

# Maven - Forge - Intégration continue

Transparents d'Emmanuel Coquery

emmanuel.coquery@univ-lyon1.fr

http://liris.cnrs.fr/~ecoquery  
→ Enseignement



## Autour du développement

Au delà du code:

- Tests (unitaires, intégration, fonctionnels)
- Documentation
- Partage des sources
- Suivi de bugs / évolutions
- Qualité du code
- Distribution

→ cycles de développement lourds à gérer



## Outils

- Frameworks de tests
- Générateurs de documentation
- Gestionnaires de version
- Gestionnaires de tickets
- Outils d'audit de code
- Scripts, *builders*



## Maven

Automatisation du traitement des phases du cycle de vie

- Peut être vu comme un “super Makefile”
  - Java comme langage de script
- Lance l'exécution d'outils:
  - Compilation
  - Test automatisés
  - Archives, Déploiement
  - Génération de documentation
  - ...

Alternatives: CMake, Premake, Grunt, Gulp, etc



## Architecture

- Basée sur un système de plugins
  - Un plugin ↔ un script Java
    - ★ i.e. une classe avec une méthode particulière
    - ★ paramétrable via un morceau de XML
  - Une exécution de maven ↔ suite d'exécution de plugins
- Nombreux plugins disponibles
  - Pas tous installés au départ
  - Système de téléchargement de plugins à la demande



## Phases et cycles de vie

- Une phase regroupe un ensemble de tâches (goals)
  - 1 tâche → 1 plugin
- Un cycle de vie est une suite de phases
  - Le déclenchement d'une phase déclenche les phases précédentes du cycle de vie
- Le cycle de vie dépend du *packaging* (jar, war, ...)
  - *packaging* = type de projet
  - Format d'archive
  - Ordre des phases
  - Affectation tâches → phases
  - Préconfiguration des tâches
  - peut être reconfiguré selon les besoins du projet



## Exemple: phases du *packaging* jar

Phase	Tâche(s)
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar
install	install:install
deploy	deploy:deploy



## Projet maven: organisation des fichiers

- pom.xml ← config. du projet
- src/ ← sources
  - main/ ← à distribuer
    - ★ java/ ← code Java
    - ★ resources/ ← fichiers à distribuer (config appli, images, etc)
    - ★ webapp/ ← ressources web (pour les war: html, jsp, js, images)
    - ★ javacc/ ← grammaire pour générer les *parsers*
    - ★ ...
  - test/ ← uniquement pour les tests
    - ★ java/, resources/, javacc/, etc
- target/ ← tout ce qui est généré, il est supprimé par `clean`  
il ne faut **pas** le versionner (utiliser `.gitignore`)



Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Projet Maven: le pom.xml

```

<project ...>
  ...
  <groupId>fr.univ-lyon1.info.ml</groupId>
  <artifactId>cv_search</artifactId>

  <dependencies>
    <dependency>...</dependency>
    <dependency>...</dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>...</plugin>
      <plugin>...</plugin>
    </plugins>
  </build>
</project>

```

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 9 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Projet Maven: dépendances dans le pom.xml

```

<project ...>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  ...
</project>

```

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 10 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Repository Maven

- Dépôt contenant:
  - Des plugins
  - Des bibliothèques (en général Java)
- Sur le web
  - Téléchargement automatique à la demande
  - Défaut: <http://repo1.maven.org>
  - Miroirs (Nexus, Archiva, etc)
- Local: `~/.m2/repository`
  - contient les archives des projets locaux
    - ★ phase `install`
  - cache pour les *repository* web

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 11 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Classpath et dépendances

Utilisation de libs externes

- Téléchargement
- Gestion des versions (⇒ build reproductible)
- Transitivité des dépendances
- Configuration du CLASSPATH

Egalement utilisé pour les plugins

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 12 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Dépendances: *scope*

	compilation	test	exécution	déploiement
compile	x	x	x	x
provided	x	x	x	
runtime		x	x	x
test		x		

(+ system, import)

Exemple: JUnit: `<scope>test</scope>` ⇒ pas dispo dans `mvn exec:java`.

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 13 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Archetypes

- Complexité inhérente aux projets maven
  - Difficultés de mise en œuvre
- Archetype = mini-projet de départ
  - D'un type particulier
  - Préconfiguré
- Exemples
  - `maven-archetype-quickstart`
  - `spring-mvc-jpa-archetype`
- En ligne de commande: `mvn archetype:generate`

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 14 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Intégration dans les EDI

Projets

- IntelliJ: par défaut
- Eclipse:
  - Plugin m2e + connecteurs
  - `mvn eclipse:eclipse`
    - ★ configure un projet Eclipse
    - ★ qui correspond au projet maven
- Netbeans: par défaut

Exécution:

- Intégrée dans tous les EDI (via plugin dans Eclipse)

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 15 / 29 >
nenti

Maven
Fiche
Maven (pom.xml)
Maven (pom.xml)
Maven (pom.xml)

## Forges logicielles

Outil de travail collaboratif pour le développement:

- Espace collaboratif
  - Partage de documents
  - Wiki
  - Dépôt (SVN, Mercurial, Git, etc)
- Gestion des tâches
  - Bug tracking
  - Support, tâches diverses
  - Calendrier, Gantt
  - Suivi via un système de tickets (Issues)

Transparents d'Emmanuel Coquery (emmanuelc)
Maven - Forge - Intégration continue
< 16 / 29 >
nenti

forge.univ-lyon1.fr

- Forge Gitlab (logiciel libre installé sur nos serveurs)
- Dépôts Git
- Intégration SI Lyon 1 (LDAP + CAS)
  - Disponible aux étudiants et personnels
  - Utilisable pour les TPs
  - Obligatoire pour le projet transversal (ex-MultiMIF)



## Git

- Gestionnaire de versions distribué
  - ▶ À la mercurial / darcs / bazaar / etc
- Utilisable
  - ▶ En ligne de commande (git)
  - ▶ Via une interface dédiée
    - \* Tortoise git, SourceTree, etc
  - ▶ Dans un EDI
    - \* Intégration Eclipse, Netbeans, IntelliJ, Emacs, etc

Conseil : apprendre la ligne de commande  
(maîtriser les concepts, les noms des commandes),  
puis choisir l'outil qui vous convient.



## Commandes de base

- Création
  - init, clone
- Fichiers
  - add, remove, mv, status
- Versions
  - commit, checkout
- Branches
  - branch, merge, rebase
- Synchronisation de dépôts
  - pull, fetch, push



## Scénario simple

- ➊ Début du travail
  - ➊ Clone d'un dépôt distant
  - ➋ Modification d'un fichier
  - ➌ Commit
    - ★ Pour l'instant, pas de modification du dépôt distant
  - ➍ Push vers le dépôt distant
- ➋ Plus tard ...
  - ➊ Pull du dépôt distant
    - ★ ou bien `fetch + rebase`
  - ➋ GOTO 1.2



## Suivi des tâches (gestionnaire de tickets)

- Les tâches ont
  - ▶ Une description
  - ▶ Un statut: fermé ou ouvert
- Les tâches peuvent avoir:
  - ▶ Une échéance
  - ▶ Une personne assignée
  - ▶ Des étiquettes
  - ▶ Une étape (milestone)
- Liens commit (hash 32fb54de)/tâche (numéro 1234):
  - ▶ Fixes #1234 dans le message de commit ⇒ ferme le ticket + lien hypertexte
  - ▶ ref 32fb54de dans les commentaires de la tâche ⇒ lien vers le commit



## Intégration Continue : l'idée

- Principe :
  - ▶ Intégrer le nouveau code au fur et à mesure qu'il est écrit
  - ▶ Garder une branche principale toujours fonctionnelle
- Outil indispensable : test automatisé à chaque `push`
- Abus de langage : « intégration continue » = « tests automatisés »



## Intégration Continue - contexte

Historiquement réservé aux projets

- de grande taille
- impliquant de nombreuses personnes
- avec des itérations courtes

Technologies actuelles → accessible sur de petits projets



## Intégration Continue - principes

- Automatisation des phases du cycle de vie
  - Compilation, test, mise à disposition de binaires
- Institutions de bonnes pratiques
  - Commit réguliers
  - La branche par défaut compile
  - ...
- Surveillance
  - Tableaux de bord, etc



[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)

# Serveurs d'IC


Permet d'exécuter régulièrement:

- Checkout
- Compilation
- Test
- Audit de code

Pour gitlab:

- peut servir de serveur d'IC;
- nécessite de gérer et de configurer des "runners" qui exécuteront les tâches (merci Manu !)

Alternatives: Jenkins, Travis, etc



Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue


< 25 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)

# SonarQube

Outil d'audit de code

- Analyse exécutée lors du cycle de vie
  - Via e.g. un goal maven
- Fournit des tableaux de bord:
  - Qualité du code
  - Couverture des tests unitaires



Transparents d'Emmanuel Coquery (emmanuelc)


Maven - Forge - Intégration continue

< 26 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)

# Merge-requests (aka pull-requests sur GitHub)

- Principe:
  - Un contributeur fait un "fork" du projet
  - Il fait un push dans une branche perso
  - Il demande à intégrer (merge) son code
- Intérêts:
  - Liste des branches à intégrer facile à voir
  - Discussion sur le code avant intégration
  - Possibilité de rejeter le mauvais code



Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 27 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)

# Tout ça ensemble?

- Début du travail:

```
git checkout -b mvc
...
git commit
git push -u origin mvc
```
- Création de la pull-request sur l'interface web
- Gitlab-CI fait passer les tests :

```
# Fichier .gitlab-ci.yml
...
junit:
  script:
    - cd cv-search/ && mvn test --batch-mode
```
- Maven s'occupe de télécharger les dépendances, tester, vérifier le style, et voilà :



Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 28 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)

# Références

- <http://maven.apache.org/>
- <https://git-scm.com/>
- <https://about.gitlab.com/>




Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 29 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)




Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 30 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)




Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 31 / 29 > nent)

[Home](#)
[Blog](#)
[Intégration continue](#)
[Merge requests](#)
[Références](#)



Transparents d'Emmanuel Coquery (emmanuelc)

Maven - Forge - Intégration continue

< 32 / 29 > nent)