Maven & Git & PR

Maven

Why do we need Maven?

Easy to download add and remove the dependencies/libraries

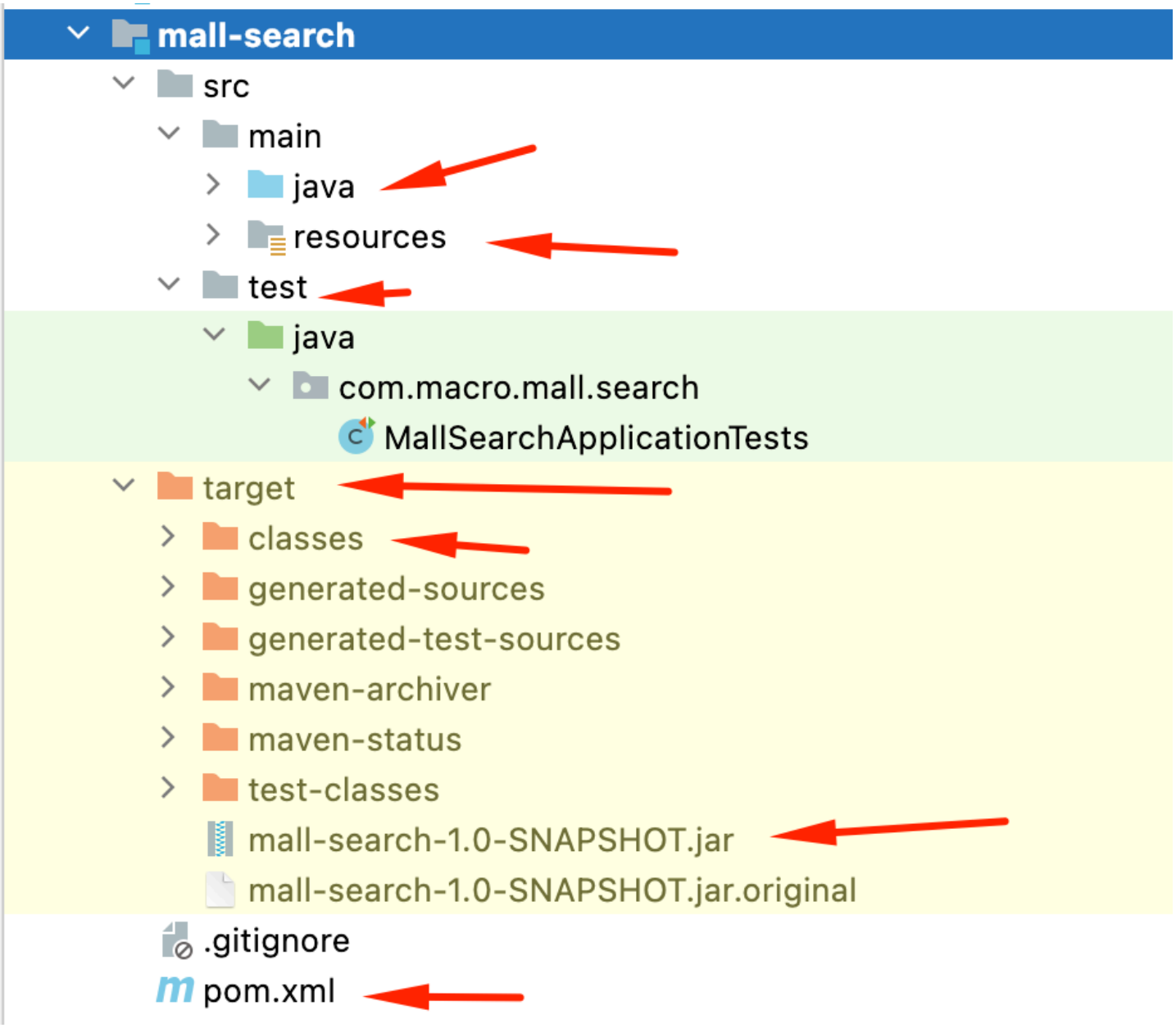
External Libraries

- < 1.8 > /Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home
- Maven: ch.qos.logback:logback-classic:1.2.3
- Maven: ch.qos.logback:logback-core:1.2.3
- Maven: cn.hutool:hutool-all:5.4.0
- Maven: com.alibaba:druid:1.1.23
- Maven: com.alibaba:druid-spring-boot-starter:1.1.23
- Maven: com.aliyun.oss:aliyun-sdk-oss:2.5.0
- Maven: com.carrotsearch.thirdparty:simple-xml-safe:2.7.1
- Maven: com.carrotsearch:hppc:0.8.1
- Maven: com.fasterxml.jackson.core:jackson-annotations:2.11.0
- Maven: com.fasterxml.jackson.core:jackson-core:2.11.0
- Maven: com.fasterxml.jackson.core:jackson-databind:2.11.0
- Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-cbor:2.11.0
- Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-smile:2.11.0
- Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-yaml:2.11.0
- Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.11.0
- Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.11.0
- Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.11.0
- Maven: com.fasterxml:classmate:1.5.1
- Maven: com.github.jsqlparser:jsqlparser:3.2
- Maven: com.github.pagehelper:pagehelper:5.2.0

Roles

- Manage **Dependencies**(Package).
- **Build** Project (Cycle)
- Documentation
- Reporting
- Others

Maven Project Architecture



Item	Default
source code	${basedir}/src/main/java
Resources	${basedir}/src/main/resources
Tests	${basedir}/src/test
Complied byte code	${basedir}/target
distributable JAR	${basedir}/target/classes

Respository & Pom.xml

Pom.xml Decomposing

Maven UIDs

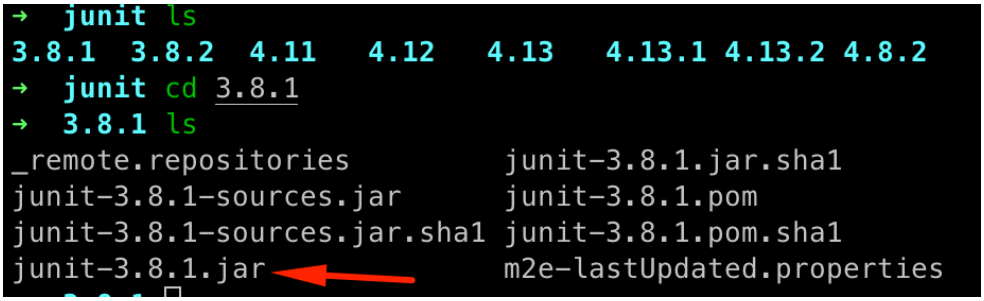
```
    <groupId>com.chuwa.mall</groupId>
    <artifactId>oms</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging> #WAR
```

produce jar file: **artifactId - version . packaging**

`oms-1.0-SNAPSHOT.jar`

Example：

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
```



Dependecy

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-aop</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
```

```
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
    </dependencies>
```

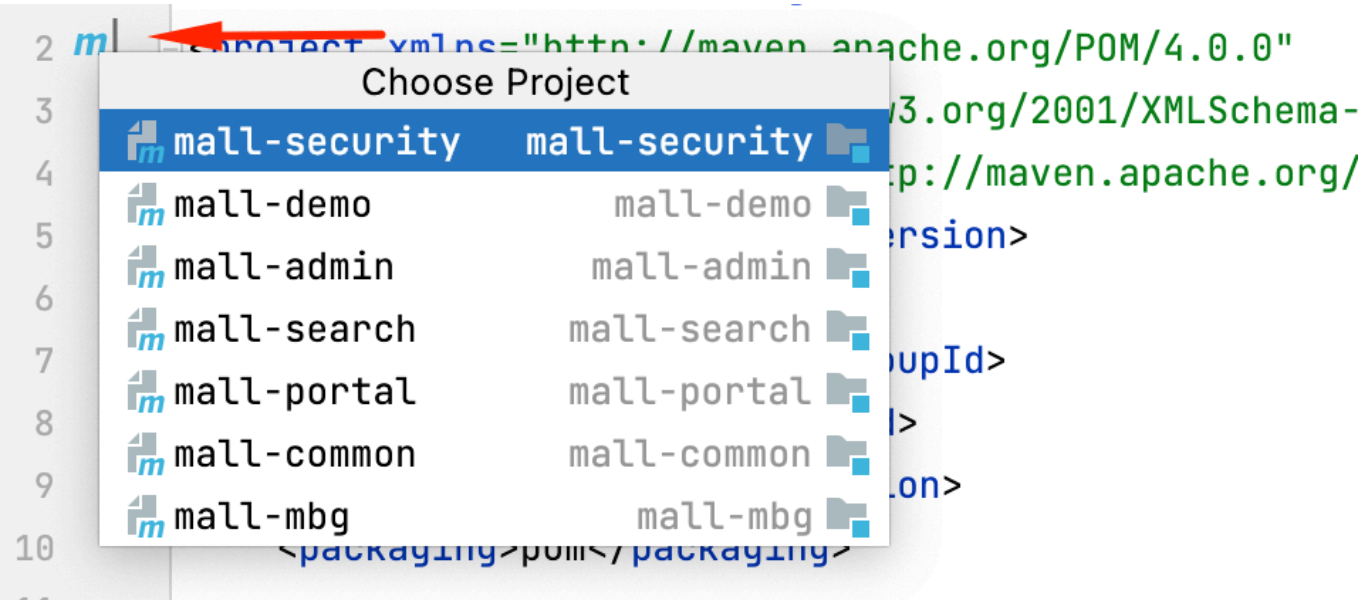
Dependecy parent-inherit

类似于Java OOP中的继承，当前pom.xml可继承来自于parent pom.xml中的配置。也可以override。

> check the parent's pom.xml

```
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.0.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
```

该pom.xml 被哪些继承了。



该pom.xml继承了谁

Properties

`${mysql-connector.version}` use the value confiured in properties.

```
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
        <mysql-connector.version>8.0.22</mysql-connector.version>
    </properties>

    <dependencies>
         <!--Mysql数据库驱动-->
         <dependency>
         <groupId>mysql</groupId>
             <artifactId>mysql-connector-java</artifactId>
             <version>${mysql-connector.version}</version>
         </dependency>

         <dependency>
         <groupId>mysql2</groupId>
             <artifactId>mysql-connector-java</artifactId>
             <version>${mysql-connector.version}</version>
         </dependency>
    </dependencies>
```

Plugins

Example

```
<build>
     <plugins>
        <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-antrun-plugin</artifactId>
           <version>1.1</version>
           <executions>
              <execution>
                 <id>id.clean</id>
                 <phase>clean</phase>
                 <goals>
                    <goal>run</goal>
                 </goals>
                 <configuration>
```

```
                    <tasks>
                       <echo>clean phase</echo>
                    </tasks>
                 </configuration>
              </execution>
           </executions>
        </plugin>
     </plugins>
  </build>
```

Commands

`mvn [plugin-name]:[goal-name]`

e.g. `mvn compiler:compile`

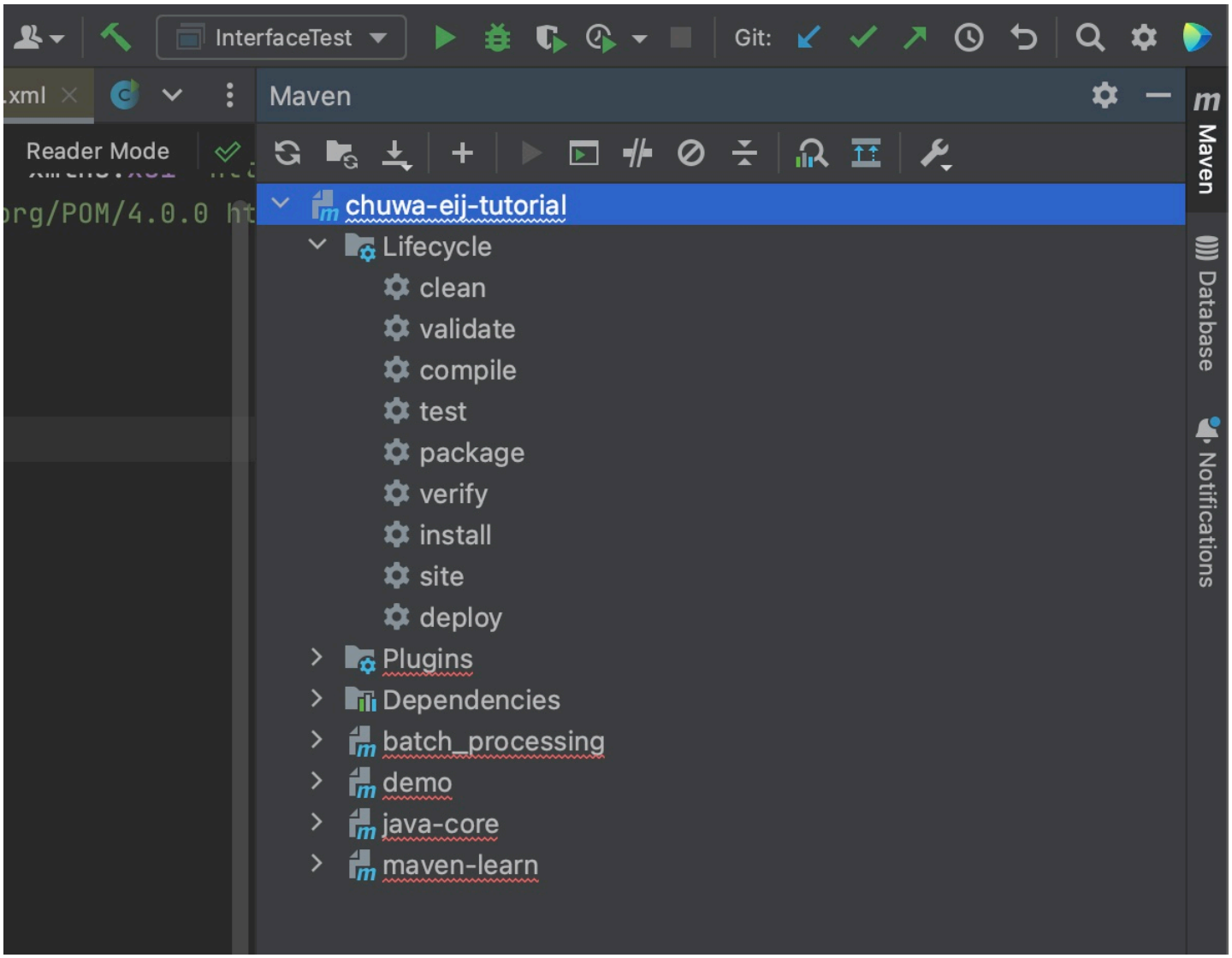
creating project in terminal:

```
$> mvn archetype:generate
-DgroupId = com.chuwa.app
-DartifactId = helloword
-DarchetypeArtifactId = maven-archetype-quickstart
-DinteractiveMode = false
```

```
mvn compile                 #Compiles source code and stores classes to target/classes
mvn validate                #Validates project
mvn test                    #Runs tests
mvn clean                   #Deletes target directory
mvn clean package           #Compiled code is packaged to WAR/JAR/deb etc
mvn clean install           #Install the artifact in local repository(.m2 Repo)
mvn clean deploy            #Copies the package(WAR/JAR/deb etc) to the remote repository
mvn verify
mvn clean verify

# Options
mvn clean install -Dmaven.test.skip=true              #Skips compiling and running tests
mvn clean install -DskipTests=true                    #Compiles but skips running tests
mvn clean install -Dmaven.test.failure.ignore=true    #Compiles and executes tests but ignores if any tests failed
mvn verify Dit.test=TestName                          #Executes specified test
mvn clean install -T 4                                # -T is used to specify number of threads used, default 2 i.e., 2 threads per CPU.
mvn clean install -X/--debug                          #Enables debug mode
```

```
mvn clean package -U/--update-snapshots                    #Force check on dependency updates
mvn dependency:purge-local-repository                      #Removes local repository

# Dependency Plugin
mvn dependency:analyze                                      #Analyzes dependencies of the project
mvn dependency:tree                                         #Prints dependency tree
mvn versions:display-dependency-updates                     #Displays dependency updates
mvn dependency:analyze -DignoreNonCompile=true              #Shows unused dependencies

# Create java project (JAR):
mvn archetype:create -DgroupId=org.onecompiler.project -DartifactId=one-compiler -DarchetypeArtifactId=maven-archetype-quickstart
```

Intellij run maven



Maven Build Life Cycle (important)

Maven Build Life Cycle (Important)

> 每次build都是从起点到制定phase，只有Test可以跳过。

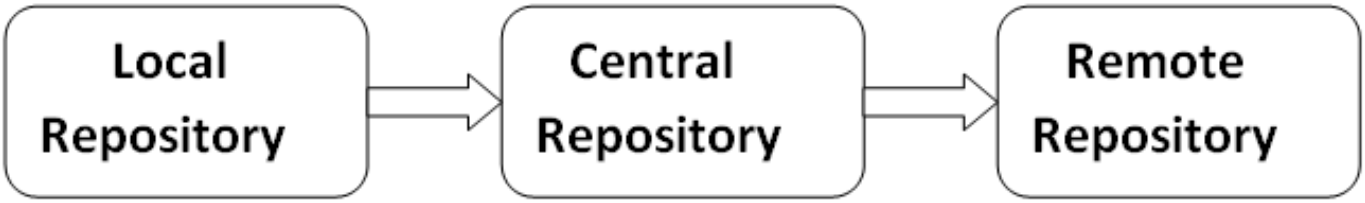
`mvn clean install -Dmaven.test.skip=true`

mvnClean -> prepare-resources -> validate -> package -> install

Phase	Handles	Description
prepare-resources	resource copying	Resource copying can be customized in this phase.
validate	Validating the information	Validates if the project is correct and if all necessary information is available.
compile	compilation	Source code compilation is done in this phase.
Test	Testing	Tests the compiled source code suitable for testing framework.
package	packaging	This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.
install	installation	This phase installs the package in local/remote maven repository.
Deploy	Deploying	Copies the final package to the remote repository.

Maven Repositories

Local repository then **Central repository** then **Remote repository**.



Maven Local Repository

Cached the dependecies in your local machine.

```
cd ~/.m2
ls
cd repository
```

```
→  .m2 ls
repository wrapper
→  .m2 cd repository
→  repository ls
antlr                      commons-fileupload       mysql
aopalliance                commons-httpclient       net
asm                        commons-io               nz
avalon-framework           commons-lang             ognl
backport-util-concurrent   commons-logging          org
biz                        commons-validator        oro
cglib                      de                       pl
ch                         dom4j                    redis
classworlds                edu                      regexp
cn                         io                       slide
com                        jakarta                  sslext
commons-beanutils          javax                    stax
commons-chain              jdom                     xerces
commons-cli                joda-time                xml-apis
commons-codec              junit                    xmlpull
commons-collections        kr                       xmlunit
commons-configuration      log4j                    xpp3
commons-digester           logkit
```

Maven Central Repository

> Maven **central repository** is located on the web. It has been created by the **apache maven community** itself.

- The path of central repository is: http://repo1.maven.org/maven2/.
- The central repository contains a lot of common libraries that can be viewed by this url http://search.maven.org/#browse.
- Get dependecies: https://mvnrepository.com/
  -

Home » junit » junit » 4.13.2

JUnit » 4.13.2

JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

License	EPL 1.0
Categories	Testing Frameworks
Organization	JUnit
HomePage	http://junit.org
Date	(Feb 13, 2021)
Files	jar (375 KB) View All
Repositories	Central Minebench Lutece Paris Xceptance
Used By	**117,509 artifacts**

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

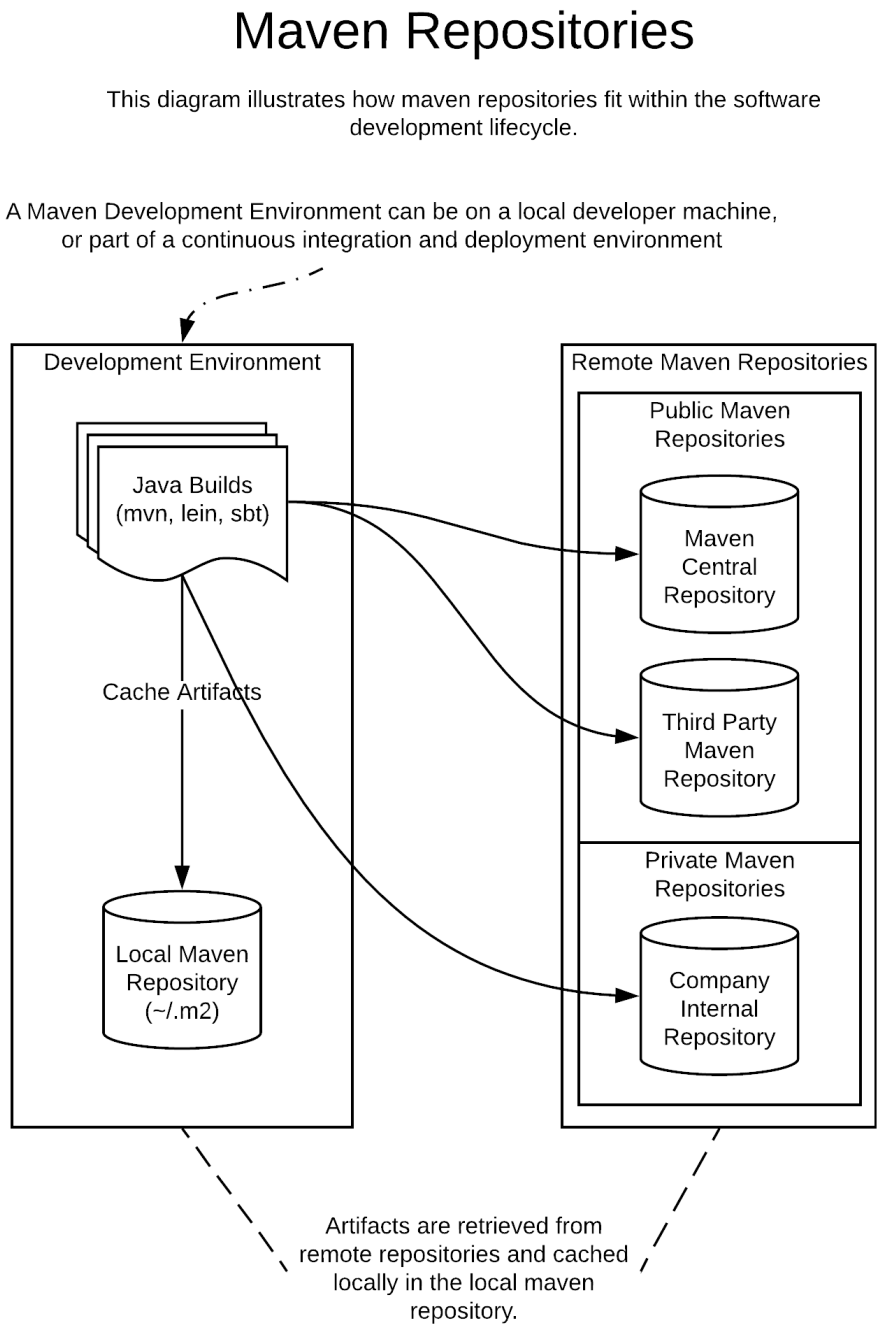
Maven Remote Repository

> Maven **remote repository** is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.
>
> Maven远程库也是位于网络上的存储库。例如一个公司可能有很多共享的jar包文件，就可以搭建一个公司内部的远程库，供众多开发人员使用；**中央库可以认为是一个特殊的远程库**。

```
<repositories>
   <repository>
      <id>Redhat.Jboss.lib2</id>
      <url>https://repository.jboss.org/</url>
   </repository>

   <repository>
      <id>maven.companyname.com.lib1</id>
      <url>http://download.companyname.org/maven2/lib1</url>
      <credential></credential>
   </repository>
</repositories>
```

Maven Dependency Search Sequence

Maven Dependency Search Sequence



When we execute Maven build commands, Maven starts looking for dependency libraries in the following sequence −

- **Step 1** − Search dependency in **local repository**, if not found, move to step 2 else perform the further processing.
- **Step 2** − Search dependency in **central repository**, if not found and remote repository/repositories is/are mentioned then move to step 4. Else it is downloaded to local repository for future reference.
- **Step 3** − If a **remote repository** has not been mentioned, Maven simply stops the processing and throws

- **Step 3** – If a **remote repository** has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency). (公司内部自己开发的lib, 其它开源平台上的dependecy, ie. Redhat)
- **Step 4** – Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference. Otherwise, Maven stops processing and throws error (Unable to find dependency).

Questions

- Which tool in **Node.js** is similiar to **Maven** ?

Git, Github

Commands

```
git init # init a git repo(repository-->source code)
git status
git add .
git commit -m "some message"
git push origin master/main

# 不commit当前的changes，但是要切换到别的branch，则stash changes,回来后再pop
# 演示IntelliJ上的操作
git stash
git stash pop

#Branches
git branch branch_name
git checkout branch_name
git checkout -b branch_name # create and checkout to branch_name

# Git Remote
git remote -v
origin  https://github.com/TAIsRich/chuwa-eij.git (fetch)
origin  https://github.com/TAIsRich/chuwa-eij.git (push)
upstream  https://github.com/B1g0/Everything-In-Java.git (fetch)
upstream  https://github.com/B1g0/Everything-In-Java.git (push)

git remote add upstream https://github.com/B1g0/Everything-In-Java.git
git remote add upstream https://github.com/B1g0/Everything-In-Java11.git
git remote add upstreamWhateverName https://github.com/B1g0/Everything-In-Java11.git

git add "localfiles.java"
git commit -m 'something'
git push
```

git branch + # branch 相关的命令

git branch + # branch 相关的命令
git remote + # remote(远程) 相关的命令

Important Concepts

`git push github-url branchName`

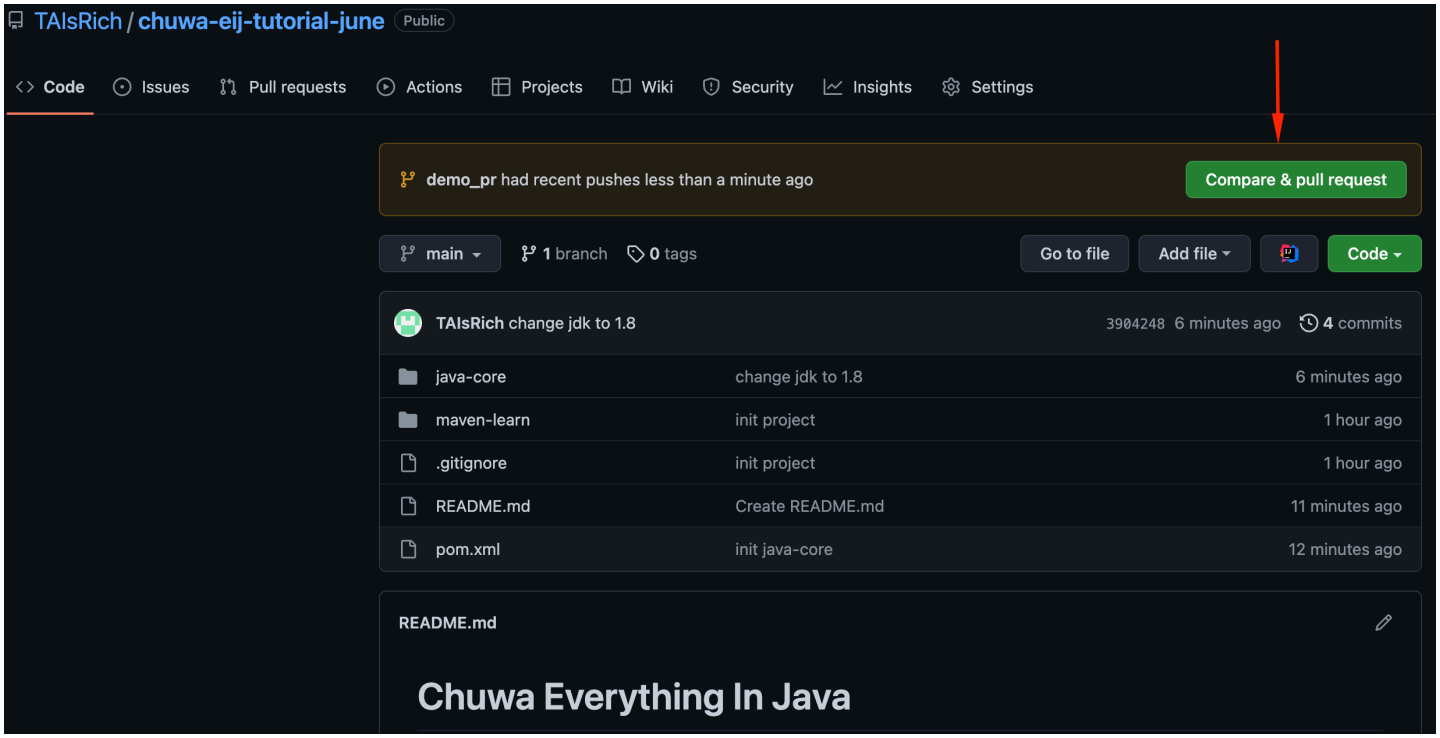
我们用**origin/upstream** 来代替很长的url

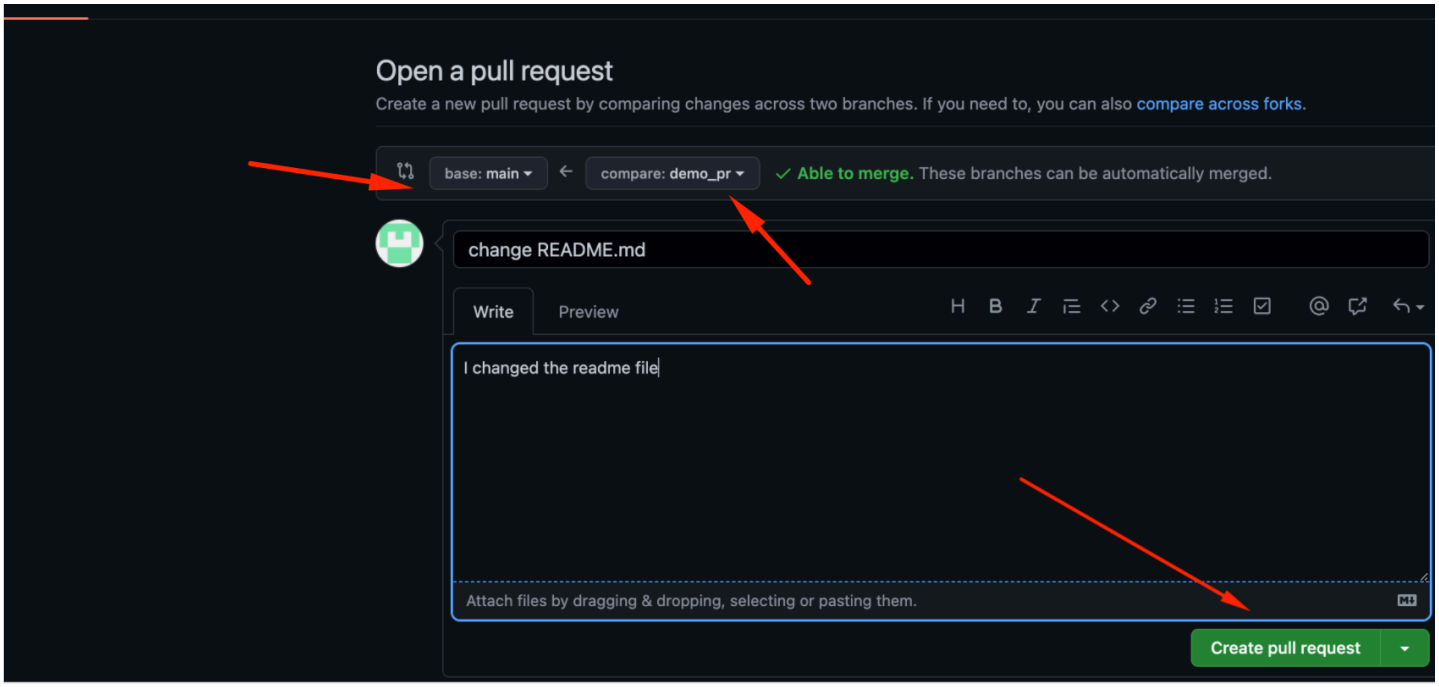
Origin -> https://github.com/TAIsRich/chuwa-eij.git

Upstream -> https://github.com/sth/else.git

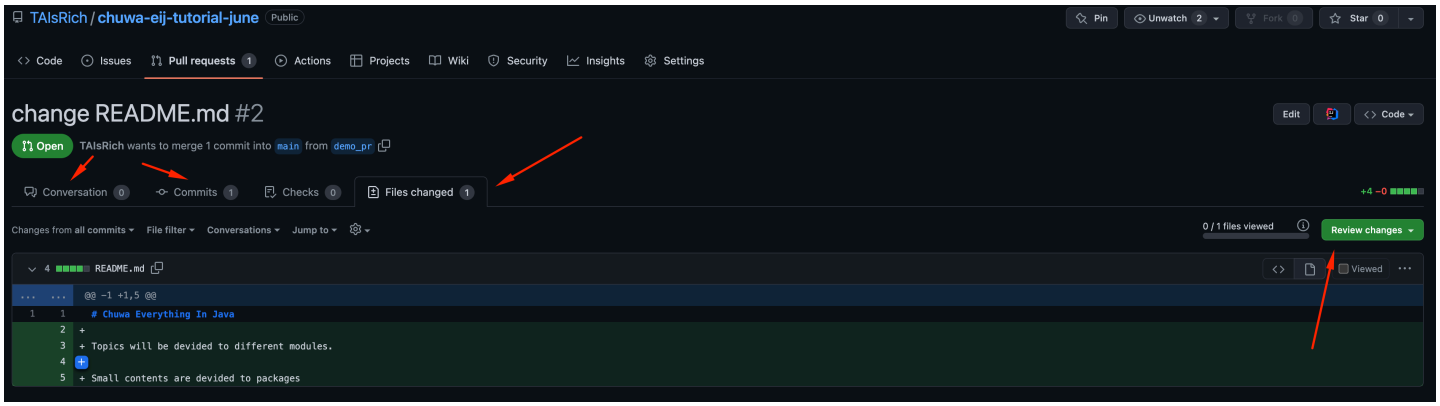
How to raise a PR?(Pull Request)

PR是基于branch的





Noticed that specify from **demo_pr** to **main(master)**



Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

change README.md #2

Open TAIsRich wants to merge 1 commit into `main` from `demo_pr`

Conversation 1 Commits 1 Checks 0 Files changed 1

TAIsRich commented 3 minutes ago Owner Tip

I changed the readme file

change README.md f631dd5

TAIsRich reviewed now View changes

TAIsRich left a comment Owner Author Tip

please add more details

Add more commits by pushing to the **demo_pr** branch on **TAIsRich/chuwa-eij-tutorial-june**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Write Preview

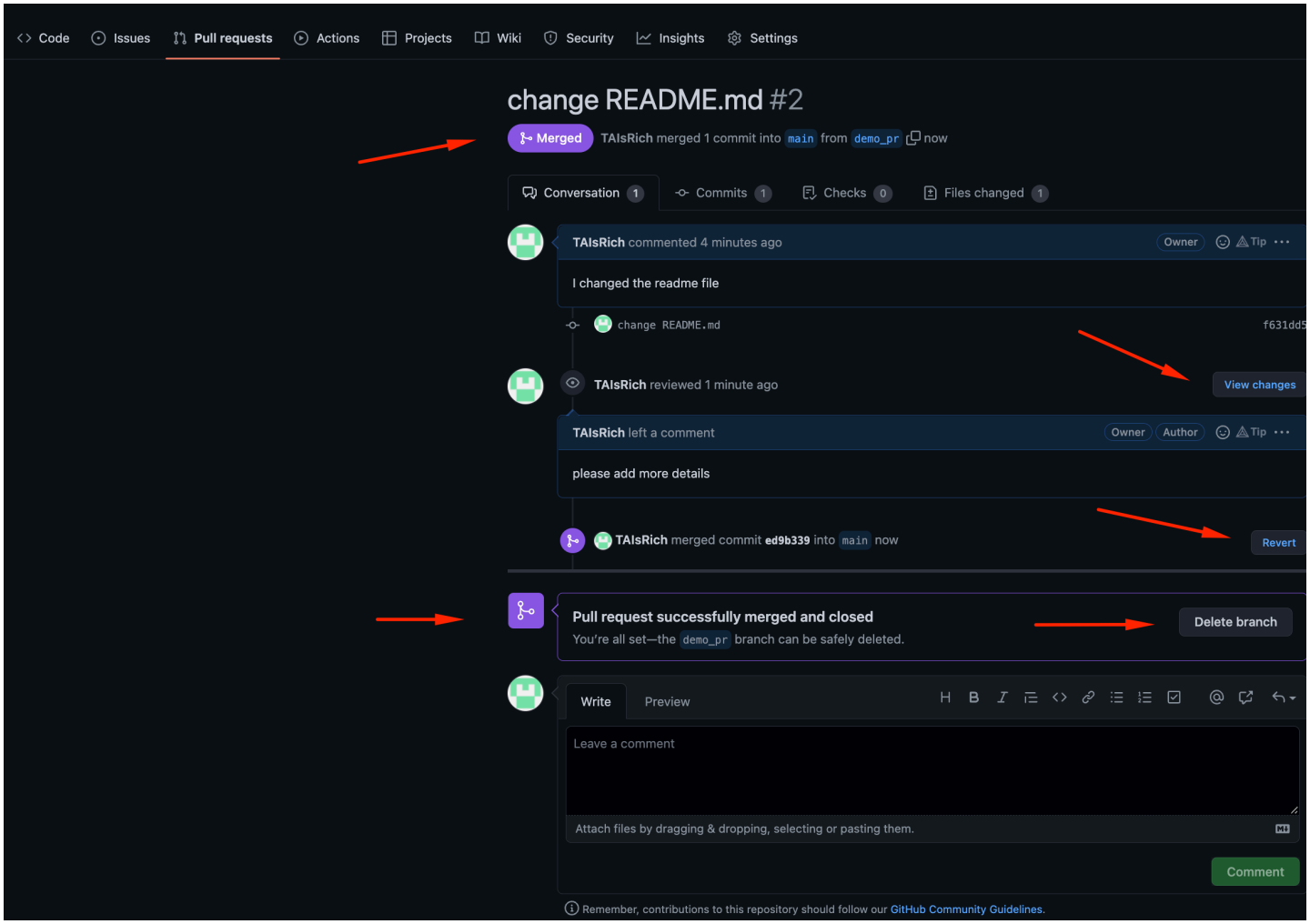
Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

TAIsRich / chuwa-eij-tutorial-june Public



`Git 的基本操作：（从git下载项目）`

1. 新建文件夹 命令是mkdir
2. 进入文件夹cd
3. 下载项目
    1. Git页面点code, 复制https链接
    2. 在下，git clone <https 链接>
4. 创建notes branch
    1. cd (比如chuwa888是名字，先ls找到名字)
    2. git branch <firstname_lastname/notes> 创建一个notes branch 在本地
    3. git checkout <firstname_lastname/notes> 切换到新创建的notes branch
    4. git push origin <firstname_lastname/notes> 上传notes branch去远程
5. 创建hw branch并修改
    1. git branch <firstname_lastnamehw1> 创建hw branch在本地
    2. git checkout <firstname_lastname/hw1> 切换到hw branch

   3. 在hw branch上写作业
6. 上传修改
   1. 打开IntelliJ 的commit tab
   2. 选上所有的changes, 右键add to VCS
   3. 打开 Terminal , git commit -m “” commit 当前的change
   4. git push origin <firstname_lastname/hw1> 把当前本地hw branch的commit 上传到远程
7. 交pr
   1. 打开git，有个 compare & pull request，点击进去
   2. Compare branch选 hw branch， base branch选 notes branch
   3. 点击create pull request

Referencies

- basic introduction: https://www.youtube.com/watch?v=KNGQ9JBQWhQ
- Maven:
  - https://www.tutorialspoint.com/maven/index.htm
  - https://www.javatpoint.com/maven-repository