Git and Maven

Web Applications

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia
Academic Year 2023/2024

Nicola Ferro

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

Outline

- Introduction to Git
- Introduction to Maven
- Use of Maven to compile, package, and document applications

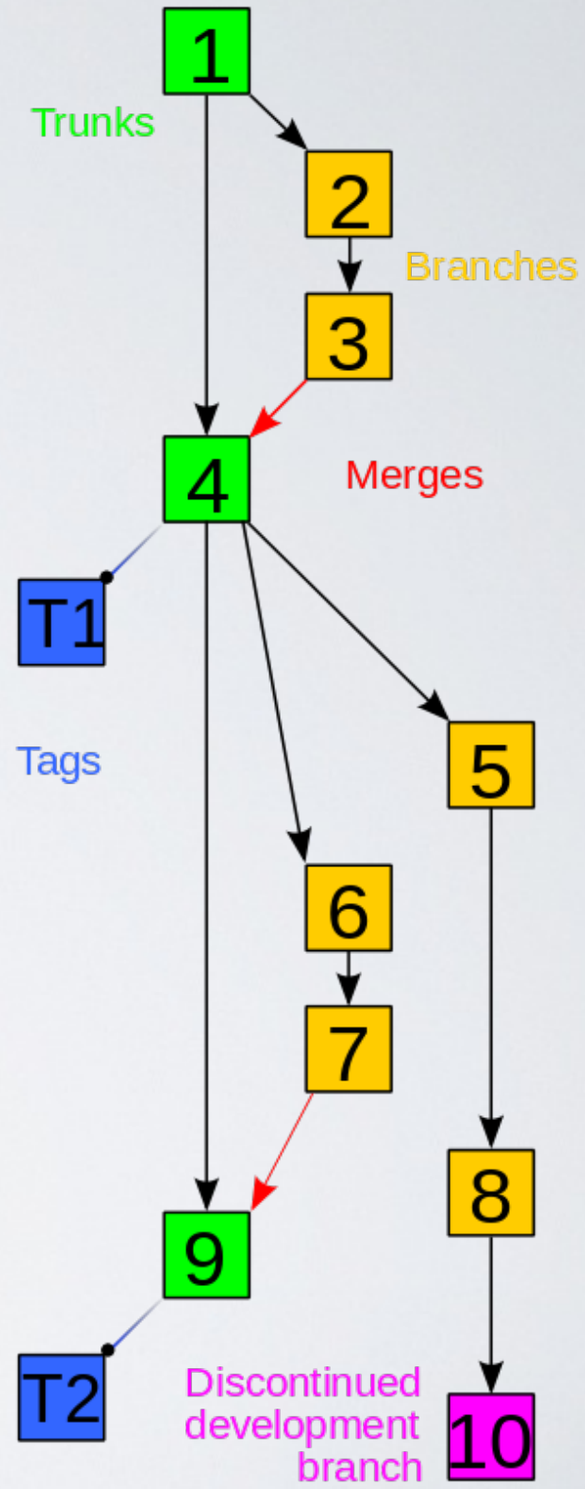
Git

Git

https://git-scm.com/

Source Code Management

- A **version control system** manages the versions, called **revisions**, of files and directories
  - it also manages conflicts, e.g. when the same file is edited concurrently, and their resolution (merge)
- **Centralized approach** (`cvs`, `svn`)
  - a single central repository manages all the versions of files and directories, allowing us to keep track of all the changes over time
  - the client uses a local copy of the files and keeps the synchronisation with the central repository
- **Distributed approach** (`git`)
  - the local copy of every client is a complete repository
  - the synchronisation happens exchanging patches among peers
- Code development is modelled ad a directed graph where, from the main development line (**master**), alternative development lines (**branch**) and/or stable versions (**tag**) can depart and join



Git: Creation of a New Repository

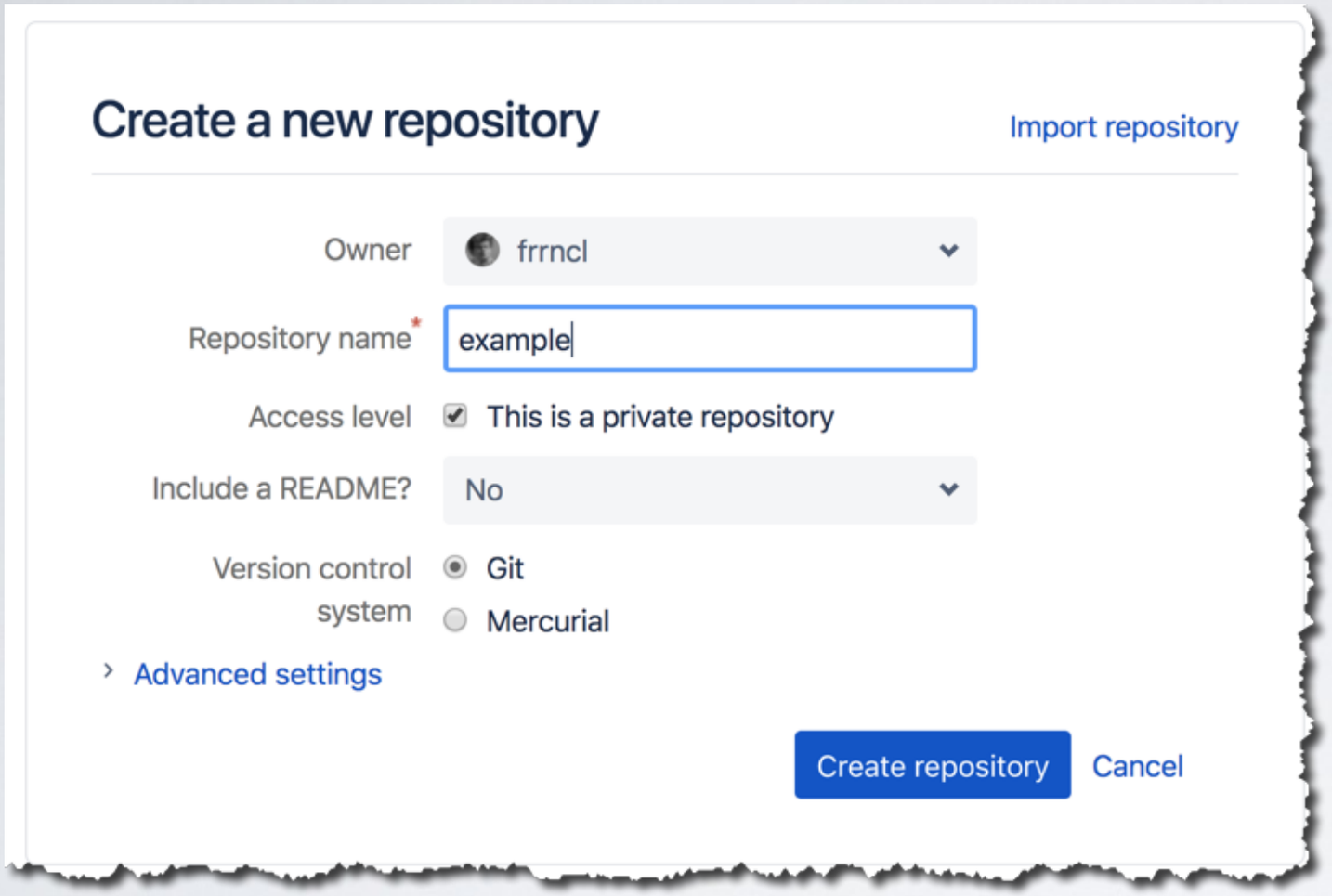
- To create a new repository, create a new folder on your disk and, within that folder, run

```
git init
```

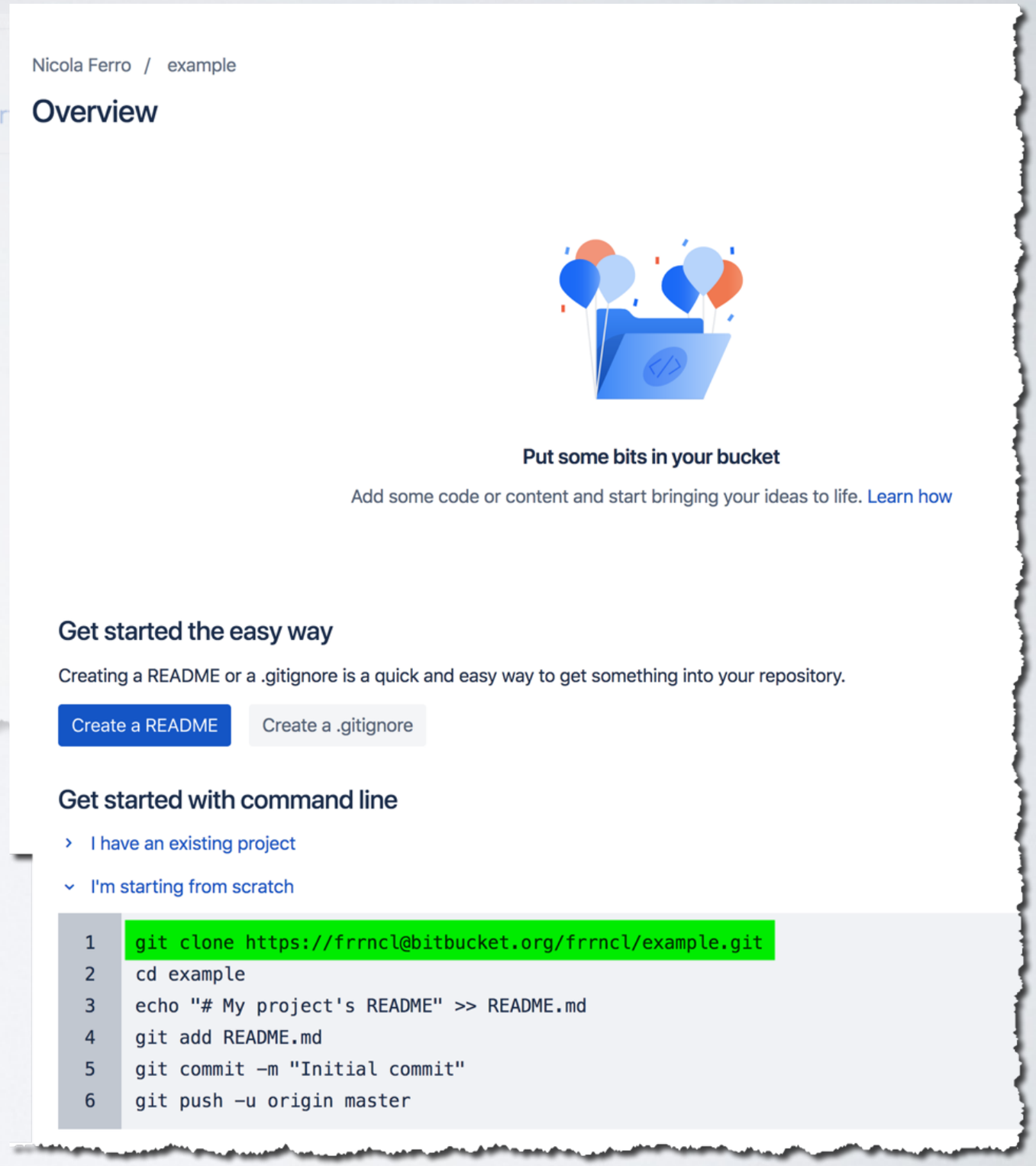
- To create a local copy of an existing repository (checkout)

```
git clone username@host:/path/to/repos
```

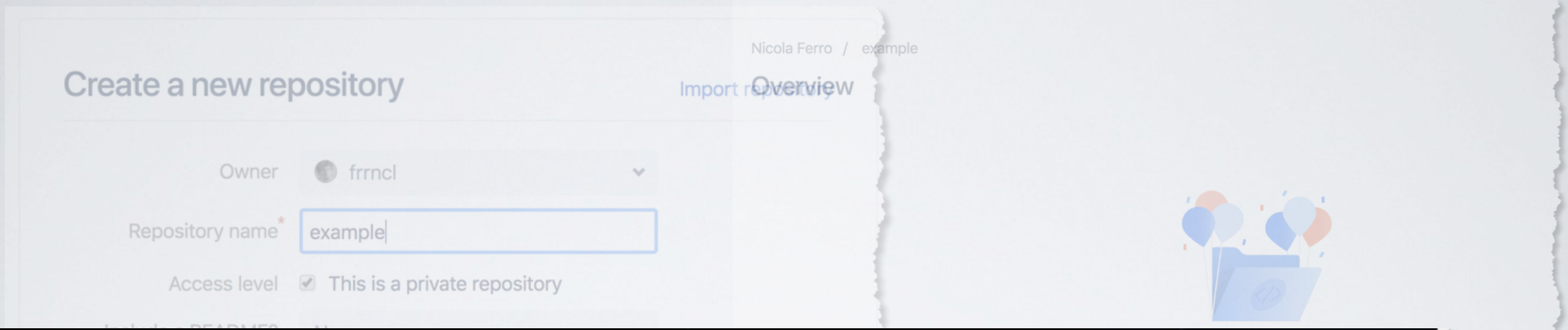
Cloning an Empty Bitbucket Repository



Cloning an Empty Bitbucket Repository

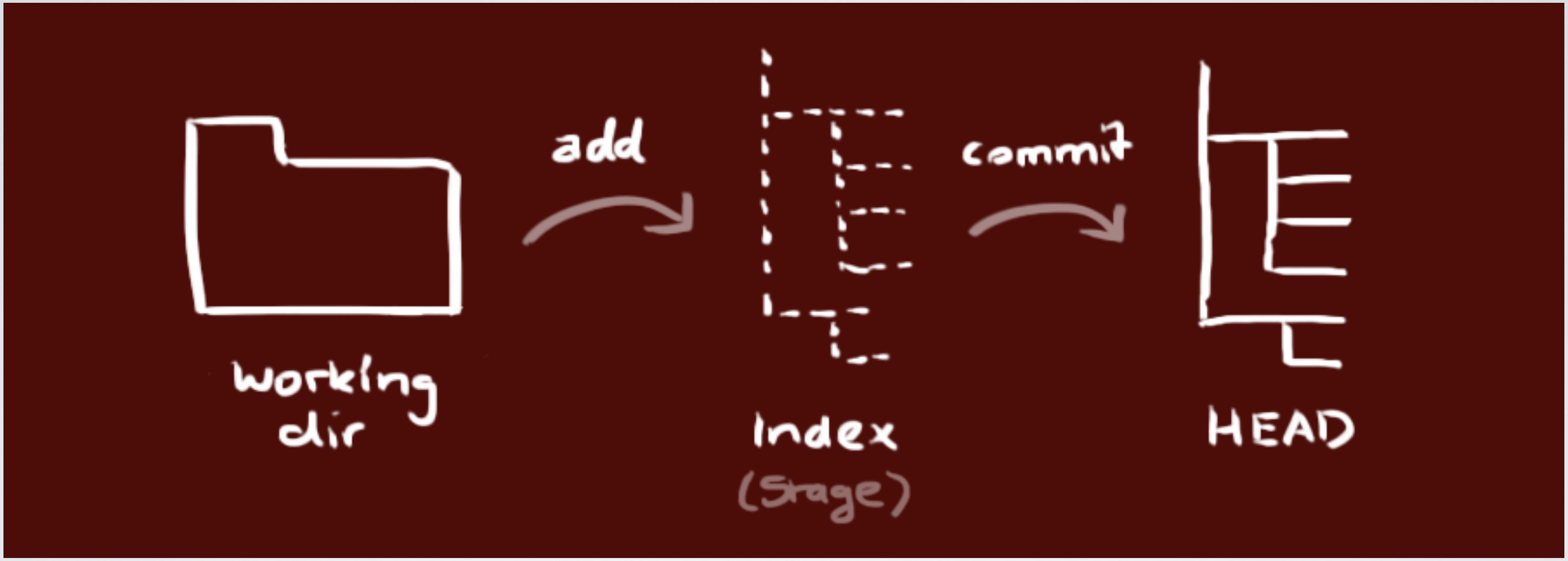


Cloning an Empty Bitbucket Repository




```
ferro@linux02-1:~$ git clone https://frrncl@bitbucket.org/frrncl/example.git
Cloning into 'example'...
Password for 'https://frrncl@bitbucket.org':
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
ferro@linux02-1:~$
```

Workflow



The local copy of a repository consists of three trees

- **Working directory**: keeps the actual files and directories, which may or may not be unversioned
- **Index**: is a staging area
- **HEAD**: represents the last commit made

Add and Commit

- You can add files/directories to the Index by

```
git add <filename>
```

- You can confirm updates and add the to the HEAD by

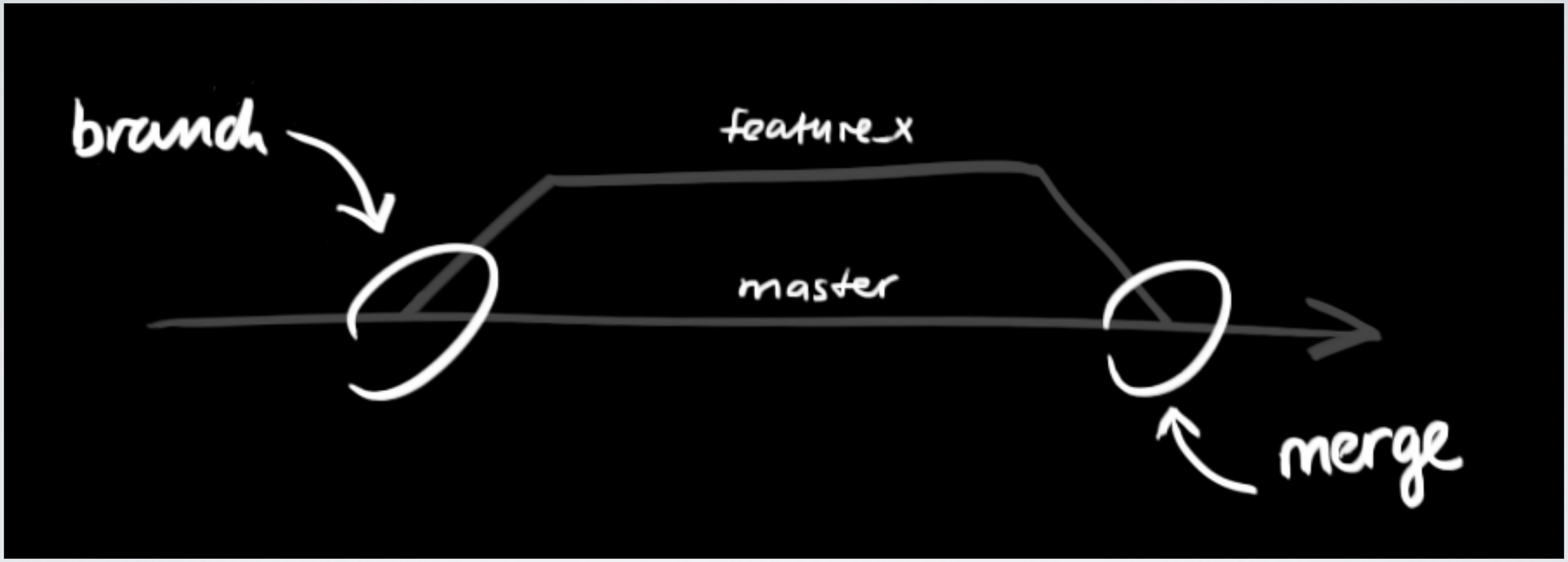
```
git commit -m "Description"
```

- You can send committed updates to a remote server by

```
git push origin master
```

  - `master` (or any other name) is the repository branch to send to the remote server
  - `origin` indicates the default remote repository, e.g. the one from which we cloned the repository

Branch



- Branches are used to develop independent features, e.g. new versions of a software
- The `master` branch is the default one when you create a new repository
- The other branches may be merged into the `master` one when appropriate

Branch Management

- To create a new branch

  `git checkout -b <branch-name>`

- To get back to the `master` branch (or any other branch name)

  `git checkout master`

- To send a branch to the remote repository

  `git push origin <branch-name>`

Update and Merge

- To update a local repository from a remote one

```
git pull origin <branch-name>
```

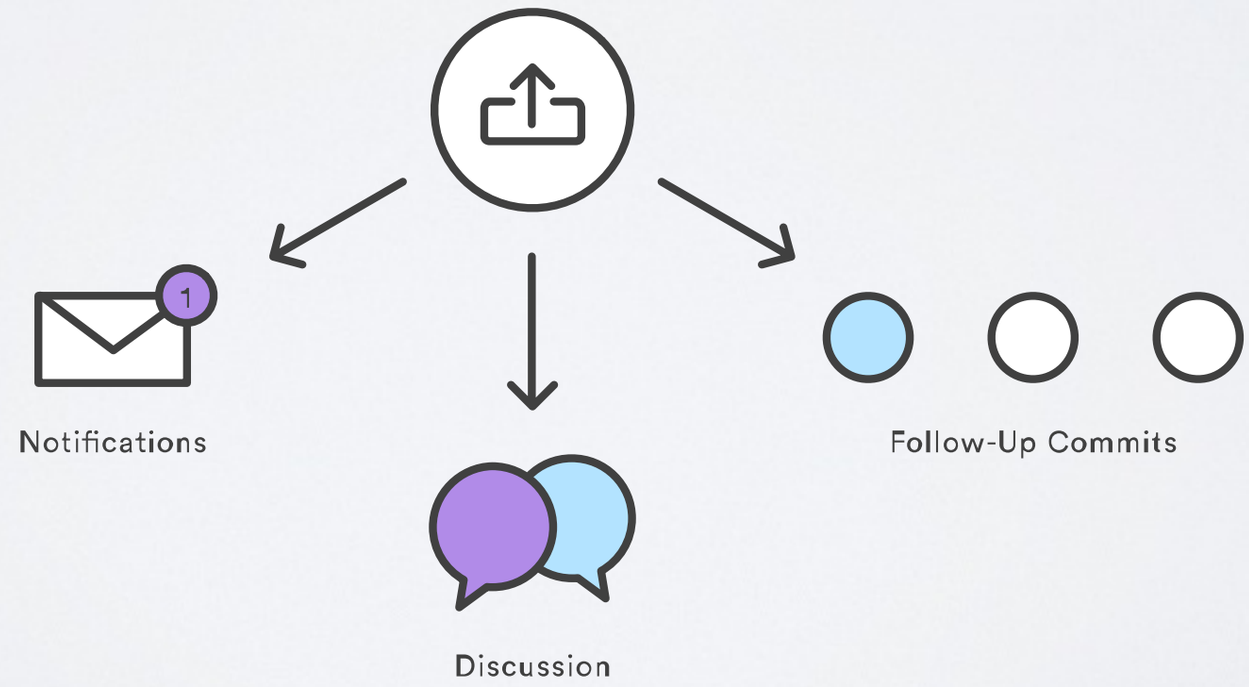
- To merge a branch into the currently selected one

```
git merge <branch-name>
```

Pull Requests

- Development platforms, such as GitHub and Bitbucket, provide **pull requests** as a mechanism to foster collaboration among developers
- Pull requests are a mechanism for a developer to notify team members that they have completed a feature.
- Once their feature branch is ready, the developer files a pull request.

This lets everybody involved know that they need to review and discuss the code and, eventually, merge it into the `master` branch



The .gitignore File

- The `.gitignore` file has to be put in the root folder of your development tree
- It specifies intentionally untracked files that Git should ignore
- Each line in a `.gitignore` file specifies a pattern to be matched to decide whether to exclude files and/or directories

`https://git-scm.com/docs/gitignore`

Example of .gitignore File

```
# IntelliJ Idea
*.iml
.idea/

# Java
*.class
target/
javadoc/
log/

# Package Files
*.jar
*.war
*.ear
*.zip
*.tar.gz
*.rar

### OSX ###
.DS_Store
```

The README File

- The `README` file has to be put in the root folder of your development tree
- It provides overall information about your project which are displayed on its Web page
  - you can use the Markdown syntax (`https://bitbucket.org/tutorials/markdowndemo`) to format it
- Example of `README.md` file

```
# Web Applications (webapp)

This directory contains the source code distribution complementing the lectures.

Web Applications lectures are held at:

* Master Degree in Computer Engineering
* Master Degree in ICT for Internet and Multimedia
* Master Degree in Cybersecurity

of the Department of Information Engineering, University of Padua, Italy

Copyright and license information can be found in the file LICENSE.
Additional information can be found in the file NOTICE.
```

Examples of Code

- All the code examples are available in the following Bitbucket repository

https://bitbucket.org/frrncl/webapp-unipd

- You can clone it and pull from it as it gets updated

Maven

Maven

- Maven, a Yiddish word meaning *accumulator of knowledge*, was originally started as an attempt to simplify the build processes in the Jakarta Turbine project, a servlet based framework to build secure web applications.
- Maven is a tool for managing Java software projects and supporting developers in keeping track of the status of a project
  - build
  - dependency management
  - deployment and packaging
  - collaboration and documentation
- Advantages
  - **coherence**: standardisation of the management of Java project, increased transparency and reduced time to get an understanding of the different projects of an organisation;
  - **reuse**: similar projects can reuse and extend the setup of previous projects;
  - **simplicity**: simplification of the creation and integration of new components as well as of the sharing of packages and executables. Moreover, the learning curve for each project is reduced;
  - **maintenance**: reduced effort and resources to keep building scripts as well as development and deployment environments

Maven Homepage

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

Maven™

Apache / Maven / Welcome to Apache Maven

Download | Get Sources | Last Published: 2024-02-26

- Welcome
- License

ABOUT MAVEN

- What is Maven?
- Features
- Download
- Use
- Release Notes

DOCUMENTATION

- Maven Plugins
- Maven Extensions
- Index (category)
- User Centre
- Plugin Developer Centre
- Maven Repository Centre
- Maven Developer Centre
- Books and Resources
- Security

COMMUNITY

- Community Overview
- Project Roles
- How to Contribute
- Getting Help

Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is and a list of some of its main features.

This site is separated into the following sections, depending on how you'd like to use Maven:

Use	**Download**, **Install**, **Configure**, **Run** Maven	**Maven Plugins** and **Maven Extensions**
	Information for those needing to build a project that uses Maven	Lists of plugins and extensions to help with your builds.
Extend	**Write Maven Plugins**	**Improve the Maven Central Repository**
	Information for developers writing Maven plugins.	Information for those who may or may not use Maven, but are interested in getting project metadata into the central repository.
Contribute	**Help Maven**	**Develop Maven**
	Information if you'd like to get involved. Maven is an open source community and welcomes contributions.	Information for those who are currently Maven developers, or who are interested in contributing to the Maven project itself.

Each guide is divided into a number of trails to get you started on a particular topic, and includes a reference area and a "cookbook" of common examples.

You can access the guides at any time from the left navigation. If you are looking for a quick reference, you can use the documentation index.

How to Get Support

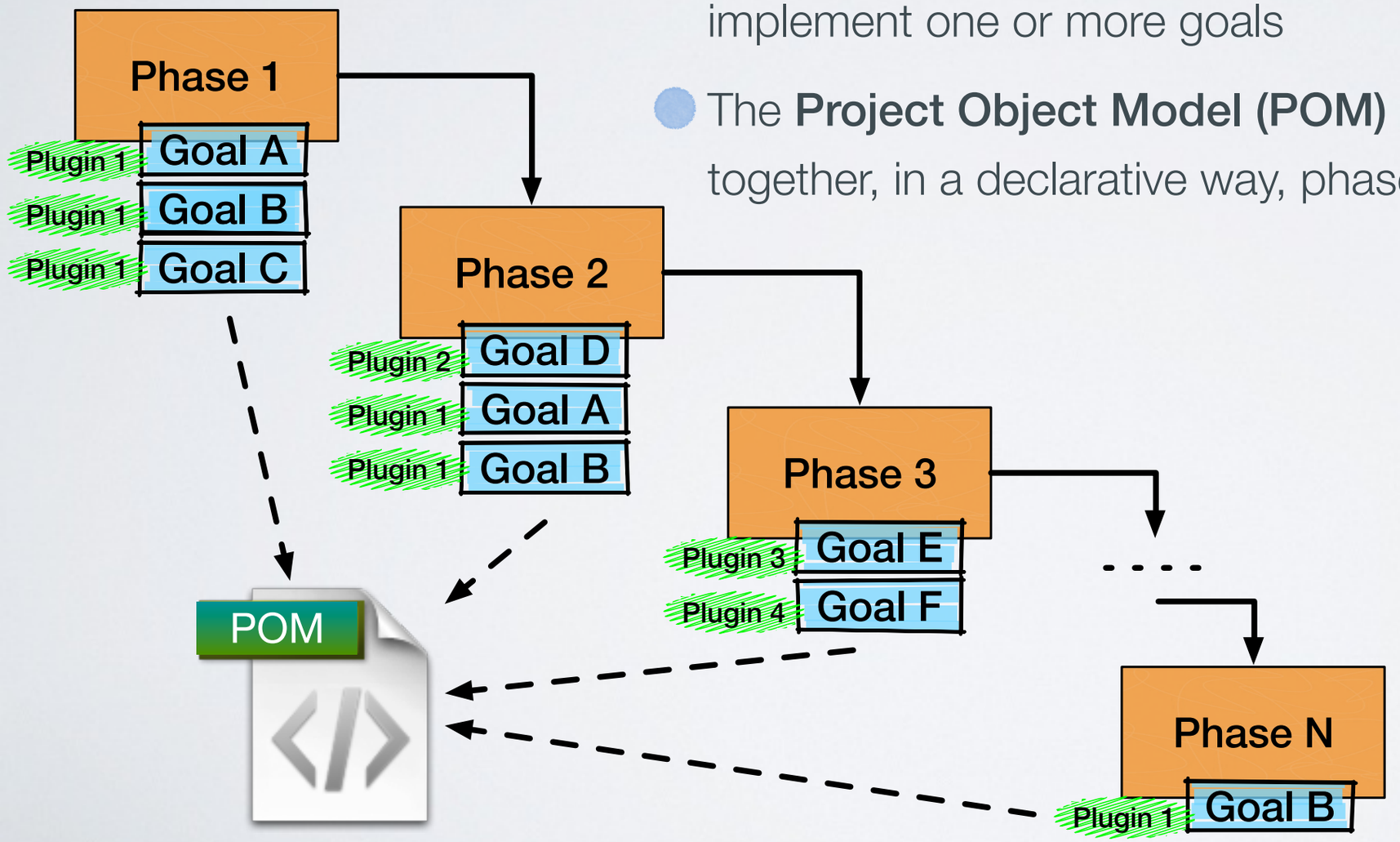
Support for Maven is available in a variety of different forms.

To get started, search the documentation, issue management system, the wiki or the mailing list archives to see if the problem has been solved or reported before.

http://maven.apache.org/

Maven: Main Concepts

- Software development happens according to a **life cycle** made up of **phases**
- Zero or more **goals** are associated to each phase and they are the operations actually carried out in that phase
- Goals are implemented by means of **plugins** and each plugin may implement one or more goals
- The **Project Object Model (POM)** is a single XML file which puts together, in a declarative way, phases, goals and plugins for a project



Build Lifecycle

- A build lifecycle is needed to create, compile, integrate, test, and distribute a software project
- The phases of a lifecycle are executed in sequence to complete that lifecycle
  - if you invoke an intermediate phase of a lifecycles, all the phases up to that phase will be execute
  - if you invoke the last phase, all the phases will be executed
- There are three predefined build lifecycles
  - **clean**: manages the cleaning of the project, i.e. it deletes all the files generated by a build
  - **default**: manages the whole development of the project
  - **site**: manages the creation of a project site and of the documentation

The Default Build LifeCycle

Setup of the project

- **validate**: validates the project is correct and all necessary information is available.
- **initialize**: initialize build state, e.g. set properties or create directories.

Source processing

- **generate-sources**: generates any source code for inclusion in compilation
- **process-sources**: processes the source code, for example to filter any values
- **generate-resources**: generates resources for inclusion in the package
- **process-resources**: copies and processes the resources into the destination directory, ready for packaging
- **compile**: compiles the source code of the project
- **process-classes**: post-processes the generated files from compilation, for example to do bytecode enhancement on Java classes

Testing

- **generate-test-sources**: generates any test source code for inclusion in compilation
- **process-test-sources**: processes the test source code, for example to filter any values
- **generate-test-resources**: creates resources for testing
- **process-test-resources**: copies and processes the resources into the test destination directory
- **test-compile**: compiles the test source code into the test destination directory
- **process-test-classes**: post-processes the generated files from test compilation, for example to do bytecode enhancement on Java classes
- **test**: runs tests using a suitable unit testing framework. These tests should not require the code be packaged or deployed

Packaging

- **prepare-package**: performs any operations necessary to prepare a package before the actual packaging.
- **package**: takes the compiled code and package it in its distributable format, such as a JAR

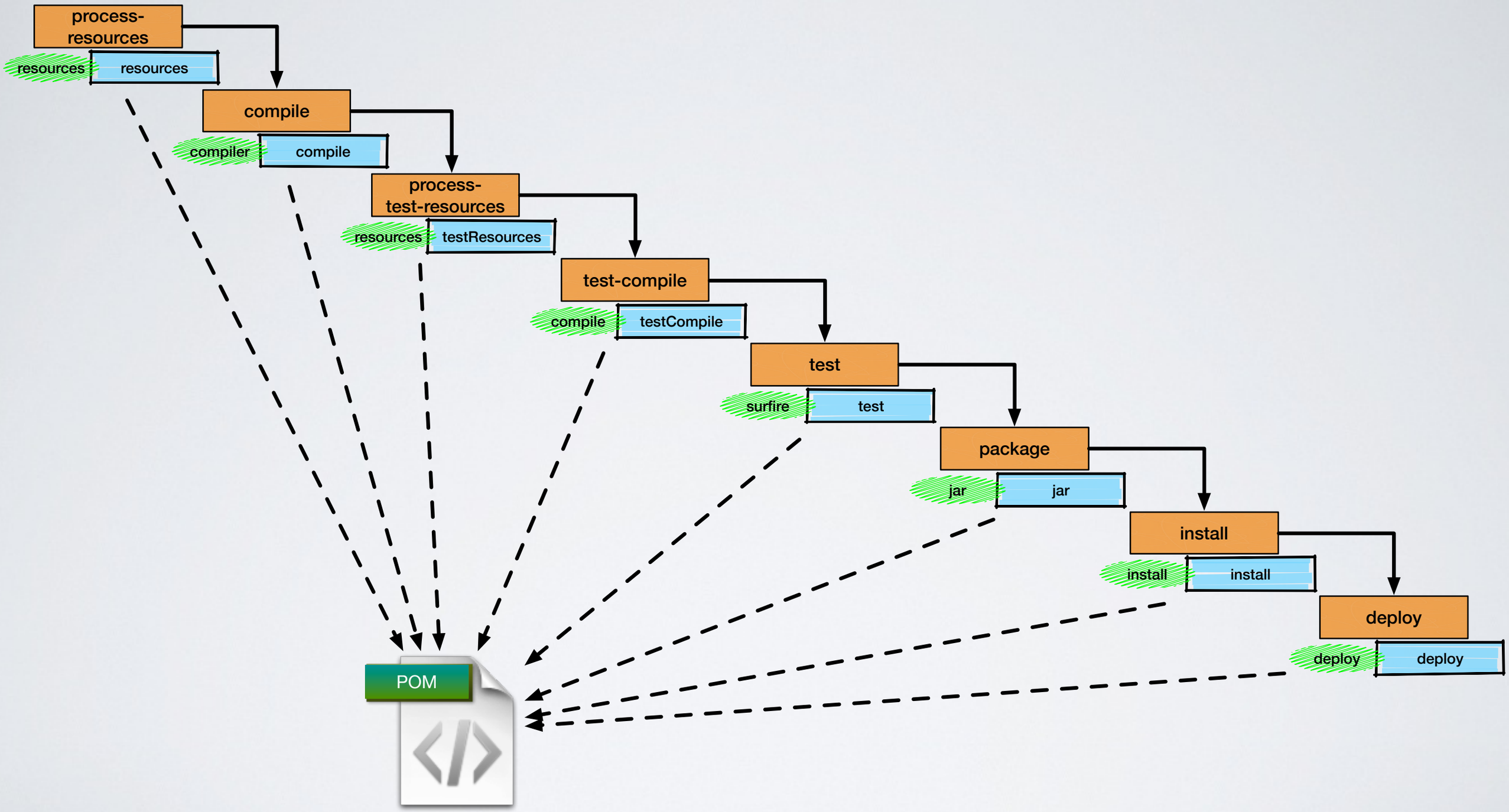
Integration

- **pre-integration-test**: performs actions required before integration tests are executed. This may involve things such as setting up the required environment.
- **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run
- **post-integration-test**: performs actions required after integration tests have been executed. This may including cleaning up the environment

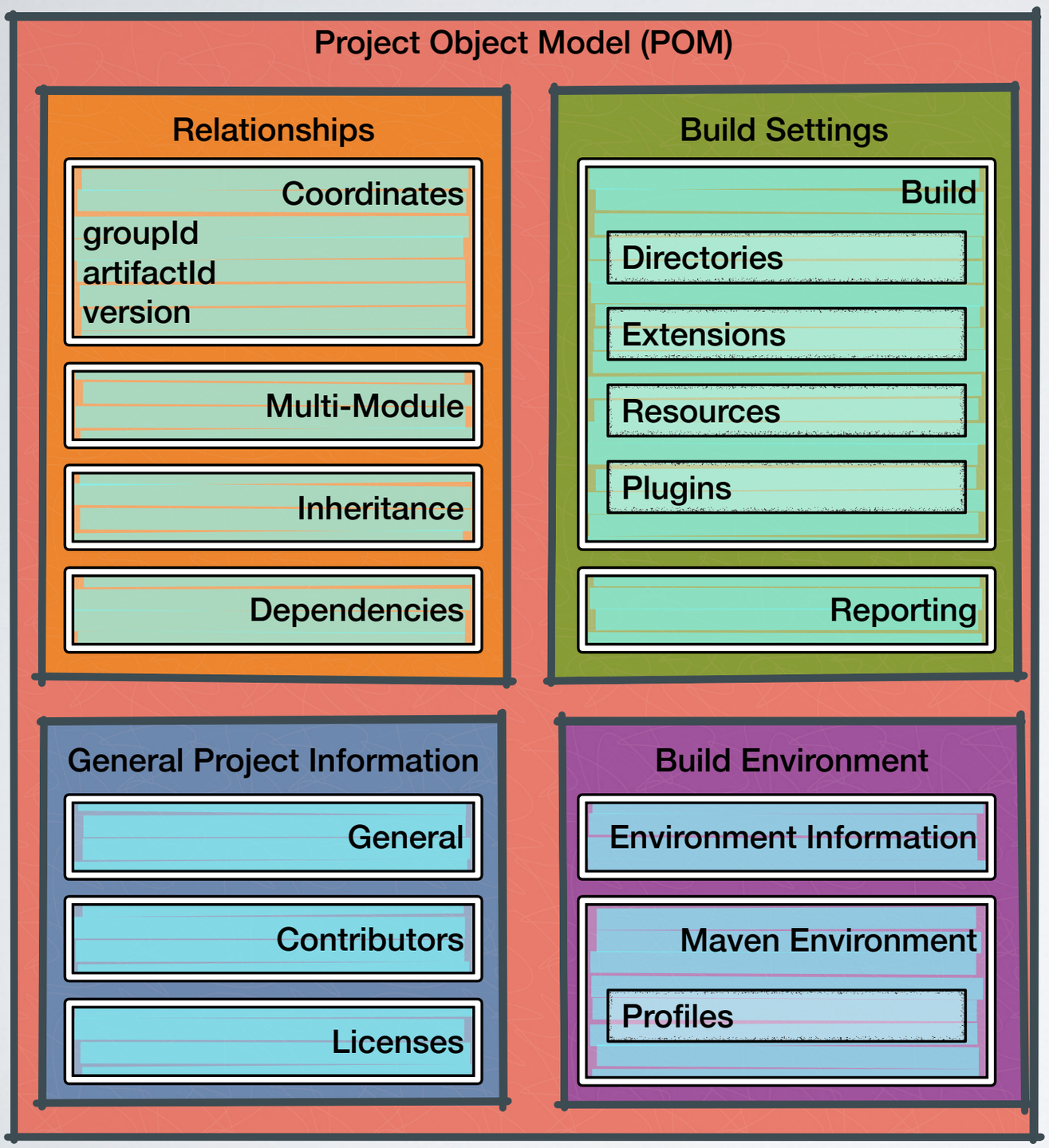
Deployment

- **verify**: runs any checks to verify the package is valid and meets quality criteria
- **install**: installs 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

Default Build Lifecycle for JAR Packages

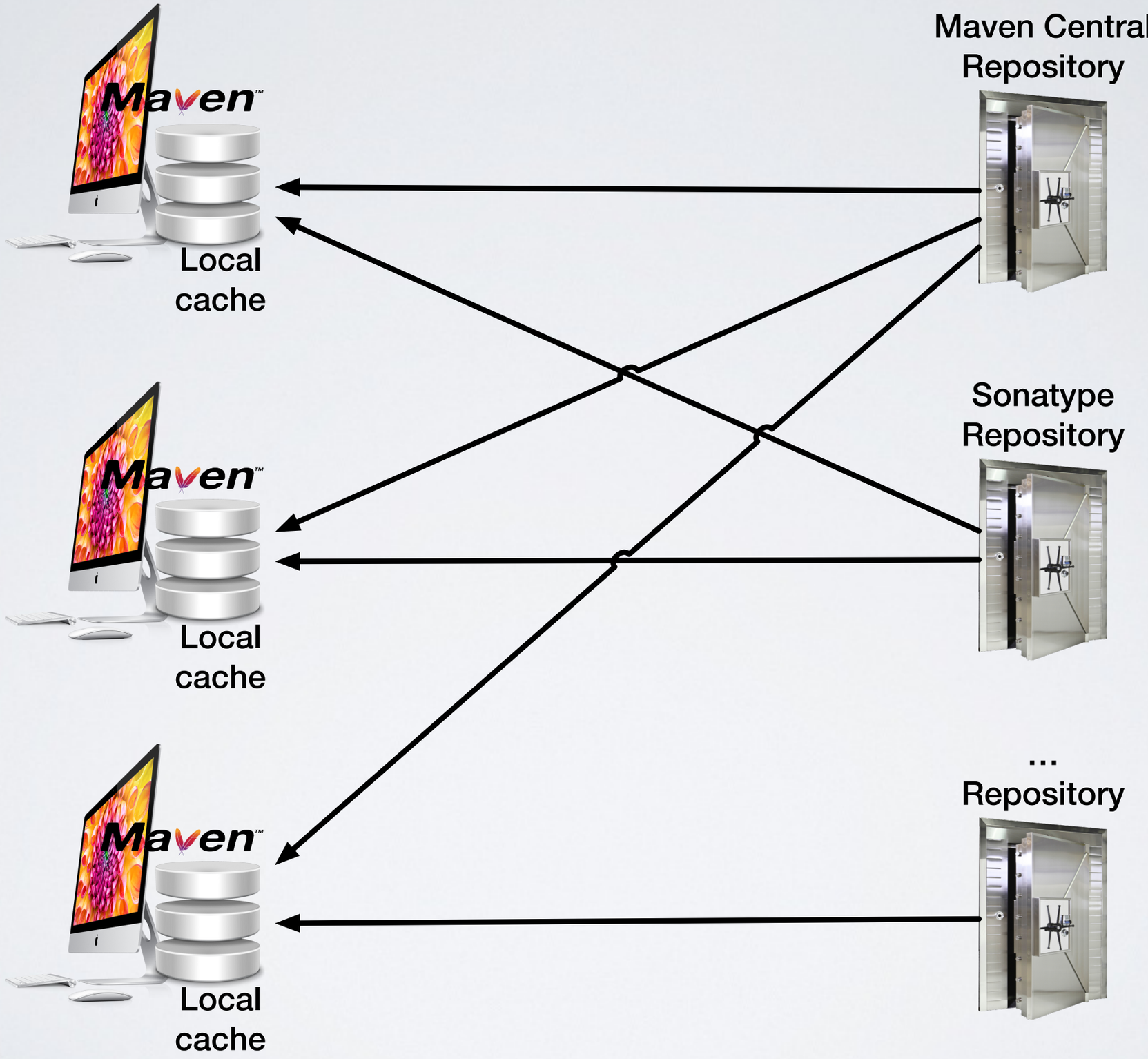


Project Object Model (POM)



- **Relationships**: defines the structure of the project (coordinates and modules), its relationships with other projects (inheritance), dependencies on other projects and libraries
- **General project information:** maintains general information about the project, such as, project name, project Web site, organization developing the project, developers, licences
- **Build settings**: customises the default build lifecycle by adding goals and plugins to the different phases as well as information about source, test, and resources
- **Build environment**: defines profiles corresponding to different environments and/or operating systems

Maven Repositories



Setting up Maven: the settings.xml file

- The **settings.xml** file contains the overall configuration for Maven
  - where to store the local cache for libraries and plugins
  - the configuration about a local repository, if any, and credentials to access it
- The `settings.xml` has to be saved in the **.m2** folder in each own home. If it does not already exist, you need to create it

Example of settings.xml

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <localRepository>/Users/ferro/.m2/repository</localRepository>
</settings>
```

Running Maven

```
mvn [options] [<goal(s)>] [<phase(s)>]
```

- **phase**: one or more phase names according to the available build lifecycles
  - remember that all the phases up to the selected one(s) will be executed
- **goal**: one or more goal names to be executed
  - goal names have the following formata:
    `<plugin-name>:<goal-name>`
- For example, the following command

```
mvn clean deploy checkstyle:check
```

  - invokes the `clean` phase of the clean build lifecycle
  - invokes the deploy `phase` (and all the phases before it) of the default build lifecycle
  - invokes the `check` goal of the `checkstyle` plugin

First Application without dependencies using Maven

Main Steps

- Only once: configure the `settings.xml` file in the `.m2` folder
- Create a repository in Bitbucket for your application
- Clone that repository on your local machine
  - add appropriate `.gitignore` and `README/README.md` files
- Create the directory structure and POM file, add source files, etc.
- Build and package with Maven
  - generate Javadoc
- Push to the repository

Setup the Project Directory Structure

- `src` development version
  - `main` sources for the main application
    - `database` sources for the database, e.g. schema creation SQL
    - `java` sources for the Java application
    - `resources` any additional application resource, e.g. property files
    - `webapp` sources for the Web application, e.g. HTML, CSS, JS
  - `test` sources for the test, e.g. JUnit
- `javadoc` documentation
- `target` folder for compiled code and packages

Project Object Model (POM)

```xml
<?xml version="1.0"?>

<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>it.unipd.dei.webapp</groupId>

    <artifactId>hello-world</artifactId>

    <version>1.00</version>

    <packaging>jar</packaging>

    <!-- Project description elements -->
    <name>Hello World</name>

    <description>Writes "Hello, world!" on the console</description>

    <url>http://www.dei.unipd.it/en/</url>

    <inceptionYear>2018</inceptionYear>

    <developers>
        <developer>
            <id>nf</id>
            <name>Nicola Ferro</name>
            <email>ferro@dei.unipd.it</email>
            <url>http://www.dei.unipd.it/~ferro/</url>
        </developer>
    </developers>

    <licenses>
        <license>
            <name>The Apache Software License, Version 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
            <distribution>repo</distribution>
        </license>
    </licenses>

    <organization>
        <name>Department of Information Engineering (DEI), University of Padua, Italy</name>
        <url>http://www.dei.unipd.it/en/</url>
    </organization>
```

Coordinates of the project

- `groupId`: a unique id of the “producer” of the project, e.g. the domain name of the company
- `artifactId`: the name of the project
- `version`: the version of the project
- `packaging`: how to package the project; `jar` for desktop applications; `war` for web

Various (optional) description elements about the project, e.g. the year the project started, the developers involved, the type of license, the organization running the project

Project Object Model (POM)

```xml
<!-- Specifies the encoding to be used for project source files -->
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<!-- Configuration of the default build lifecycle -->
<build>
    <defaultGoal>compile</defaultGoal>

    <!-- source code folder -->
    <sourceDirectory>${basedir}/src/main/java</sourceDirectory>

    <!-- compiled code folder -->
    <directory>${basedir}/target</directory>

    <!-- name of the generated package -->
    <finalName>${project.artifactId}-${project.version}</finalName>

    <!-- configuration of the plugins for the different goals -->
    <plugins>

        <!-- compiler plugin: source and target code is for Java 1.8 -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.7.0</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>


        <!-- javadoc plugin: output in the javadoc folder -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <version>3.0.0</version>
            <configuration>
                <reportOutputDirectory>${basedir}/javadoc</reportOutputDirectory>
                <author>true</author>
                <nosince>false</nosince>
                <show>protected</show>
            </configuration>
        </plugin>

    </plugins>
</build>
</project>
```

Project files are encoded in UTF-8

Source input and compiled output folders. You can omit these if you use the default directory structure: **convention over configuration**

Name of the generated `jar`/`war` package file

Source code and generated class files are for Java 1.8

Configuration for generating the Javadoc

- `reportOutputDirectory`: where to write the generated Javadoc
- `author`: whether to print the `@author` tag
- `nosince`: whether to print the `@since` tag
- `show`: the scope of methods/variables which has to be reported

The HelloWorld Class

```java
package it.unipd.dei.webapp;

/**
 * Sample class to say "Hello, world".
 *
 * @author Nicola Ferro  (ferro@dei.unipd.it)
 * @version 1.0
 * @since 1.0
 */
public class HelloWorld {

    /**
     * Main method of the class.
     *
     * Just prints "Hello, world!".
     *
     * @param args input arguments from the command line, if any.
     */
    public static void main(String[] args) {

        System.out.printf("Hello, world!%n");

    }

}
```

Cleaning, Compiling, Packaging, Documenting, and Running

```
ferro~hello-world$ mvn clean package javadoc:javadoc
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread count of 1
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Hello World 1.00
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hello-world ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello-world ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ hello-world ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ hello-world ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ hello-world ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ hello-world ---
[INFO]
```

Cleaning, Compiling, Packaging, Documenting, and Running

```
ferro~hello-world$ ls -la
total 24
drwxr-xr-x   7 ferro  staff   224 16 Apr 15:41 .
drwxr-xr-x  12 ferro  staff   384 16 Apr 15:41 ..
-rw-r--r--@  1 ferro  staff  6148 16 Apr 12:19 .DS_Store
drwxr-xr-x   3 ferro  staff    96 16 Apr 15:41 javadoc
-rw-r--r--@  1 ferro  staff  3254 16 Apr 12:18 pom.xml
drwxr-xr-x   4 ferro  staff   128 16 Apr 10:23 src
drwxr-xr-x   9 ferro  staff   288 16 Apr 15:41 target
ferro~hello-world$
ferro~hello-world$ ls javadoc/apidocs/
allclasses-frame.html           it                              package-search-index.zip
allclasses-noframe.html         jquery                          resources
constant-values.html            member-search-index.js          script.js
deprecated-list.html            member-search-index.zip         search.js
help-doc.html                   overview-tree.html              stylesheet.css
index-all.html                  package-list                    type-search-index.js
index.html                      package-search-index.js         type-search-index.zip
ferro~hello-world$
ferro~hello-world$ ls target/
classes                 hello-world-1.00.jar    maven-archiver          test-classes
generated-sources       javadoc-bundle-options  maven-status
ferro~hello-world$
```

Cleaning, Compiling, Packaging, Documenting, and Running

```
ferro~hello-world$ java -cp target/hello-world-1.00.jar it.unipd.dei.webapp.HelloWorld
Hello, world!
ferro~hello-world$
```

Cleaning, Compiling, Packaging, Documenting, and Running

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED I FIELD I CONSTR I METHOD DETAIL: FIELD I CONSTR I METHOD

SEARCH: Search

Package it.unipd.dei.webapp

Class HelloWorld

java.lang.Object
it.unipd.dei.webapp.HelloWorld

```
public class HelloWorld
extends Object
```

Sample class to say "Hello, world".

Since:

1.0

Version:

1.0

Author:

Nicola Ferro (ferro@dei.unipd.it)

Constructor Summary

Constructors

Constructor	Description
`HelloWorld()`	

Method Summary

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String[] args)	Main method of the class.

First Application with dependencies using Maven

JFigLet

https://github.com/dtmo/jfiglet

Banner

```
######
#     #  ##   #    # #    # ###### #####
#     # #  #  #    # ##   # #      #    #
######  #    # #  # # #  # #####  #    #
#     # ###### #  # # #  # #      #####
#     # #    # #   ## #   ## #      #   #
######  #    # #    # #    # ###### #    #
```

Big

```
  ___  _
 |  _ \(_)
 | |_) |_  __ _
 |  _ <| |/ _` |
 | |_) | | (_| |
 |____/|_|\__, |
           __/ |
          |___/
```

Block

```
_|_|_|    _|                      _|
_|    _|  _|    _|_|      _|_|_|  _|  _|
_|_|_|    _|  _|    _|  _|        _|_|
_|    _|  _|  _|    _|  _|        _|  _|
_|_|_|    _|    _|_|      _|_|_|  _|    _|
```

Bubble

```
  _   _   _   _   _   _
 / \ / \ / \ / \ / \ / \
( B | u | b | b | l | e )
 \_/ \_/ \_/ \_/ \_/ \_/
```

Digital

```
+-+-+-+-+-+-+-+
|D|i|g|i|t|a|l|
+-+-+-+-+-+-+-+
```

The HelloWorldFigLet Class

Print help and exit, if the are no arguments

Name of the Figlet font to be used and “parsing” of the command line

Create a new Figlet renderer, render the “Hello, world!” string, and print it to the terminal

```java
package it.unipd.dei.webapp;

import com.github.dtmo.jfiglet.FigFontResources;
import com.github.dtmo.jfiglet.FigletRenderer;

import java.io.IOException;

public class HelloWorldFiglet {

    public static void main(String[] args) throws IOException {

        if (args.length == 0) {
            System.out.printf("Please, pick on of the following Figlet fonts:%n");
            System.out.printf("- Banner%n");
            System.out.printf("- Big%n");
            System.out.printf("- Block%n");
            System.out.printf("- Bubble%n");
            System.out.printf("- Digital%n");
            System.out.printf("- Ivrit%n");
            System.out.printf("- Lean%n");
            System.out.printf("- Mini%n");
            System.out.printf("- Mnemonic%n");
            System.out.printf("- Script%n");
            System.out.printf("- Shadow%n");
            System.out.printf("- Slant%n");
            System.out.printf("- Small%n");
            System.out.printf("- SmScript%n");
            System.out.printf("- SmShadow%n");
            System.out.printf("- SmSlant%n");
            System.out.printf("- Standard%n");
            System.out.printf("- Terminal%n");

            System.exit(0);
        }

        // name of the font to be used
        final String font;

        // "parse" the command line and set the proper Figlet font name or throw an error
        switch (args[0].trim().toLowerCase()) {
            case "banner":
                font = FigFontResources.BANNER_FLF;
                break;
            case "big":
                font = FigFontResources.BIG_FLF;
                break;
            case "block":
                font = FigFontResources.BLOCK_FLF;
                break;
            case "bubble":
                font = FigFontResources.BUBBLE_FLF;
                break;
            case "digital":
                font = FigFontResources.DIGITAL_FLF;
                break;
            case "ivrit":
                font = FigFontResources.IVRIT_FLF;
                break;
            case "lean":
                font = FigFontResources.LEAN_FLF;
                break;
            case "mini":
                font = FigFontResources.MINI_FLF;
                break;
            case "mnemonic":
                font = FigFontResources.MNEMONIC_FLF;
                break;
            case "script":
                font = FigFontResources.SCRIPT_FLF;
                break;
            case "shadow":
                font = FigFontResources.SHADOW_FLF;
                break;
            case "slant":
                font = FigFontResources.SLANT_FLF;
                break;
            case "small":
                font = FigFontResources.SMALL_FLF;
                break;
            case "smscript":
                font = FigFontResources.SMSCRIPT_FLF;
                break;
            case "smshadow":
                font = FigFontResources.SMSHADOW_FLF;
                break;
            case "smslant":
                font = FigFontResources.SMSLANT_FLF;
                break;
            case "standard":
                font = FigFontResources.STANDARD_FLF;
                break;
            case "terminal":
                font = FigFontResources.TERM_FLF;
                break;
            default:
                throw new IllegalArgumentException("Invalid Figfont: " + args[0]);
        }

        // render to write ASCII-art with the given font
        final FigletRenderer figletRenderer = new FigletRenderer(FigFontResources.loadFigFontResource(font));

        // ASCII-art
        final String output = figletRenderer.renderText("Hello, world!");

        // write to the console
        System.out.printf("%s%n", output);
    }

}
```

Looking for the JFiglet Library Dependency

http://search.maven.org/

sonatype | maven central repository

Browse

Official search by the maintainers of Maven Central Repository

Discover Java packages, and publish your own

jfiglet or... Browse all components

jfiglet 0.0.9 published 10 months ago in com.github.lalyos

jfiglet 1.0.1 published 2 years ago in com.github.dtmo.jfiglet

jfiglet-maven-plugin 0.0.2 published 4 years ago in io.github.mcwarman

Most Popular Packages in Last 90 Days

slf4j-nop org.slf4j

Version 2.0.6 View all

Published 2 months ago

Used in 220 projects

slf4j-log4j12 org.slf4j

Version 2.0.6 View all

Published 2 months ago

Used in 66 projects

log4j-over-slf4j org.slf4j

Version 2.0.6 View all

org.slf4j

24 projects

org.apache.commons

127 projects

com.fasterxml.jackson.core

3 projects

commons-io

1 project

Popular Categories

Build Tools

MULTI-PROJECT

Other

Data Management

Programming Language Utilities

Cloud Computing

Developer Tools

Security

Looking for the JFiglet Library Dependency

http://search.maven.org/

sonatype | maven central repository

jfiglet

Browse

Filtering

Namespace

e.g. org.mycompany...

Dependency Of

e.g. pkg:maven/mynamespace...

Dependent On

e.g. pkg:maven/mynamespace...

Contributor Email

e.g. contributor@email.server...

Category

e.g. License Management...

Search Results for jfiglet

Sort by: Best Match

Showing 3 results out of the 523386 available packages

jfiglet com.github.lalyos

No description provided

#Programming Language Utilities

Latest version 0.0.9 View all
Published 10 months ago
Licenses GNU General Public License v2.0
Used in 12 projects
BOM Doctor Report View
OSS Index No vulnerabilities found View

jfiglet com.github.dtmo.jfiglet

No description provided

Latest version 1.0.1 View all
Published 2 years ago
Licenses The 3-Clause BSD License
Used in 34 projects
Sonatype Safety Rating 5 out of 10
BOM Doctor Report View
OSS Index No vulnerabilities found View

jfiglet-maven-plugin io.github.mcwarman

No description provided

Latest version 0.0.2 View all
Published 4 years ago
Licenses MIT License

Looking for the JFiglet Library Dependency

http://search.maven.org/

sonatype | maven central repository jfiglet Browse

jfiglet

1.0.1

pkg:maven/com.github.dtmo.jfiglet/jfiglet@1.0.1

Overview Versions Dependents Dependencies

Overview

Java FIGfont rendering API

Snippets

Apache Maven Copy to clipboard

```
<dependency>
    <groupId>com.github.dtmo.jfiglet</groupId>
    <artifactId>jfiglet</artifactId>
    <version>1.0.1</version>
</dependency>
```

POM Files

Copy to clipboard

Sonatype Safety Rating

An aggregate rating designed to represent a project's readiness against vulnerabilities.

5 out of 10

How did we get this score?

Metadata

2 years ago

Licenses
The 3-Clause BSD License

12.3 kB

Organization

com.github.dtmo.jfiglet

The Updated POM File

```xml
<!-- generates jar files including any dependencies -->
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>3.3.0</version>
    <configuration>
        <descriptorRefs>
            <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
    </configuration>
    <executions>
        <execution>
            <id>make-assembly</id> <!-- this is used for inheritance merges -->
            <phase>package</phase> <!-- bind to the packaging phase -->
            <goals>
                <goal>single</goal> <!-- the only goal of the assembly plugin -->
            </goals>
        </execution>
    </executions>
</plugin>

    </plugins>
</build>

<!-- Dependencies -->
<dependencies>
    <dependency>
        <groupId>com.github.dtmo.jfiglet</groupId>
        <artifactId>jfiglet</artifactId>
        <version>1.0.1</version>
    </dependency>
</dependencies>
</project>
```

The name of the `jar` file including dependencies will be appended with `jar-with-dependencies`

Binds the `single` goal of the `maven-assembly-plugin` to the `package` phase of the default build lifecycle

Adds the dependency on the JFiglet library

Cleaning, Compiling, Packaging, Documenting, and Running

```
V hello-world-figlet % mvn clean package javadoc:javadoc
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------< it.unipd.dei.webapp:hello-world-figlet >---------------
[INFO] Building Hello World Figlet 1.00
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/com/github/dtmo/jfiglet/jfiglet/1.0.1/jfiglet-1.0.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/dtmo/jfiglet/jfiglet/1.0.1/jfiglet-1.0.1.pom (1.3 kB at 4.2 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/com/github/dtmo/jfiglet/jfiglet/1.0.1/jfiglet-1.0.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/dtmo/jfiglet/jfiglet/1.0.1/jfiglet-1.0.1.jar (94 kB at 1.4 MB/s)
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ hello-world-figlet ---
[INFO] Deleting /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello-world-figlet ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ hello-world-figlet ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ hello-world-figlet ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ hello-world-figlet ---
[INFO] Changes detected - recompiling the module!
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ hello-world-figlet ---
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ hello-world-figlet ---
[INFO] Building jar: /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/target/hello-world-figlet-1.00.jar
[INFO]
[INFO] --- maven-assembly-plugin:3.3.0:single (make-assembly) @ hello-world-figlet ---
[INFO] Building jar: /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/target/hello-world-figlet-1.00-jar-with-depende
ncies.jar
[INFO]
[INFO] >>> maven-javadoc-plugin:3.2.0:javadoc (default-cli) > generate-sources @ hello-world-figlet >>>
[INFO]
[INFO] <<< maven-javadoc-plugin:3.2.0:javadoc (default-cli) < generate-sources @ hello-world-figlet <<<
[INFO]
[INFO]
[INFO] --- maven-javadoc-plugin:3.2.0:javadoc (default-cli) @ hello-world-figlet ---
[INFO] No previous run data found, generating javadoc.
[INFO]
Loading source files for package it.unipd.dei.webapp...
Constructing Javadoc information...
Standard Doclet version 14.0.1
Building tree for all the packages and classes...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/it/unipd/dei/webapp/HelloWorldFiglet.html
...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/it/unipd/dei/webapp/package-summary.html.
..
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/it/unipd/dei/webapp/package-tree.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/constant-values.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/it/unipd/dei/webapp/class-use/HelloWorldF
iglet.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/it/unipd/dei/webapp/package-use.html...
Building index for all the packages and classes...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/overview-tree.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/deprecated-list.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/index-all.html...
Building index for all classes...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/allclasses-index.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/allpackages-index.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/system-properties.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/index.html...
Generating /Users/ferro/Documents/progetti/software/webapp-unipd/hello-world-figlet/javadoc/apidocs/help-doc.html...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.162 s
[INFO] Finished at: 2021-02-21T11:05:43+01:00
[INFO] ------------------------------------------------------------------------
```

Cleaning, Compiling, Packaging, Documenting, and Running

```
[√ hello-world-figlet % ls -la target
total 224
drwxr-xr-x  13 ferro  staff    416 21 Feb 11:05 .
drwxr-xr-x   9 ferro  staff    288 21 Feb 11:05 ..
drwxr-xr-x   2 ferro  staff     64 21 Feb 11:05 archive-tmp
drwxr-xr-x   3 ferro  staff     96 21 Feb 11:05 classes
drwxr-xr-x   3 ferro  staff     96 21 Feb 11:05 generated-sources
drwxr-xr-x   3 ferro  staff     96 21 Feb 11:05 generated-test-sources
-rw-r--r--   1 ferro  staff  99521 21 Feb 11:05 hello-world-figlet-1.00-jar-with-dependencies.jar
-rw-r--r--   1 ferro  staff   5523 21 Feb 11:05 hello-world-figlet-1.00.jar
drwxr-xr-x   4 ferro  staff    128 21 Feb 11:05 javadoc-bundle-options
drwxr-xr-x   3 ferro  staff     96 21 Feb 11:05 maven-archiver
-rw-r--r--   1 ferro  staff   3822 21 Feb 11:05 maven-javadoc-plugin-stale-data.txt
drwxr-xr-x   3 ferro  staff     96 21 Feb 11:05 maven-status
drwxr-xr-x   2 ferro  staff     64 21 Feb 11:05 test-classes
√ hello-world-figlet %
```

Cleaning, Compiling, Packaging, Documenting, and Running

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Package it.unipd.dei.webapp

Class HelloWorldFiglet

java.lang.Object
  it.unipd.dei.webapp.HelloWorldFiglet

```
public class HelloWorldFiglet
extends Object
```

Sample class to say "Hello, world", using JFiglet. See https://github.com/dtmo/jfiglet.

Since:

1.0

Version:

1.0

Author:

Nicola Ferro (ferro@dei.unipd.it)

Constructor Summary

Constructors

Constructor	Description
`HelloWorldFiglet()`	

Method Summary

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`main(String[] args)`	Main method of the class.

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Details

HelloWorldFiglet

```
public HelloWorldFiglet()
```

Method Details

main

```
public static void main(String[] args)
                throws IOException
```

Main method of the class. Prints "Hello, world!", using JFiglet. You can pick one of the following Figlet fonts:

- Banner
- Big
- Block
- Bubble
- Digital
- Ivrit
- Lean
- Mini
- Mnemonic
- Script
- Shadow
- Slant
- Small
- SmScript
- SmShadow
- SmSlant
- Standard
- Terminal

Parameters:

`args` - command line arguments. `args[0]` is one of the Figlet fonts above.

Throws:

`IOException` - if unable to load the required font.

Cleaning, Compiling, Packaging, Documenting, and Running

```
[?1 hello-world-figlet % java -cp target/hello-world-figlet-1.00-jar-with-dependencies.jar it.unipd.dei.webapp.HelloWorldFiglet Slant
    __  __     ____                             __    ____
   / / / /__  / / /___      _      ______  _____/ /___/ / /
  / /_/ / _ \/ / / __ \    | | /| / / __ \/ ___/ / __  / /
 / __  /  __/ / / /_/ /    | |/ |/ / /_/ / /  / / /_/ /_/
/_/ /_/\___/_/_/\____( )   |__/|__/\____/_/  /_/\__,_(_)
                     |/
√ hello-world-figlet %
```

```
[√ hello-world-figlet % java -cp target/hello-world-figlet-1.00-jar-with-dependencies.jar it.unipd.dei.webapp.HelloWorldFiglet
Please, pick on of the following Figlet fonts:
- Banner
- Big
- Block
- Bubble
- Digital
- Ivrit
- Lean
- Mini
- Mnemonic
- Script
- Shadow
- Slant
- Small
- SmScript
- SmShadow
- SmSlant
- Standard
- Terminal
√ hello-world-figlet %
```

```
[√ hello-world-figlet % java -cp target/hello-world-figlet-1.00-jar-with-dependencies.jar it.unipd.dei.webapp.HelloWorldFiglet Bigger
Exception in thread "main" java.lang.IllegalArgumentException: Invalid Figfont: Bigger
        at it.unipd.dei.webapp.HelloWorldFiglet.main(HelloWorldFiglet.java:153)
?1 hello-world-figlet %
```