
- Export...
- Refresh F5
- Assign Working Sets...
- Validate
- Run As**
- Debug As
- Replace With
- Maven
- Team
- Compare With
- Source
- Properties ⌘ I

- m2 1 Maven build ⌘ ⌘ X M
- m2 2 Maven build...
- m2 3 Maven clean
- m2 4 Maven generate-sources
- m2 5 Maven install
- m2 6 Maven test
- Run Configurations...

Overview

Problem

<terminal

No

No

Yes

Yes

Yes

Yes

Yes

No

Yes

Yes



Make



***ma*ven**

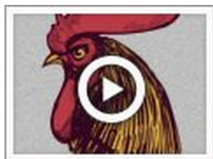


GRUNT

The JavaScript Task Runner

Latest Version

- Stable: [v0.4.5](#) (npm)
- Development: [v0.4.6](#) (github)



Free screencasts about JavaScript, Flexbox, Node.js and more from the experts at Bocoup.

Ads by [Bocoup](#).

Why use a task runner?

In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it through a [Gruntfile](#), a task runner can do most of that mundane work for you—and your team—with basically zero effort.

Why use Grunt?

The Grunt ecosystem is huge and it's growing every day. With literally hundreds of plugins to choose from, you can use Grunt to automate just about anything with a minimum of effort. If someone hasn't already built what you need, authoring and publishing your own Grunt plugin to npm is a breeze. See how to [get started](#).

<http://gruntjs.com/sample-gruntfile>

[Docs](#)

[Search pack](#)




Bower

A package manager for
the web

Chess project: also use Maven!

(either using an existing pom.xml
or using an archetype)

```
macher-wifi:miagedemo macher1$ mvn archetype:generate
```

 fresnault on 7 Jul Add parameters depth and multipv

1 contributor

44 lines (42 sloc) | 1.16 KB

[Raw](#) [Blame](#) [History](#)

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>chess-analysis</groupId>
5   <artifactId>chess-analysis</artifactId>
6   <version>0.0.1</version>
7
8   <build>
9     <plugins>
10       <plugin>
11         <groupId>org.apache.maven.plugins</groupId>
12         <artifactId>maven-jar-plugin</artifactId>
13         <version>2.4</version>
14         <configuration>
15           <archive>
16             <manifest>
17               <addClasspath>true</addClasspath>
18               <mainClass>main.java.Main</mainClass>
19             </manifest>
20           </archive>
21         </configuration>
22       </plugin>
23     </plugins>
24   </build>
25
26   <dependencies>
27     <dependency>
28       <groupId>mysql</groupId>
29       <artifactId>mysql-connector-java</artifactId>
30       <version>5.1.35</version>
31       <scope>compile</scope>
32     </dependency>
33     <dependency>
34       <groupId>jline</groupId>
35       <artifactId>jline</artifactId>
36       <version>2.12.1</version>
37     </dependency>
38     <dependency>
39       <groupId>com.beust</groupId>
40       <artifactId>jcommander</artifactId>
```

Today

- Build System (maven)
- **Manage your source code**
- Refactoring
- Logging, Debugging and Test
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

Manage your source code

Documentation

- Source code: one of the best artefact for documenting a project
- Javadoc (JDK)
 - Automatic **generation** of HTML documentation
 - Using comments in java files
- Syntax

```
/**  
 * This is a <b>doc</b> comment.  
 * @see java.lang.Object  
 * @todo fix {@uunderline this !}  
 */
```
- Includes
 - class hierarchy, interfaces, packages
 - detailed summary of class, interface, methods, attributes
- Note
 - Add doc generation to your favorite **compile chain**



Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

See:

[Description](#)

Interface Summary

| | |
|---|---|
| Action | The <code>Action</code> interface provides a useful extension to the <code>ActionListener</code> interface in cases where the same functionality may be accessed by several controls. |
| BoundedRangeModel | Defines the data model used by components like Sliders and ProgressBars. |
| ButtonModel | State Model for buttons. |
| CellEditor | This interface defines the methods any general editor should be able to implement. |
| ComboBoxEditor | The editor component used for JComboBox components. |
| ComboBoxModel | A data model for a combo box. |
| DesktopManager | DesktopManager objects are owned by a JDesktopPane object. |
| Icon | A small fixed size picture, typically used to decorate components. |
| JComboBox.KeySelectionManager | The interface that defines a <code>KeySelectionManager</code> . |
| ListCellRenderer | Identifies components that can be used as "rubber stamps" to paint the cells in a JList. |
| ListModel | This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list. |
| ListSelectionModel | This interface represents the current state of the selection for any of the components that display a list of values with stable indices. |
| MenuItem | Any component that can be placed into a menu should implement this interface. |
| MutableComboBoxModel | A mutable version of <code>ComboBoxModel</code> . |
| Renderer | Defines the requirements for an object responsible for "rendering" (displaying) a value. |
| RootPaneContainer | This interface is implemented by components that have a single <code>JRootPane</code> child: <code>JDialog</code> , <code>JFrame</code> , <code>JWindow</code> , <code>JApplet</code> , <code>JInternalFrame</code> . |
| Scrollable | An interface that provides information to a scrolling container like <code>JScrollPane</code> . |
| ScrollPaneConstants | Constants used with the <code>JScrollPane</code> component. |
| SingleSelectionModel | A model that supports at most one indexed selection. |
| SpinnerModel | A model for a potentially unbounded sequence of object values. |
| SwingConstants | A collection of constants generally used for positioning and orienting components on the screen. |
| UIDefaults.ActiveValue | This class enables one to store an entry in the defaults table that's constructed each time it's looked up with one of the <code>getXXX(key)</code> methods. |
| UIDefaults.LazyValue | This class enables one to store an entry in the defaults table that isn't constructed until the first time it's looked up with one of the <code>getXXX(key)</code> methods. |
| WindowConstants | Constants used to control the window closing operation. |

public class **JFrame**
extends [Frame](#)
implements [WindowConstants](#), [Accessible](#), [RootPaneContainer](#)

An extended version of `java.awt.Frame` that adds support for the JFC/Swing component architecture. You can find task-o

The `JFrame` class is slightly incompatible with `Frame`. Like all other JFC/Swing top-level containers, a `JFrame` contains a `JFrame` content pane, unlike the AWT `Frame` case. For example, to add a child to an AWT frame you'd write:

```
frame.add(child);
```

However using `JFrame` you need to add the child to the `JFrame`'s content pane instead:

```
frame.getContentPane().add(child);
```

The same is true for setting layout managers, removing components, listing children, and so on. All these methods should not throw an exception. The default content pane will have a `BorderLayout` manager set on it.

update

```
public void update(Graphics g)
```

Just calls `paint(g)`. This method was overridden to prevent an unnecessary call to clear the background.

Overrides:

[update](#) in class [Container](#)

Parameters:

`g` - the Graphics context in which to `paint`

See Also:

[Component.update\(Graphics\)](#)



Kornel Kisielewicz @epyoncf

12 Aug

ProTip: "//" is the speedup operator. Use // before the statement you want to speed up. Works in C++, Java and a few others!

 Retweeted by Mathieu Acher

[Collapse](#)

[← Reply](#) [↻ Retweeted](#) [★ Favorite](#) [⋮ More](#)

1,253

RETWEETS

295

FAVORITES



12:31 AM - 12 Aug 13 · [Details](#)

Coding Conventions

- Rules on the coding style :
 - Apache, Oracle and others template
 - e.g.
 - <http://www.oracle.com/technetwork/java/codeconv-138413.html>
 - <http://geosoft.no/development/javastyle.html>
- Verification tools
 - CheckStyle, PMD, JackPot, Spoon Vsuite...
 - Some integrated into IDEs

Why Coding Standards are Important?

- Lead to greater **consistency** within your code and the code of your teammates
- Easier to **understand**
- Easier to **develop**
- Easier to **maintain**
- Reduces overall cost of application

Example

8. Private class variables should have underscore suffix.

```
class Person
{
    private String name_;
    ...
}
```

Apart from its name and its type, the *scope* of a variable is its most higher significance than method variables, and should be treated w

A side effect of the underscore naming convention is that it nicely r

```
void setName(String name)
{
    name_ = name;
}
```

Code Style and Code Conventions: many languages (Java, XML, JavaScript, HTML, CSS, etc.)

An example:

<https://maven.apache.org/developers/conventions/code.html>

The following is the recommended ordering for all Maven POM files:

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
   e.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2.   <modelVersion/>
3.
4.   <parent/>
5.
6.   <groupId/>
7.   <artifactId/>
8.   <version/>
9.   <packaging/>
10.
11.  <name/>
12.  <description/>
13.  <url/>
14.  <inceptionYear/>
15.  <organization/>
16.  <licenses/>
17.
18.  <developers/>
19.  <contributors/>
20.
21.  <mailingLists/>
22.
23.  <prerequisites/>
24.
25.  <modules/>
26.
27.  <scm/>
28.  <issueManagement/>
29.  <ciManagement/>
30.  <distributionManagement/>
31.
32.  <properties/>
33.
34.  <dependencyManagement/>
35.  <dependencies/>
36.
37.  <repositories/>
38.  <pluginRepositories/>
39.
```

<https://maven.apache.org/developers/conventions/code.html>

<https://maven.apache.org/developers/conventions/code.html>

Generic Code Style And Convention

All working files (java, xml, others) should respect the following conventions:

- **License Header:** Always add the current [ASF license header](#) in all versioned files.
- **Trailing Whitespaces:** Remove all trailing whitespaces. If you are an Eclipse user, you could use the [Anyedit Eclipse Plugin](#).

and the following style:

- **Indentation: Never** use tabs!
- **Line wrapping:** Always use a 120-column line width.

Note: The specific styles and conventions, listed in the next sections, could override these generic rules.

Java

Java Code Style

The Maven style for Java is mainly:

- **White space:** One space after control statements and between arguments (i.e. `if (foo)` instead of `if(foo)`), `myFunc(foo, bar, baz)` instead of `myFunc(foo,bar,baz)`). No spaces after methods names (i.e. `void myMethod()`, `myMethod("foo")`)
- **Indentation:** Always use 4 space indents and **never** use tabs!
- **Blocks:** Always enclose with a new line brace.
- **Line wrapping:** Always use a 120-column line width for Java code and Javadoc.

Tools to Improve your Source code

- Formatting tools
 - Indenteurs (Jindent), beautifiers, stylers (JavaStyle), ...
- Code conventions/styles:
 - Eg Checkstyle
 - Exists as a Maven plugin (<https://maven.apache.org/plugins/maven-checkstyle-plugin/>)
- Quality report tools: code metrics
 - Number of Non Comment Code Source, Number of packages, Cyclomatic numbers, ...
 - JavaNCCS, Eclipse Metrics ...

Today

- Build System (maven)
- Manage your source code
- **Refactoring**
- Logging, Debugging and Test
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

Refactoring

What's Code Refactoring?

“A series of *small* steps, each of which changes the program's *internal structure* without changing its *external behavior*”



Martin Fowler

Example

Which code segment is easier to read?

Sample 1:

```
if (markT>=0 && markT<=25 && markL>=0 && markL<=25) {  
    float markAvg = (markT + markL) / 2;  
    System.out.println("Your mark: " + markAvg);  
}
```

Sample 2:

```
if (isValid(markT) && isValid(markL)) {  
    float markAvg = (markT + markL) / 2;  
    System.out.println("Your mark: " + mark);  
}
```

Why do we Refactor?

- Improves the design of our software
 - Design pattern!
- Minimizes technical debt
- Keep development at speed
- To make the software easier to understand
- To help find bugs
- To “Fix broken windows”

Non exhaustive (code smell)

(and not necessarily smells in all situations)

- Duplicated code
- Feature Envy
- Inappropriate Intimacy
- Comments
- Long Method
- Long Parameter List
- Switch Statements
- Improper Naming

Code Smell examples (1)

```
public void display(String[] names) {
    System.out.println("-----");
    for(int i=0; i<names.length; i++){
        System.out.println(" + " + names[i]);
    }
    System.out.println("-----");
}
```

```
public void listMember(String[] names) {
    System.out.println("List all member: ");
    System.out.println("-----");
    for(int i=0; i<names.length; i++){
        System.out.println(" + " + names[i]);
    }
    System.out.println("-----");
}
```

Duplicated code

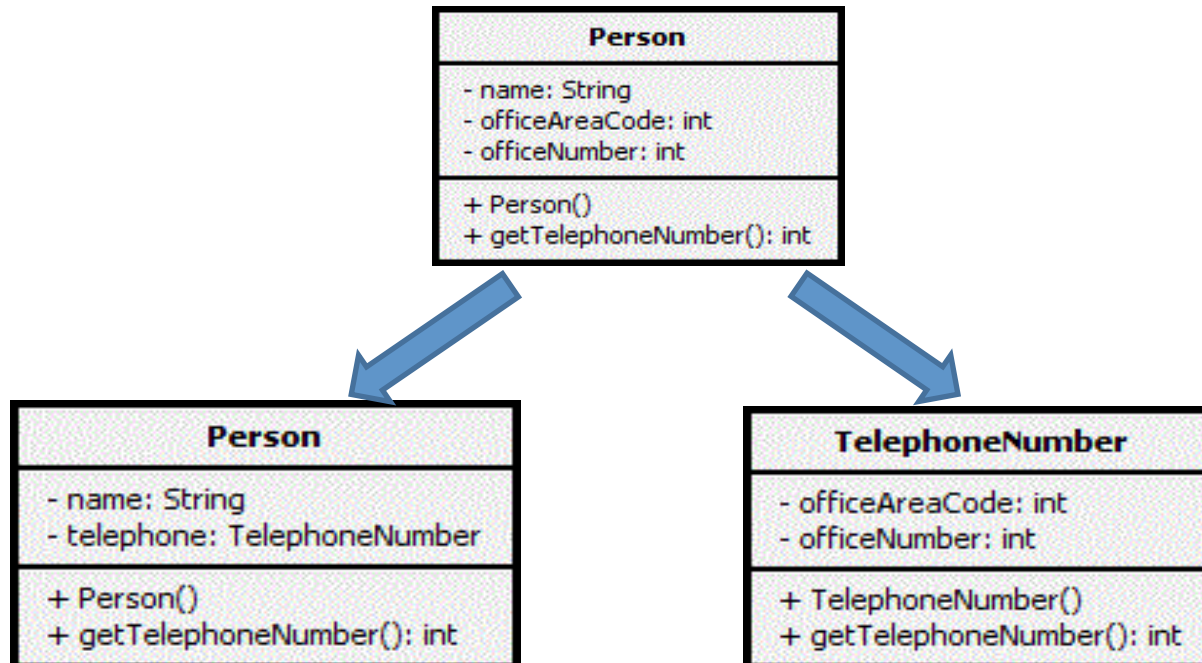
Code Smell examples (2)

```
public String formatStudent( int id,  
                             String name,  
                             Date dob,  
                             String province,  
                             String address,  
                             String phone ){  
  
    //TODO:  
    return null;  
}
```

Long list of parameters

Improving design

- Move Method or Move Field – move to a more appropriate Class or source file
- Rename Method or Rename Field – changing the name into a new one that better reveals its purpose
 - Pull Up – in OOP, move to a superclass
 - Push Down – in OOP, move to a subclass



How do we Refactor?

- Manual Refactoring
 - Code Smells
- Automated/Assisted Refactoring
 - Refactoring by hand is time consuming and prone to error
 - Tools (IDE)
- In either case, **test your changes**

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

Problems Javadoc Declaration Console Error Log

<terminated> MyFirstClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe

Hello Eclipse!
5050

Extract Method

Method name:

Access modifier: public protected default private

Parameters:

| Type | Name |
|------|------|
| int | sum |
| | |
| | |
| | |
| | |

Declare thrown runtime exceptions
 Generate method comment
 Replace additional occurrences of statements with method

Method signature preview:
private static int calculateSum(int sum)

Preview > OK Cancel

```
package de.vogella.eclipse.ide.first;

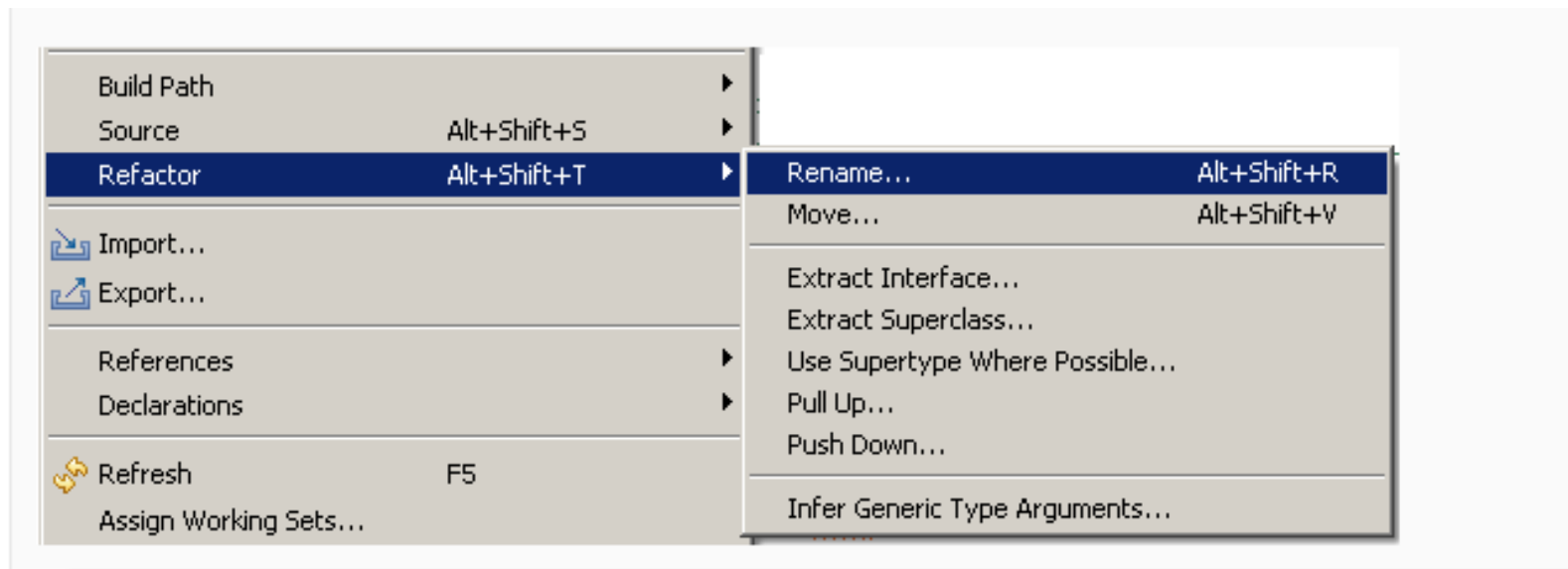
public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        sum = calculateSum(sum);
        System.out.println(sum);
    }

    private static int calculateSum(int sum) {
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}
```

Typical refactoring patterns

- Rename variable / class / method / member
- Extract method
- Extract constant
- Extract interface
- Encapsulate field



You have constructors on subclasses with mostly identical bodies.

Create a superclass constructor; call this from the subclass methods.

Pull Up Constructor Body

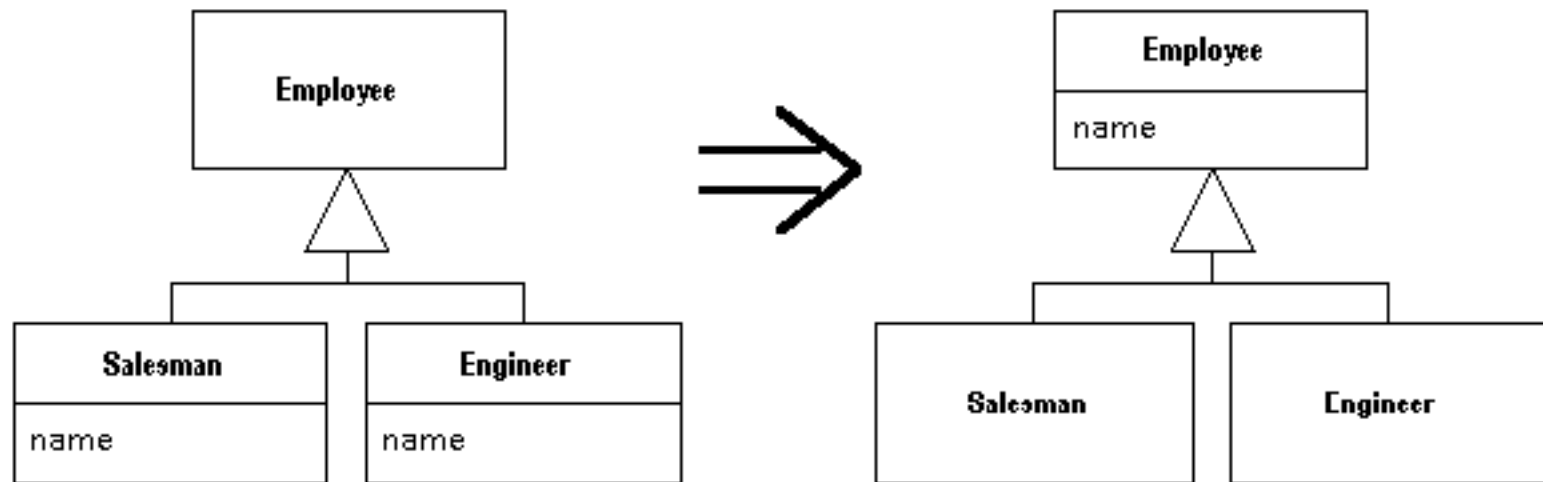
```
class Manager extends Employee...  
    public Manager (String name, String id, int grade) {  
        _name = name;  
        _id = id;  
        _grade = grade;  
    }
```

```
    public Manager (String name, String id, int grade) {  
        super (name, id);  
        _grade = grade;  
    }
```

You
Create

Two subclasses have the same field.

Move the field to the superclass.



You have a complicated expression.

Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
      (browser.toUpperCase().indexOf("IE") > -1) &&
      wasInitialized() && resize > 0 )
{
    // do something
}

final boolean isMacOs      = platform.toUpperCase().indexOf("MAC") > -1;
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE") > -1;
final boolean wasResized  = resize > 0;

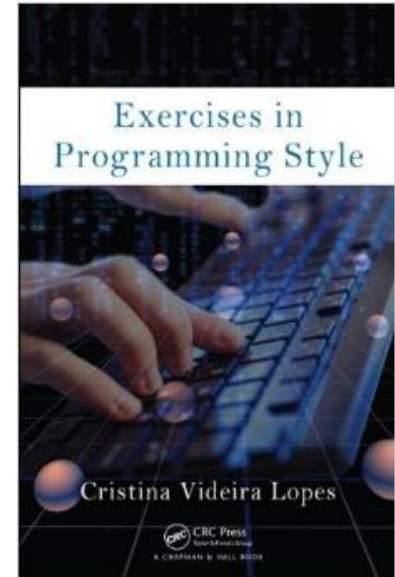
if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
{
    // do something
}
```


Today

- Build System (maven)
- Manage your source code
- Refactoring
- **Logging, Debugging and Test**
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

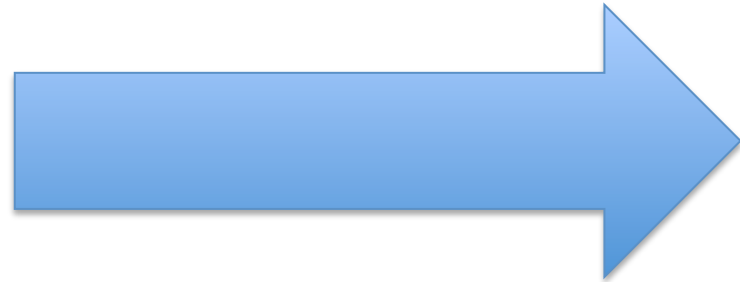
Logging,
Debugging,
Testing

Given a text file, output a list of the N most frequently-occurring non stop, words, ordered by decreasing frequency



N: integer

I2: File



O1: string

N=25



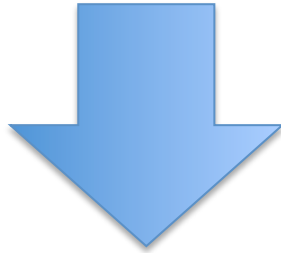
crista on 22 Sep 2013 Added input files

1 contributor

I2=

3 lines (2 sloc) | 0.067 kb

```
1 | White tigers live mostly in India
2 | Wild lions live mostly in Africa
```



O1=

```
« live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 1
white - 1
wild - 1 »
```

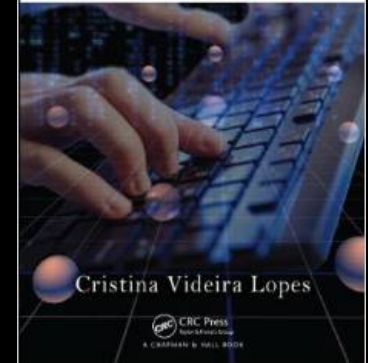
« Given a text file, output a list of the N most frequently-occurring non stop words, ordered by decreasing frequency »

```

1 import sys, string
2 # the global list of [word, frequency] pairs
3 word_freqs = []
4 # the list of stop words
5 with open('../stop_words.txt') as f:
6     stop_words = f.read().split(',')
7     stop_words.extend(list(string.ascii_lowercase))
8
9 # iterate through the file one line at a time
10 for line in open(sys.argv[1]):
11     start_char = None
12     i = 0
13     for c in line:
14         if start_char == None:
15             if c.isalnum():
16                 # We found the start of a word
17                 start_char = i
18         else:
19             if not c.isalnum():
20                 # We found the end of a word. Process it
21                 found = False
22                 word = line[start_char:i].lower()
23                 # Ignore stop words
24                 if word not in stop_words:
25                     pair_index = 0
26                     # Let's see if it already exists
27                     for pair in word_freqs:
28                         if word == pair[0]:
29                             pair[1] += 1
30                             found = True
31                             found_at = pair_index
32                             break
33                     pair_index += 1
34                 if not found:
35                     word_freqs.append([word, 1])
36                 elif len(word_freqs) > 1:
37                     # We may need to reorder
38                     for n in reversed(range(pair_index)):
39                         if word_freqs[pair_index][1] >
40                             word_freqs[n][1]:
41                             # swap
42                             word_freqs[n], word_freqs[
43                                 pair_index] = word_freqs[
44                                     pair_index], word_freqs[n]
45                             pair_index = n
46                 # Let's reset
47                 start_char = None
48             i += 1
49
50 for tf in word_freqs[0:25]:
51     print tf[0], ' - ', tf[1]

```

Exercises in Programming Style



```
# the global list of [word, frequency] pairs
word_freqs = []
# the list of stop words
with open('../stop_words.txt') as f:
    stop_words = f.read().split(',')
stop_words.extend(list(string.ascii_lowercase))
```

```
13     for c in line:
14         if start_char == None:
15             if c.isalnum():
16                 # We found the start of a word
17                 start_char = i
18         else:
19             if not c.isalnum():
20                 # We found the end of a word. Process it
21                 found = False
22                 word = line[start_char:i].lower()
23                 # Ignore stop words
24                 if word not in stop_words:
25                     pair_index = 0
26                     # Let's see if it already exists
27                     for pair in word_freqs:
28                         if word == pair[0]:
29                             pair[1] += 1
30                             found = True
31                             found_at = pair_index
32                             break
33                     pair_index += 1
34                 if not found:
35                     word_freqs.append([word, 1])
36                 elif len(word_freqs) > 1:
37                     # We may need to reorder
38                     for n in reversed(range(pair_index)):
39                         if word_freqs[pair_index][1] >
40                             word_freqs[n][1]:
41                             # swap
42                             word_freqs[n], word_freqs[
43                                 pair_index] = word_freqs[
44                                 pair_index], word_freqs[n]
45                             pair_index = n
46                 # Let's reset
47                 start_char = None
48                 i += 1
49
50 for tf in word_freqs[0:25]:
51     print tf[0], ' - ', tf[1]
```

```
1 import sys, string
2 # the global list of [word, frequency] pairs
3 word_freqs = []
4 # the list of stop words
5 with open('../stop_words.txt') as f:
6     stop_words = f.read().split(',')
```

```
for line in open(sys.argv[1]):
```

```
    for c in line:
```

```
19         if not c.isalnum():
20             # We found the end of a word. Process it
21             found = False
22             word = line[start_char:i].lower()
23             # Ignore stop words
24             if word not in stop_words:
25                 pair_index = 0
26                 # Let's see if it already exists
27                 for pair in word_freqs:
28                     if word == pair[0]:
29                         pair[1] += 1
30                         found = True
31                         found_at = pair_index
32                         break
33                 pair_index += 1
34             if not found:
35                 word_freqs.append([word, 1])
36             elif len(word_freqs) > 1:
37                 # We may need to reorder
38                 for n in reversed(range(pair_index)):
39                     if word_freqs[pair_index][1] >
40                         word_freqs[n][1]:
41                         # swap
42                         word_freqs[n], word_freqs[
43                             pair_index] = word_freqs[
44                             pair_index], word_freqs[n]
45                 pair_index = n
46             # Let's reset
47             start_char = None
48         i += 1
49
50 for tf in word_freqs[0:25]:
51     print tf[0], ' - ', tf[1]
```

N=25



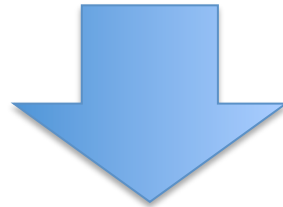
crista on 22 Sep 2013 Added input files

1 contributor

I2=

3 lines (2 sloc) | 0.067 kb

```
1 | White tigers live mostly in India
2 | Wild lions live mostly in Africa
```



O1=

« live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 3
white - 1
wild - 1 »

« Given a text file,
output a list of the N
most frequently-
occurring non stop,
words, ordered by
decreasing frequency »

Let say...

N=25



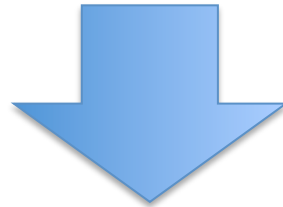
crista on 22 Sep 2013 Added input files

1 contributor

I2=

3 lines (2 sloc) | 0.067 kb

```
1 | White tigers live mostly in India
2 | Wild lions live mostly in Africa
```



O1=

« live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 3
white - 1
wild - 1 »

« Given a text file,
output a list of the N
most frequently-
occurring non stop,
words, ordered by
decreasing frequency »

System.out.println (Logging)?
Debugging?
Profiling?
Testing?

Debugging

Debug Package Explorer

- FML [Java Application]
 - fr.familiar.standalone.FML at localhost:51307
 - Thread [main] (Suspended (breakpoint at line 137 in FML))
 - FML.main(String[]) line: 137
 - Daemon Thread [EMF Reference Cleaner] (Running)
 - Daemon Thread [com.google.inject.internal.util.\$Finalizer] (Running)
 - /Library/Java/JavaVirtualMachines/jdk1.7.0_13.jdk/Contents/Home/bin/java (17 nov. 2014 12:58:37)

Variables Breakpoints

| Name | Value |
|----------|-------------------------|
| args | String[0] (id=19) |
| jsap | JSAP (id=20) |
| sw1 | Switch (id=24) |
| sw2 | Switch (id=28) |
| sw3 | Switch (id=29) |
| qsw1 | QualifiedSwitch (id=30) |
| output1 | FlaggedOption (id=34) |
| opt2 | UnflaggedOption (id=35) |
| config | JSAPResult (id=37) |
| help | false |
| filename | null |

```
129         + e.getMessage();
130     }
131     }
132 }
133
134 FMLShell shell = FMLShell.instantiateStandalone(in);
135
136 boolean verbose = config.getBoolean("verbose");
137 shell.setVerbose(verbose);
138
139 boolean version = config.getBoolean("version");
140 if (version) {
141     System.out.println("version " + FMLShell.FML_VERSION);
142     return;
143 }
144
145 String outputpath = config.getString("output");
146
147 String[] paths = config.getStringArray("paths");
148 for (int i = 0; i < paths.length; ++i) {
149     String path = paths[i];
150     File f = new File(path);
151     if (!f.exists()) {
152         System.err.println("Path " + path + " does not exist");
153         return;
154     }
155 }
```

Outline

- fr.familiar.standalone
 - FML
 - displayUsage(JSAP, Pr
 - main(String[]) : void
 - FML()

HelloWorldScala > src > Reviews.scala

Project

- HelloWorldScala [ScholarServicesScala] (~/Docume)
 - .idea
 - src
 - Reviews
 - ReviewService.scala
 - ScholarServicesScala.iml
 - services.html
 - services.sc
 - External Libraries
 - < 1.6 > (/System/Library/Java/JavaVirtualMach)
 - scala-library

```
val journals = format(reviews.event.filter(e => e.kind == KJournal))
val confs = format(reviews.event.filter(e => (e.kind == KConference) || (e.kind == KWorkshop)))
val file = new File("services.html")
val doct = DocType("html", PublicID("-//W3C//DTD XHTML 1.0 Strict//EN", "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"), Nil)
val doc =
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>M. Acher, PhD, Associate Professor (personal webpage)</title>
<link rel="stylesheet" href="css/myStyle.css" type="text/css" media="screen" />
<link rel="stylesheet" href="css/nivo-slider.css" type="text/css" />
<link rel="stylesheet" href="css/jquery.fancybox-1.3.4.css" type="text/css" />
```

Debugger Console

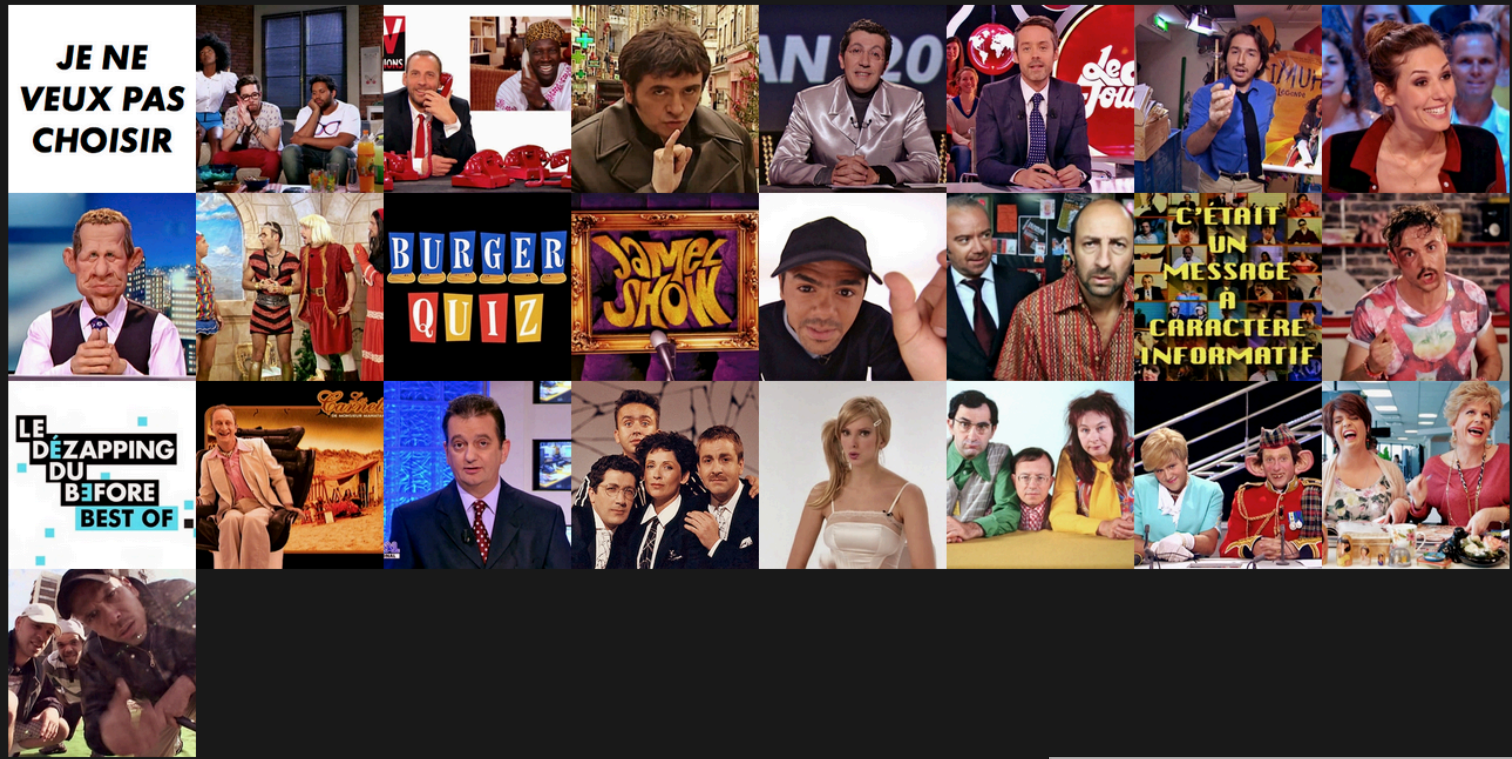
Frames

- main@1 in group "main": RUN...
- main():50, Reviews\$
- main():-1, Reviews

Variables

- this = {Reviews\$@984}
- args = {java.lang.String[0]@989}
- reviews = {ReviewServices@990} "ReviewServices(List(Venue(SAC'15 (SE),2015,30th ACM Symposium on Applied Computing - Software Engineering (SE) Track,KConference,PCMember,...
- event = {scala.collection.immutable.\$colon\$colon@1168} ":" size = 33
- journals = {scala.xml.Elem@991} "Elem" size = 1
 - (0) = {scala.xml.Elem@991} "Elem" size = 1

ETAPE 2 : CHOISIS 3 BONS SOUVENIRS



```
a97d2829.main.js (index) a97d2829.main.js:formatted x
5058   in( "wm.send", event, this._analytics, navigation, playVideo ), this.element[this.subwidg
5059   },pause: function() {
5060     this.element[this.subwidget]("pause")
5061   },resume: function() {
5062     this.element[this.subwidget]("resume")
5063   },getPosition: function() {
5064     return this.element[this.subwidget]("getPosition")
5065   },getDuration: function() {
5066     return this.element[this.subwidget]("getDuration")
5067   },playM3u8: function(a, c, d, e) {
5068     var f = null;
5069     c && (f = this.element[this.subwidget]("getPosition")), b.debug("play keeping position",
5070   },preloadHLS: function(a) {
5071     this._hlsArray = a.slice(0), this.element[this.subwidget]("supportDataUri") && (this._pre
5072   },playHLS: function(a, b, c, d) {
5073     h("wm.send", "event", this._analytics, "navigation", "playVideo"), 1 == d && h("wm.send",
5074     var e = this;
5075     this._preloading ? this._preloading.done(function(a) {
```

Debugger interface showing a list of breakpoints. The first breakpoint is selected and active.

- Watch Expressions +
- Call Stack Async
- Not Paused
- Scope Variables
- Not Paused
- Breakpoints
 - a97d2829.main.js:formatted:5068
var f = null;
 - DOM Breakpoints
 - XHR Breakpoints +
 - Event Listener Breakpoints

```
a97d2829.main.js (index) a97d2829.main.js:formatted x
5058     in XMLHttpRequest.prototype.send, event, this._analytics, navigation, replayvideo, this.element[this.subwidg
5059     },pause: function() {
5060         this.element[this.subwidget]("pause")
5061     },resume: function() {
5062         this.element[this.subwidget]("resume")
5063     },getPosition: function() {
5064         return this.element[this.subwidget]("getPosition")
5065     },getDuration: function() {
5066         return this.element[this.subwidget]("getDuration")
5067     },playM3u8: function(a, c, d, e) {
5068         var f = null;
5069         c && (f = this.element[this.subwidget]("getPosition")), b.debug("play keeping position", f)
5070     },preloadHLS: function(a) {
```

▶ Watch Expressions + ↻

No Watch Expressions

▼ Call Stack Async

- a97d2829.main.j...formatted:5068
i.playM3u8
- a97d2829.main.j...formatted:5078
(anonymous function)
- a97d2829.main.j...formatted:1630
k
- a97d2829.main.j...formatted:1664
l.fireWith
- a97d2829.main.j...formatted:1695
fb.each.e.(anonymous function)
- a97d2829.main.j...formatted:5114

▶ Watch Expressions + ↻

No Watch Expressions

▶ Call Stack Async

▼ Scope Variables

▼ Local

- a: "#EXTM3U#EXT-X-VERSION:..."
- c: 0
- d: "M+présente"
- e: 1
- f: undefined
- ▶ this: i

▶ Closure

▶ Global Window

▼ Breakpoints

- a97d2829.main.js:formatted:5068
var f = null;

Profiling

- ~**Debugging** the, e.g., performance
 - Say the computation of term frequency takes 5 minutes (instead of a few seconds)

N: integer

I2: File



O1: string

- Performance can be **tested** as well!

Logging

Logging, why? (claims)

- Logging is easier than debugging
- Logging is faster than debugging
- Logging can work in environments where debugging is not supported
- Can work in production environments
- Logs can be referenced anytime in future as the data is stored

Logging



- Logging is chronological and systematic record of data processing events in a program
 - e.g. the Windows Event Log
- Logs can be saved to a persistent medium to be studied at a later time
- Use logging in the development phase:
 - Logging can help you **debug** the code
- Use logging in the production environment:
 - Helps you **troubleshoot problems**

Logging Methods, How?

- The evil `System.out.println()`
- Custom Solution to Log to various datastores, eg text files, db, etc...
- Use Standard APIs
 - Don't reinvent the wheel

Log4J



- Popular logging frameworks for Java
- Designed to be reliable, fast and extensible
- Simple to understand and to use API
- Allows the developer to control which log statements are output with arbitrary granularity
- Fully configurable at runtime using external configuration files

Log4J Architecture



- Log4J has three main components: loggers, appenders and layouts
 - Loggers
 - Channels for printing logging information
 - Appenders
 - Output destinations (console, File, Database, Email/SMS Notifications, Log to a socket, and many others...)
 - Layouts
 - Formats that appenders use to write their output
- Priorities

Logger

- Responsible for Logging
- Accessed through java code
- Configured Externally
- Every Logger has a name
- Prioritize messages based on level
 - TRACE, DEBUG, INFO, WARN, ERROR & FATAL
- Usually named following dot convention like java classes do.
 - Eg com.foo.bar.ClassName
- Follows inheritance based on name

Logger API

- Factory methods to get Logger

- `Logger.getLogger(Class c)`
- `Logger.getLogger(String s)`

- Method used to log message

- `trace()`, `debug()`, `info()`, `warn()`, `error()`, `fatal()`
- Details
 - `void debug(java.lang.Object message)`
 - `void debug(java.lang.Object message, java.lang.Throwable t)`
- Generic Log method
 - `void log(Priority priority, java.lang.Object message)`
 - `void log(Priority priority, java.lang.Object message, java.lang.Throwable t)`

Root Logger

- The root logger resides at the top of the logger hierarchy. It is exceptional in two ways:
 1. it always exists,
 2. it cannot be retrieved by name.
- `Logger.getRootLogger()`

Appender

- Appenders put the log messages to their actual destinations.
- No programatic change is require to configure appenders
- Can add multiple appenders to a Logger.
- Each appender has its Layout.
- ConsoleAppender, DailyRollingFileAppender, FileAppender, JDBCAppender, JMSAppender, NTEventLogAppender, RollingFileAppender, SMTPAppender, SocketAppender, SyslogAppender, TelnetAppender

Layout

- Used to customize the format of log output.
- Eg. HTMLLayout, PatternLayout, SimpleLayout, XMLLayout
- Most commonly used is PatternLayout
 - Uses C-like syntax to format.
 - Eg. `"%-5p [%t]: %m%n"`
 - `DEBUG [main]: Message 1 WARN [main]: Message 2`

Log4j Basics

- Who will log the messages?
 - The Loggers
- What decides the priority of a message?
 - Level
- Where will it be logged?
 - Decided by Appender
- In what format will it be logged?
 - Decided by Layout

Log4j in Action

```
// get a logger instance named "com.foo"
Logger logger = Logger.getLogger("com.foo");

// Now set its level. Normally you do not need to set the
// level of a logger programmatically. This is usually done
// in configuration files.
logger.setLevel(Level.INFO);

Logger barlogger = Logger.getLogger("com.foo.Bar");

// This request is enabled, because WARN >= INFO.
logger.warn("Low fuel level.");

// This request is disabled, because DEBUG < INFO.
logger.debug("Starting search for nearest gas station.");

// The logger instance barlogger, named "com.foo.Bar",
// will inherit its level from the logger named
// "com.foo" Thus, the following request is enabled
// because INFO >= INFO.
barlogger.info("Located nearest gas station.");

// This request is disabled, because DEBUG < INFO.
barlogger.debug("Exiting gas station search");
```

Layouts bis (eg colorizing Logs)

- <http://logging.apache.org/log4j/2.x/manual/layouts.html>
- <http://jeanchristophegay.com/de-la-couleur-dans-les-logs/>

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration status="OFF">
  <appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %highlight{%5level} %logger{36} - %msg%n"/>
    </Console>
  </appenders>
  <loggers>
    <root level="trace">
      <appender-ref ref="Console"/>
    </root>
  </loggers>
</configuration>
```

```
19:54:42.838 [main] TRACE com.github.jcgay.example.log.log4j2.Main - a trace message.
19:54:42.841 [main] DEBUG com.github.jcgay.example.log.log4j2.Main - a debug message.
19:54:42.842 [main] INFO com.github.jcgay.example.log.log4j2.Main - an info message.
19:54:42.842 [main] WARN com.github.jcgay.example.log.log4j2.Main - a warn message.
19:54:42.843 [main] ERROR com.github.jcgay.example.log.log4j2.Main - a error message.
19:54:42.845 [main] FATAL com.github.jcgay.example.log.log4j2.Main - a fatal message.
```

Log4j Optimization & Best Practises

- User logger as private static variable
- Only one instance per class
- Name logger after class name
- Don't use too many appenders
- Don't use time-consuming conversion patterns (see javadoc)
- Use `Logger.isDebugEnabled()` if need be
- Prioritize messages with proper levels

Logging in JavaScript / NodeJS

- Not only in Java!
- Alternatives also in other languages...
- <https://github.com/flatiron/winston>

```
var winston = require('winston');  
  
winston.log('info', 'Hello distributed log files!');  
winston.info('Hello again distributed logs');
```

Using Logging Levels

Setting the level for your logging message can be accomplished in one of two ways. You can pass a string representing the logging level to the `log()` method or use the level specified methods defined on every winston Logger.

```
//  
// Any Logger instance  
//  
logger.log('silly', "127.0.0.1 - there's no place like home");  
logger.log('debug', "127.0.0.1 - there's no place like home");  
logger.log('verbose', "127.0.0.1 - there's no place like home");  
logger.log('info', "127.0.0.1 - there's no place like home");  
logger.log('warn', "127.0.0.1 - there's no place like home");  
logger.log('error', "127.0.0.1 - there's no place like home");  
logger.info("127.0.0.1 - there's no place like home");  
logger.warn("127.0.0.1 - there's no place like home");  
logger.error("127.0.0.1 - there's no place like home");
```


Colorization (back)

```
var myCustomLevels = {
  levels: {
    foo: 0,
    bar: 1,
    baz: 2,
    foobar: 3
  },
  colors: {
    foo: 'blue',
    bar: 'green',
    baz: 'yellow',
    foobar: 'red'
  }
};

var customLevelLogger = new (winston.Logger)({ levels: myCustomLevels.levels });
customLevelLogger.foobar('some foobar level-ed message');
```

You can't test everything (so one advice by Martin Fowler)

Whenever you are tempted to type something into a print statement or a debugger expression, **write it as a test instead.**



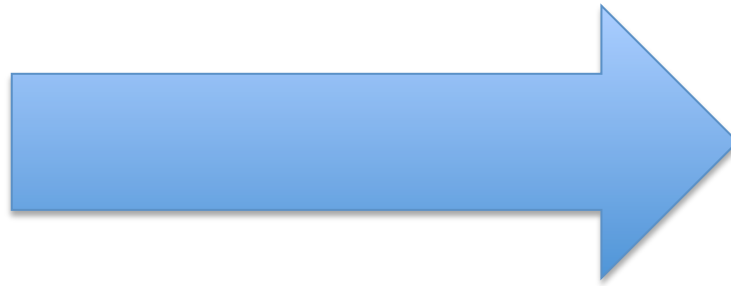
From Logging to Testing

- Testing: “the activity of finding out whether a piece of code produces the intended behavior”
 - Debugging can help
 - Testing is better than debugging

Whenever you are tempted to type something into a print statement or a debugger expression, **write it as a test instead.**



Inputs

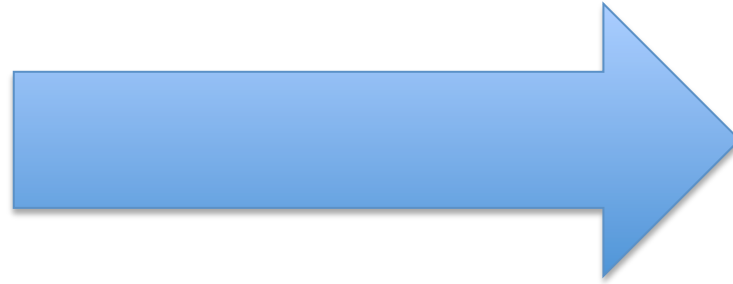


Outputs

**Logging
(manual
inspection of
some values at
specific points)**

N: integer

I2: File

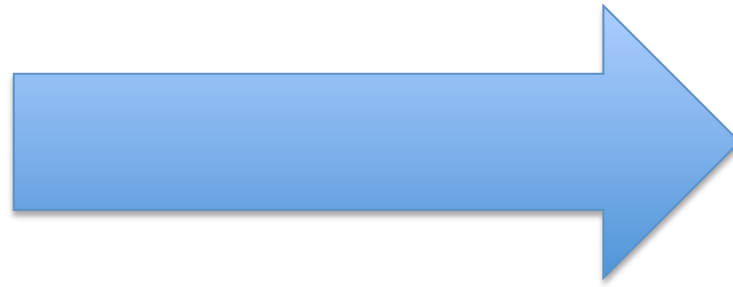


O1: integer

**Debugging
(manual
inspection of
values at some
points)**

N: integer

I2: File



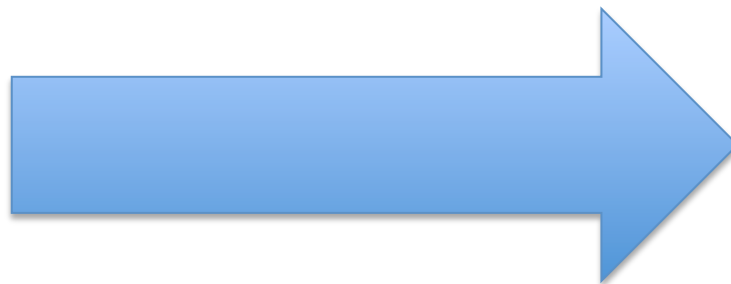
O1: integer

Testing

(automated
verification)

N: integer

I2: File

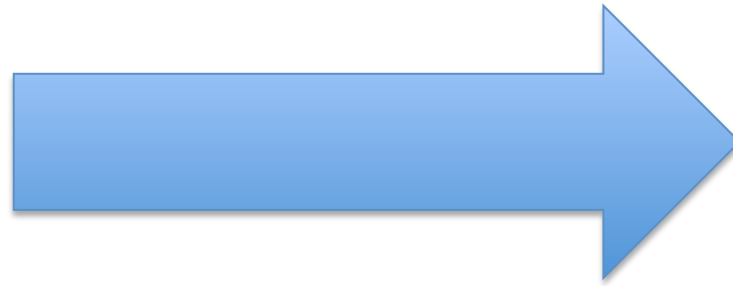


O1: integer

Testing

(automated
verification)

Inputs



Outputs

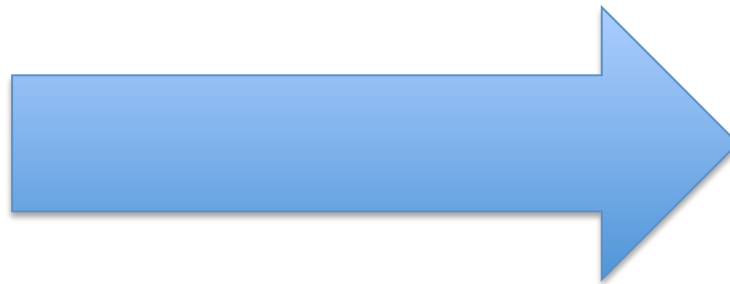
CONTROLLABILITY

ability to manipulate the software's input as well as to place this software into a particular state

OBSERVABILITY

deals with the possibility to observe the outputs and state changes that

Inputs



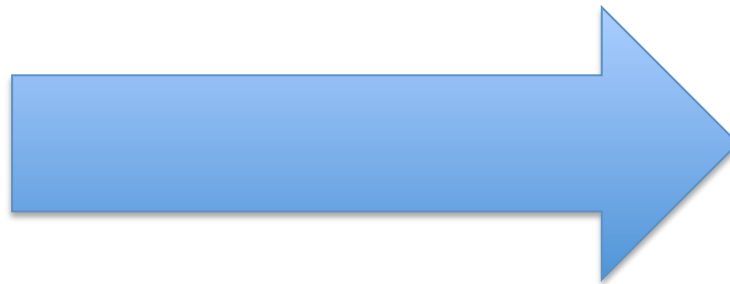
Outputs

TESTABILITY

degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.

Controllability + Observability

Inputs



Outputs

Conclusion

- How to improve Testability?
 - Refactoring, Design patterns
 - Separation of concerns, Modularity, Abstractions
- Logging
- Debugging
- Testing



Whenever you are tempted to type something into a print statement or a debugger expression, **write it as a test instead.**

- **Logging**
 - Manual observation
 - (Usually) manual control on input values
 - Pre-defined exploration of values
- **Debugging**
 - Manual observation
 - (Usually) manual control on input values
 - Interactive, fine-grained exploration of values
- **Testing**
 - Automated control and observation (assertions)
 - More amenable to re-executing on different inputs
 - Not to understand, but to verify some properties

Today

- Build System (maven)
- Manage your source code
- Refactoring
- Logging, Debugging and Test
- IDE (eg, Eclipse)
- Workflow: git, intégration continue, et tous les points ci-dessous

Document, refactor... Execute your tests... Debug.. Write test..

And so on!

Documenting

Refactoring

Debugging

Testing

With modern IDE and tools!

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

Extract Method dialog box:

Method name:

Access modifier: public protected default private

Parameters:

| Type | Name |
|------|------|
| int | sum |

Declare throw name exceptions
 Generate method comment
 Replace additional occurrences of statements with method

Method signature preview:
`private static int calculateSum(int sum)`

Buttons: Preview >, OK, Cancel

Run Test Suite dialog box:

Enter the name of the TestCase class:
.suite()

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and Failures:

Finished: 0.50 seconds

Continuous Integration

« L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque **modification** de code source que le résultat des modifications ne produit pas de **régression** dans l'application développée. »

maven

Gestionnaire
de Build

Gestionnaire
de tests

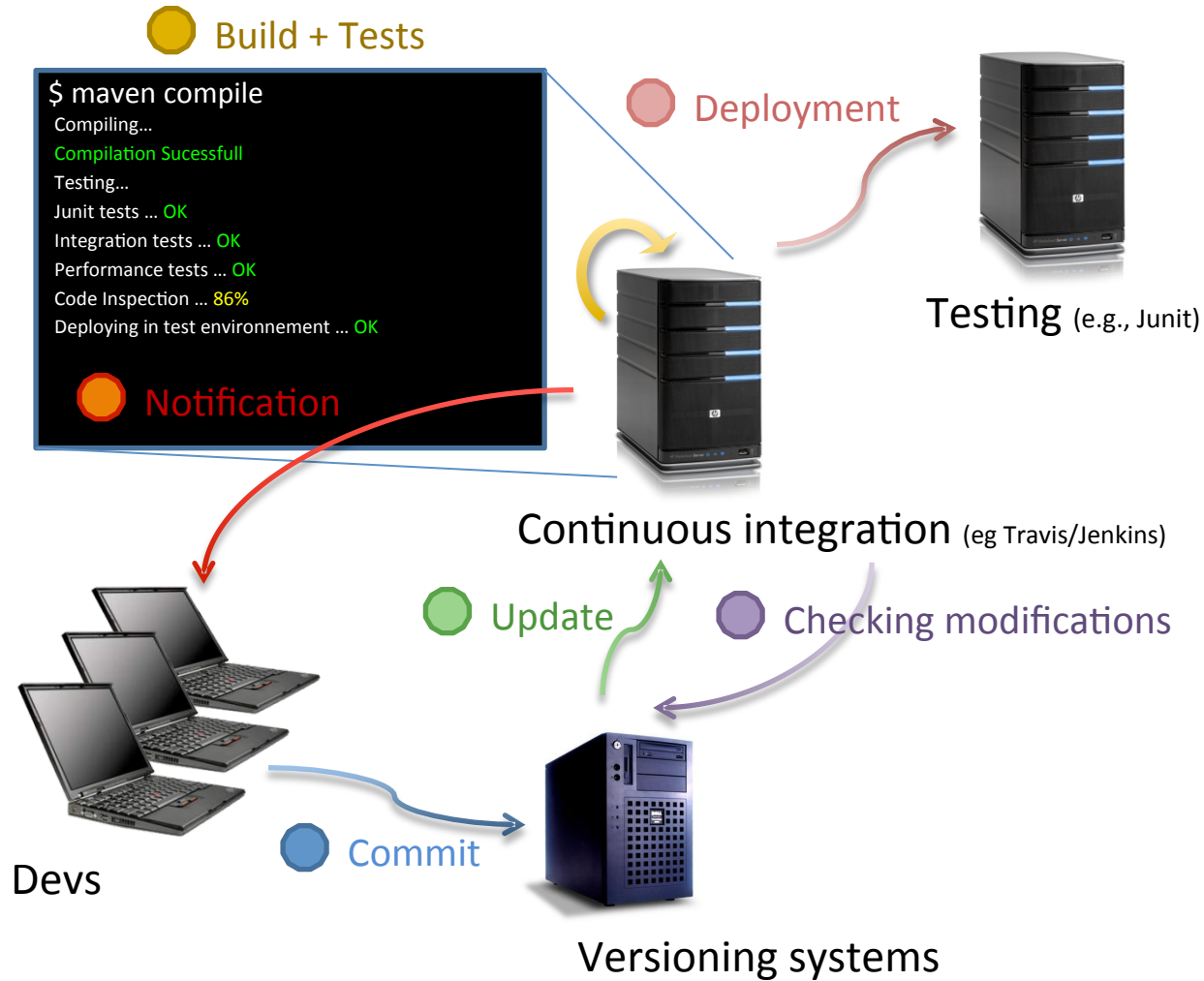
JUnit.org


git

Gestionnaire
de versions

Gestionnaire
de
notifications

Usage





OpenCompare / **getting-started**

Branch: **master** ▾

getting-started / **.travis.yml**



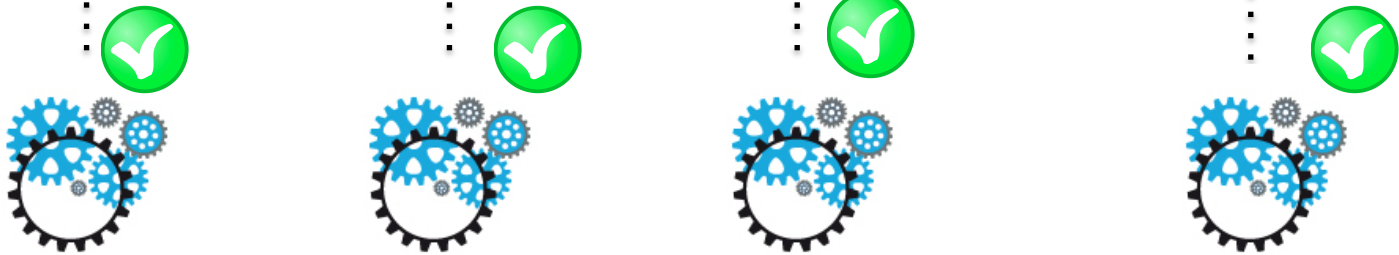
gbecan 10 days ago add .gitignore and .travis.yml file

1 contributor

6 lines (4 sloc) | 62 Bytes

```
1 language: java
2 jdk:
3   - oraclejdk8
4
5 script: mvn clean install
```

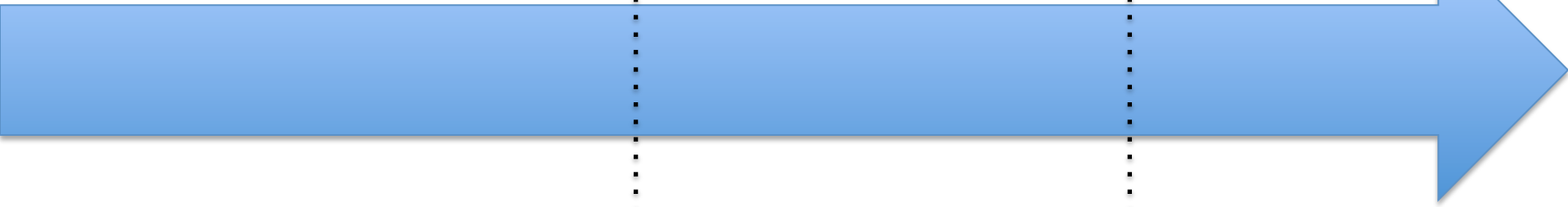
SP (sprints; implémentation)



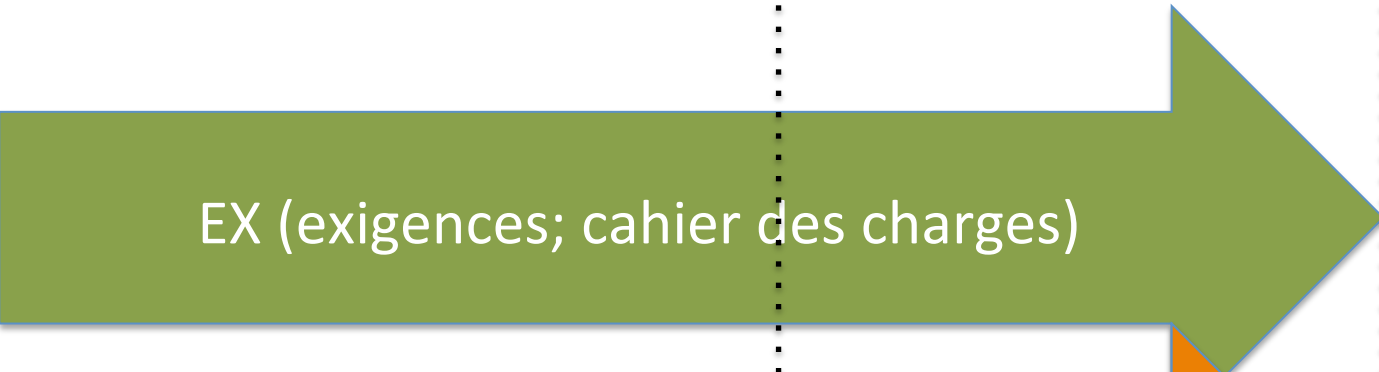
Execute the tests before/after each commit
Don't break (no regression)
Continuous validation

10 novembre

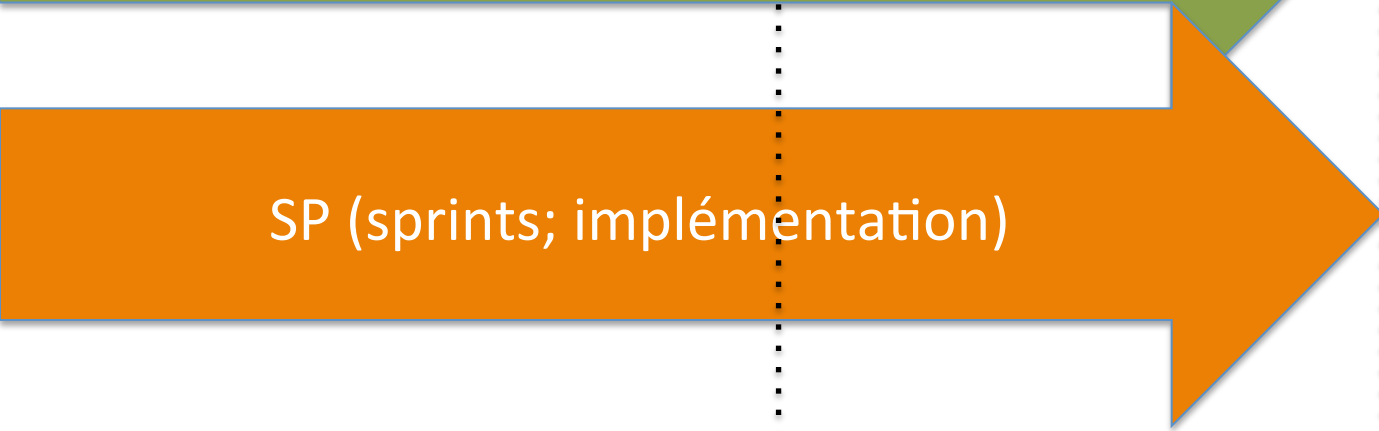
18 décembre



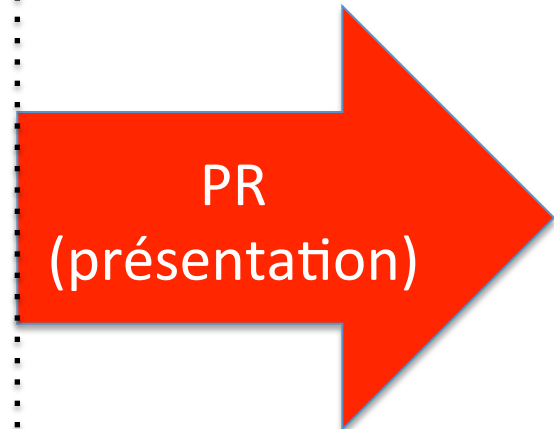
EX (exigences; cahier des charges)



SP (sprints; implémentation)



PR
(présentation)



Relationship with
PDL (your project)

Impacts

- Use/experiment with all these tools
 - IDE in general (Eclipse, IntelliJ, etc.) and all services...
 - Debugging
 - Refactoring
 - Testing
 - Documentation
 - Maven
 - Versioning systems (git)
- You will have to in your professional career!