Maven - Forge - Intégration continue

Transparents d'Emmanuel Coquery

emmanuel.coquery@univ-lyon1.fr http://liris.cnrs.fr/~ecoquery → Enseignement



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 1 / 29 > nent)

Outils

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



ransparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

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



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 5 / 29 > nent)

Exemple: phases du packaging jar

Phase	Tâche(s)		
process-resources	resources:resources		
•	1 CDOULCED. 1 CDOULCED		
compile	compiler:compile		
process-test-resources	resources:testResources		
test-compile	compiler:testCompile		
test	surefire:test		
package	jar:jar		
install	install:install		
deploy	deploy:deploy		



Autour du développement

Au delà du code:

- Tests (unitaires, intégration, fonctionnels)
- Documentation
- Partage des sources
- Suivi de bugs / évolutions
- Qualité du code
- Distribution
- ightarrow cycles de développement lourds à gérer



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 2 / 29 > nent

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



nsparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

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
 - $\textbf{Affectation } t \\ \hat{\textbf{a}} ches \rightarrow \textbf{phases}$
 - Préconfiguration des tâches
 - ▶ peut être reconfiguré selon les besoins du projet



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 6 / 29 > nent)

Projet maven: organisation des fichiers

- pom.xml ← config. du projet
- src/ ← sources
 - ▶ main/ ← à distribuer
 - $\star \ \, \text{java/} \leftarrow \text{code Java}$

 - * 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)



```
Projet Maven: le pom.xml
oject ...>
  <groupId>fr.univ-lyon1.info.m1
  <artifactId>cv_search</artifactId>
  <dependencies>
     <dependency>...</dependency>
     <dependency>...</dependency>
  </dependencies>
  <build>
     <plugins>
       <plugin>...</plugin>
       <plugin>...</plugin>
     </plugins>
                                                           (A)121
    /build>
ts d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue
                                                        < 9 / 29 > nent)
</project>
```

Repository Maven

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



ransparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

Dépendances: scope

Scope vs Classpath

Ocope vs Olasspalii							
		compilation	test	exécution	déploiement		
	compile	х	Х	х	х		
	provided	x	Х	x			
	runtime		Х	x	x		
	test		Х				

(+ system, import)

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



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 13 / 29 > nent)

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)



Projet Maven: dépendances dans le pom.xml

```
ct ...>
 <dependencies>
   <dependency>
     <groupId>junit
     <artifactId>junit</artifactId>
     <version>4.12
     <scope>test</scope>
   </dependency>
  </dependencies>
</project>
                                             (A)
```

Classpath et dépendances

Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

Utilisation de libs externes

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

Egalement utilisé pour les plugins



< 10 / 29 > nent

Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

/ 12 / 29 \

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
- \bullet En ligne de commande: mvn archetype:generate



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 14 / 29 > nent)

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 (emmanue Maven - Forge - Intégration continue

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)



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 17 / 29 > nent)

Commandes de base

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



fransparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

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):
 - ightharpoonup Fixes #1234 dans le message de commit \Rightarrow ferme le ticket + lien hypertexte
 - ► ref 32fb54de dans les commentaires de la tâche ⇒ lien vers le



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 21 / 29 > nent)

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



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.



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

< 18 / 29 > nent

Scénario simple

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



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

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 »



Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue

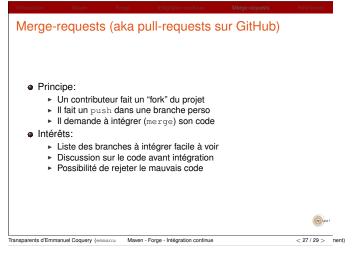
< 22 / 29 > nent)

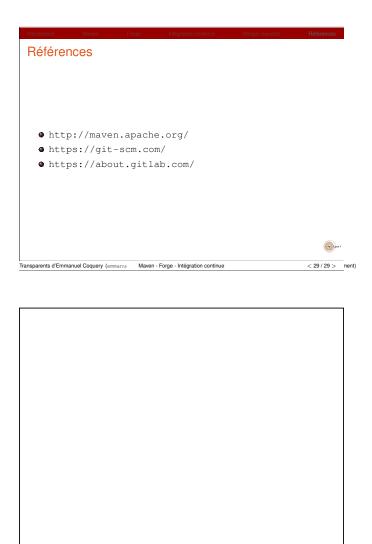
Intégration Continue - principes

- → 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
- $\to \, \text{Surveillance}$
 - Tableaux de bord, etc



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





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

(G) 1921

Transparents d'Emmanuel Coquery (emmanue Maven - Forge - Intégration continue < 26 / 29 > nent) 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 : (a) jas 1 updated 1 day ago < 28 / 29 > nent)

