ISIMA

Java EE : Construire une application Web

Build, Maven, Intégration continue !

Previously in cours de JEE

- Rappel cours précédents :
- Nous avons appris :
  - A utiliser les Servlets, Filters
  - A créer une JSP et à la lier à une Servlet
  - A dynamiser notre interface avec Ajax
  - A jouer avec les scopes proposés par JEE

- **Mais comment produire des applications de façon industrielles ?**

ISIMA

Java™

ORACLE

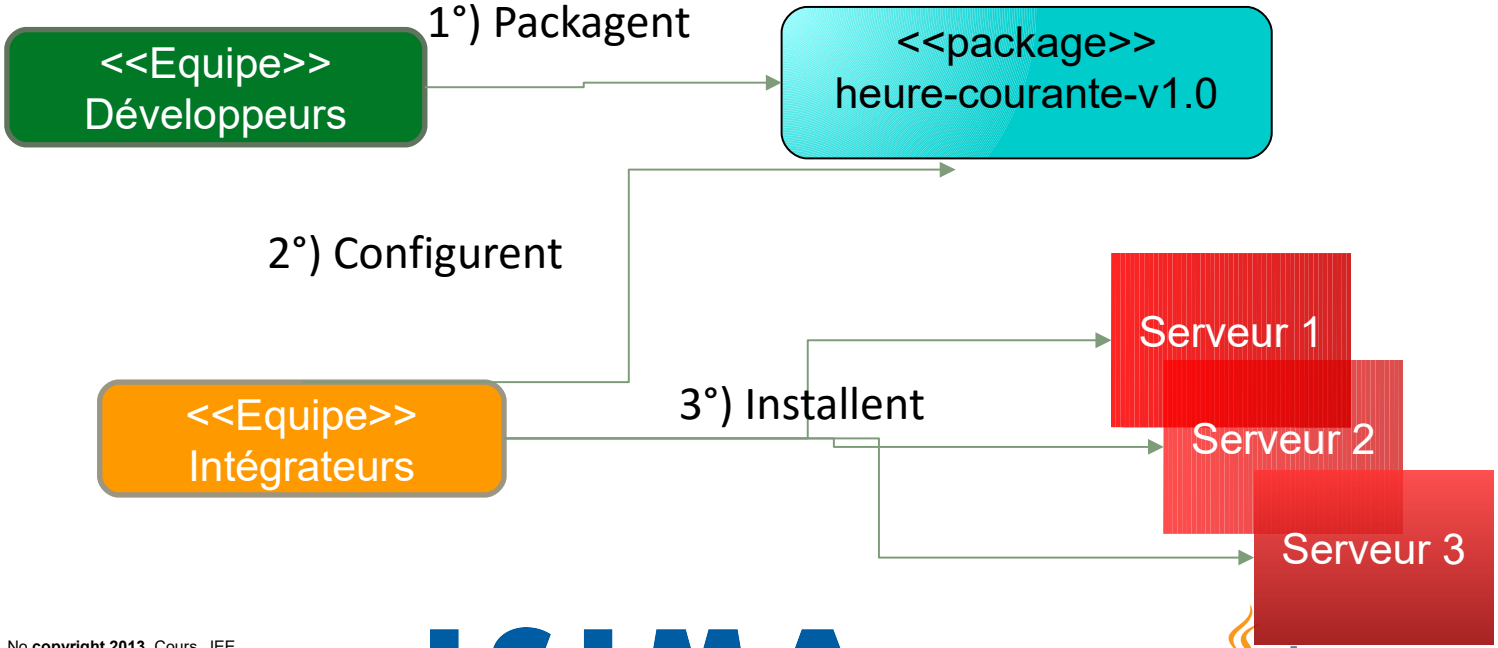
Mettre en production

- Définition
- Mettre en production consiste à publier l'application au public souhaité
  - Sur des serveurs publics ou internes à une entreprise
- Des contraintes lourdes pèsent sur une mise en production
  - respecter les données sensibles
  - s'assurer de la disponibilité
  - s'assurer de la fiabilité de la livraison

ISIMA Java ORACLE

Comment se passe la publication ?

- Voyons l'organisation d'une entreprise
- L'équipe prépare un « package » de l'application



ISIMA

Java™ ORACLE

Construire une application

- Quel package choisir ?
- Le Jar ?
  - Permet de grouper un ensemble de classes et de services
  - Possibilité de le sécuriser (signature)
  - Utilisé pour
    - Librairies
    - Programmes batchs ou clients lourds
- Fournit dans le JDK un outil jar permet de les créer
  - *Java -jar*

Construire une application

- Quel package choisir ?
- L'EAR ?
  - Permet de publier plusieurs modules Web en même temps
  - Utilisé pour packager l'ensemble des applications d'un AS par exemple
- Le War !
  - Spécialisé pour une application Web
  - Créable avec la commande Jar

Comment industrialiser?

- Sous Eclipse il est possible d'exporter manuellement un War, mais nous sommes loin de l'industrialisation
- Ce n'est certainement pas une solution pour assurer un déploiement en masse
- Comment s'assurer de la qualité du livrable proposé au client ?
- Comment s'assurer de la sauvegarde des sources ?

Usine logicielle

- La fabrication industrielle du logiciel
- Une usine logicielle permet de fabriquer des applications (web ou batchs) dans un seul point
- Elle permet de configurer les applications en fonction de l'environnement où elles vont être utilisées et installées
- Elle permet d'installer les applications
- Elle stocke les binaires générés et leurs dépendances dans des containers prévus à cet effet

Usine logicielle

- La fabrication industrielle du logiciel
- Elle assure que la qualité des logiciels soit respectée
- Elle permet de conserver la capacité à fabriquer des logiciels dans le temps (y compris après la disparition des équipes)

Deux concepts majeurs

- Intégration continue
- L'intégration continue est la capacité à construire l'application tout le temps
- Elle s'accompagne de tests
  - Des tests unitaires
  - Avec une bonne couverture du code
- La reconstruction se fait quotidiennement
- S'accompagne souvent de la gestion de projet dit « Agile »

Deux concepts majeurs

- Déploiement continu
- Les déploiements continus consistent à déployer souvent, sur beaucoup de serveurs
  - Mettre à disposition rapidement des correctifs aux utilisateurs
  - Les développeurs apprécient cette méthode avec des feedbacks rapides
- Pour déployer souvent et en sécurité, en plus des tests unitaires il faut s'appuyer sur des tests de validations automatiques
  - Robot Framework couplé à Sélénium, Fitnesse!, HQTP etc…

L’automatisation des tâches !

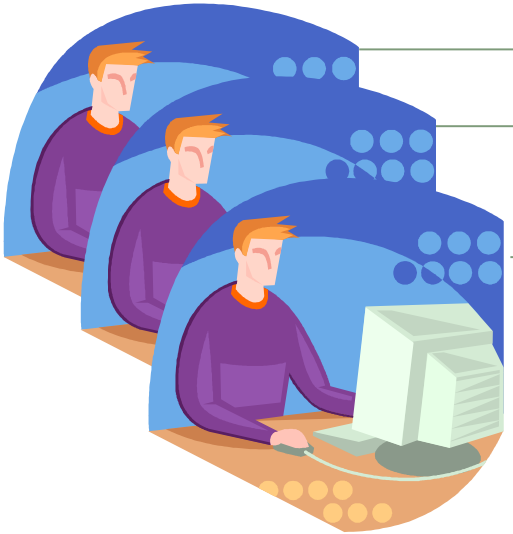
- Gagner du temps et assurer les livraisons
- Toute tâche manuelle qui ne sert pas à produire est une perte de temps
- Automatiser permet d’assurer la continuité du produit dans l’avenir, la conservation des connaissances
- Il permet de gagner du temps et de se concentrer sur le produit et non plus sur les tâches annexes

Que trouve-t-on dans une usine ?

- Une usine logicielle possède de nombreux composants
- Il existe toutefois des contraintes à respecter pour pouvoir intégrer facilement nos projets dans ce monde
  - Avoir un révérenciel de source où stocker l'ensemble du code source généré
  - Avoir un révérenciel de binaries où stocker nos projets construits
  - Avoir un chef d'orchestre pour gérer tout ce petit monde – un serveur d'intégration continue
  - Enfin les projets doivent utiliser un système de construction industrialisé

I La gestion de configuration

- Un point central pour le partage du code source



Publie

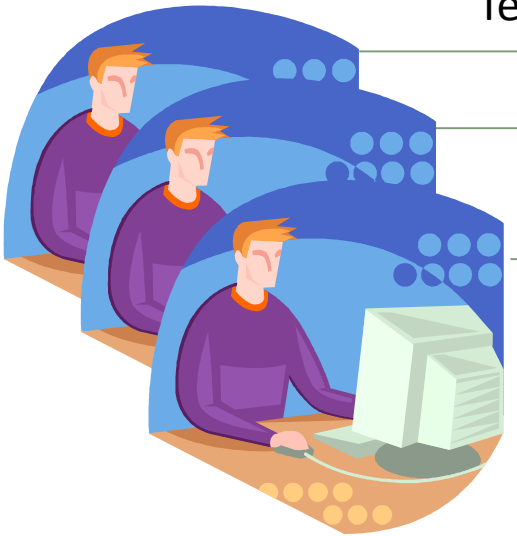
Serveur de gestion de conf

Développeurs

ISIMA

Java™ ORACLE

II le serveur de stockage des binaries



Télécharge les binaires

Développeurs

Repository

Nexus

artifactory

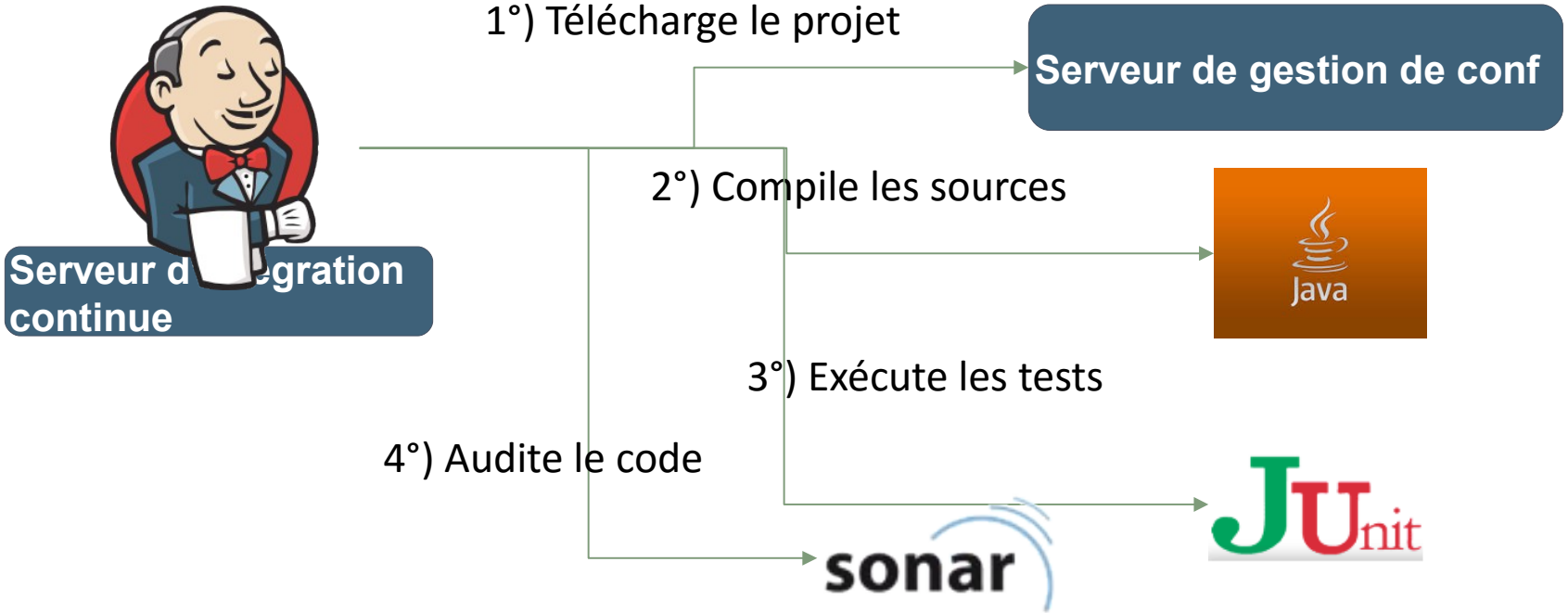
ISIMA

Java™

ORACLE

Le serveur d'intégration continue

- Que fait il (1) ?

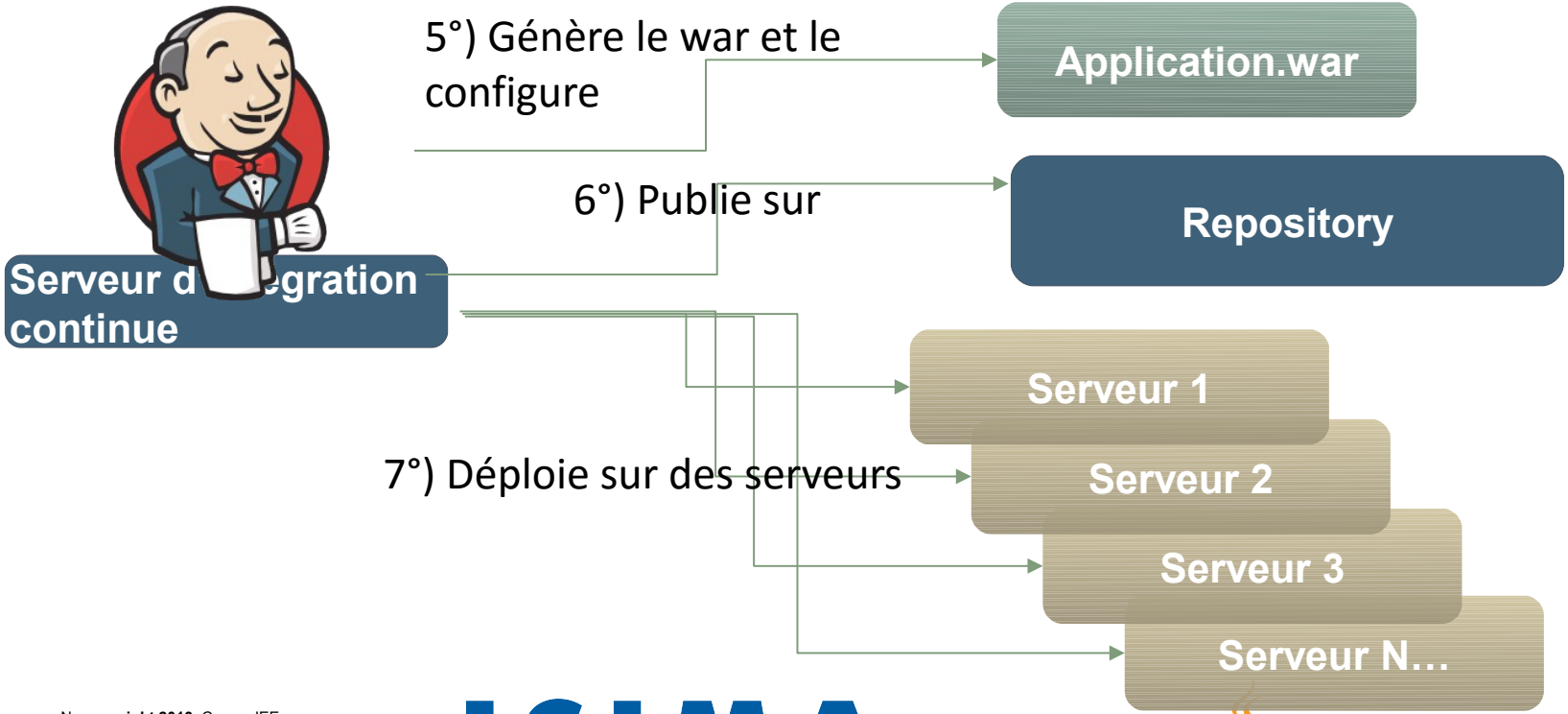


ISIMA

Java™ ORACLE

Le serveur d'intégration continue

- Que fait il (2) ?



ISIMA

Java™ ORACLE

Préparer nos projets à intégrer l'usine

- Méthode de construction
- Apache Ant est une méthode de construction avec des scripts xml qui permet de décrire la façon dont le projet doit être construit
- Toutefois cette méthode est fastidieuse et difficile à normaliser sur un SI important

ISIMA Java ORACLE

Préparer nos projets à intégrer l'usine

- Méthode de construction
- Maven joue sur la norme plutôt que sur la description
- Il fournit un ensemble d'outils permettant automatiquement de :
  - Compiler
  - Tester
  - Construire une archive
- Il apporte également une gestion des dépendances extrêmement intéressante

ISIMA

Java™

ORACLE

La notion de « goal »

- « Le build lifecycle »
- Maven fonctionne par « phases »
- Ces phases correspondent à une étape importante de la construction du projet
- Pour pouvoir atteindre une phase il faut exécuter maven en ciblant un « goal ». Toutes les phases permettant d'arriver jusqu'à ce « goal », sont exécutées

Les goals intéressants

- **compile**
  - Compile le code source
- **test**
  - Lance les tests unitaires avec TestNG ou Junit par exemple
- **package**
  - génère le war, jar ou EAR
- **install**
  - installe le jar (le copie localement)
- **deploy**
  - déploie sur l'environnement cible le binary généré (c'est souvent un serveur de binary)

Les goals intéressants

- Il faut également souvent combiner avec l'instruction clean pour être certain que le build maven soit correct
  - Par exemple un « clean install » ou un « clean deploy »

La gestion des dépendances

- Une autre fonction majeure
- Maven gère les dépendances en dehors du projet
  - Les binaires ne sont plus dans un répertoire du projet ce qui allège considérablement le poids
- Une dépendance, c'est une libraire et un numéro de version. Les dépendances sont recherchées dans les repository
- La notion de version est **CAPITALE**. Cette gestion doit être rigoureuse et correctement incrémentée

La notion de repository

- Comment s'organise Maven
- La notion de repository maven est très importante. Tous les binaries produits et toutes les dépendances sont stockés dans des repositorys
- Au passage les serveurs de stockages type Nexus ou Artifactory sont des repositories

Les différents niveaux de repository

- Le repository local
  - Il regroupe toutes les archives utilisées par les projets
  - Il contient également toutes les archives installées avec « maven install »
- Le repository d'entreprise
  - Il regroupe toutes les ressources produites par l'entreprise
  - Toutes les archives utilisées par l'ensemble des développeurs sur l'ensemble des projets

Les différents niveaux de repository

- Les repositorys externes ou centraux
  - Ils permettent de retrouver les jars produits par des tiers et il y'en a beaucoup
  - http://search.maven.org/
  - http://mvnrepository.com/
- Pour notre cours et nos TP nous n'aurons pas de repository d'entreprise

Migrons le projet heure courante

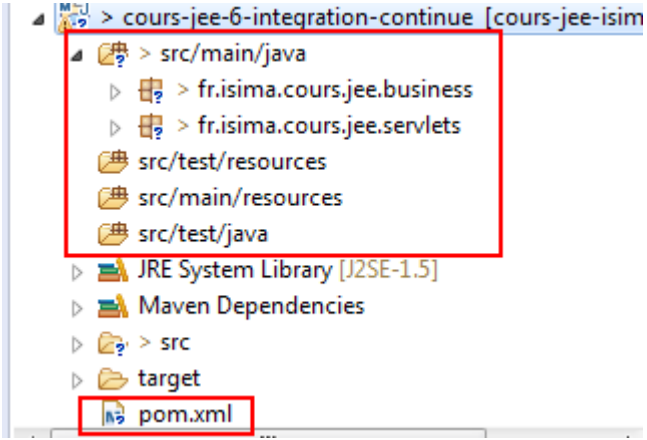
- Installer maven localement
- Pour pouvoir migrer le projet heure-courante à maven, il faut tout d'abord installer l'outil localement
- Je vous propose une documentation d'installation de maven que vous devrez suivre en préalable du TP

Création du projet Maven

- Les étapes de création du projet Maven sont détaillées dans la documentation d'installation
- Nous allons passer directement au contenu du projet Maven nouvellement créé

Le projet Maven migré

- Découvrons de quoi est fait notre projet maven.
- Les sources sont exactement identiques à celle d'heure courante du cours 5
- Toutefois la structure a changé :



L’organisation des sources

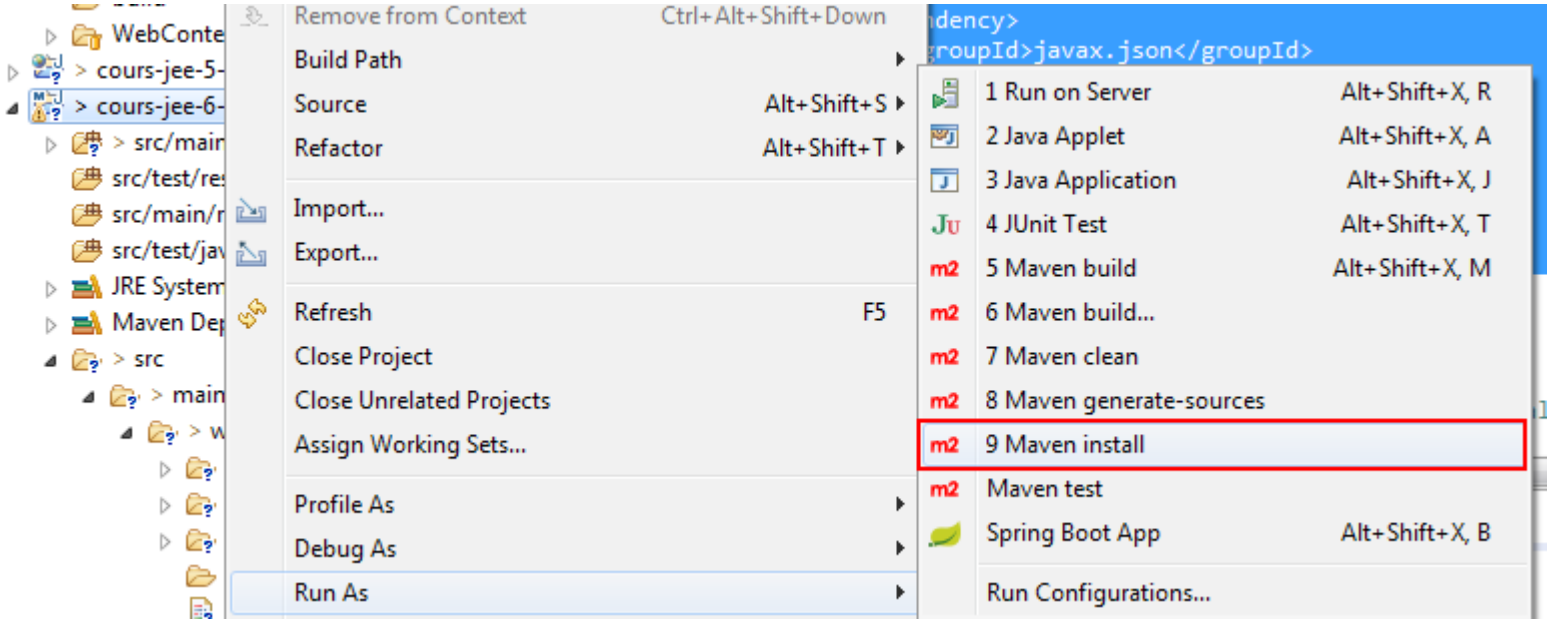
- 4 répertoires
- **src/main/java**
  - Contient toutes les sources
- **src/main/resources**
  - Contient toutes les ressources
    - Fichiers de configuration, images, sons, videos etc…
- **src/test/java** et **src/test/resources**
  - Ce sont des répertoires réservés aux tests unitaires

ISIMA

Java™

ORACLE

Exécutons le build du projet



Build réussi

```
[INFO] Webapp assembled in [531 msecs]
[INFO] Building war: C:\Users\Benjamin\Documents\cours\onGitHub\cours-jee-isima-2014\sources\cours\cours-jee-6-integration-continue\target\cours-jee-6-integration-continue.war
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ cours-jee-6-integration-continue ---
[INFO] Installing C:\Users\Benjamin\Documents\cours\onGitHub\cours-jee-isima-2014\sources\cours\cours-jee-6-integration-continue\target\cours-jee-6-integration-continue.war to C:\Users\Benjamin\.m2\repository\fr\cours\isima\cours-jee-6-integration-continue\1.0-SNAPSHOT\cours-jee-6-integration-continue-1.0-SNAPSHOT.war
[INFO] Installing C:\Users\Benjamin\Documents\cours\onGitHub\cours-jee-isima-2014\sources\cours\cours-jee-6-integration-continue\pom.xml to C:\Users\Benjamin\.m2\repository\fr\cours\isima\cours-jee-6-integration-continue\1.0-SNAPSHOT\cours-jee-6-integration-continue-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.501 s
[INFO] Finished at: 2015-01-23T23:37:06+01:00
[INFO] Final Memory: 11M/114M
[INFO] ------------------------------------
```

- Il précise le chemin dans lequel est publié le war :
  - Le repository local

Le fichier pom.xml

- *Project Object Model*
- C'est le fichier de description des applications
- Il permet de préciser les différents paramètres à appliquer
- Il permet également de décrire les dépendances du projet qui sont téléchargées
- Vous pourrez voir, par l'exemple publié du cours et par le tutoriel dans la documentation d'installation, des usages concrets

Quelques propriétés intéressantes

```
<groupId>fr.cours.isima</groupId>
```

- Le groupId désigne un ensemble dans lequel s'inscrit le projet (souvent le nom de l'entreprise ou du contributeur apparait)
- Les noms de packages du projet doivent faire apparaitre en préfixe le group-id

```
<artifactId>cours-jee-6-integration-continue</artifactId>
```

- L'artifactId correspond au nom du projet

ISIMA

Java™

ORACLE

Quelques propriétés intéressantes

```
<packaging>war</packaging>
```

- Désigne le package que maven doit produire. Généralement war (appli-web) ou jar.

```
<version>1.0-SNAPSHOT</version>
```

- La version est un point très intéressant, et il faut être sensible à la numérotation. Globalement, il y'a deux politiques de numérotation :
  - « la snapshot », les développeurs commit la snapshot jusqu'à ce qu'elle soit considérée comme prête
  - L'autre politique consiste à incrémenter à chaque commit, une livraison de code d'un développeur, en considérant que chaque version peut potentiellement partir en production

Quelques propriétés intéressantes

- Analysons maintenant une dépendance

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

- Notez que le groupId et l'artifactId sont les deux paramètres que l'on retrouve dans le pom.xml
- Ce n'est pas un hasard puisque ce sont ces références qui permettent de retrouver un projet que vous pourriez publier

Quelques propriétés intéressantes

- Passons le numéro de version dont le rôle est évident
- La notion de scope est une autre des forces de maven. En effet il est possible de considérer une dépendance uniquement dans des phases limitées
- Voici des exemples de scope :
  - Compile
  - Provided/system
  - Runtime
  - Test
  - Import (très particulier pour des pom.xml de type « pom », qui ne sont pas des projets à proprement parlé)

Conclusion

- Maven
- Permet de construire un projet facilement et dans des environnements variés
- S'inscrit dans un cadre plus large que l'intégration continue et le déploiement continu