SOMMAIRE

I. PRESENTATION

Apache Maven (couramment appelé Maven) est un outil de gestion et d'automatisation de production des projets logiciels Java en général et Java EE en particulier. Il est utilisé pour automatiser l'intégration continue lors d'un développement de logiciel. Maven est géré par l'organisation Apache Software Foundation. L'outil était précédemment une branche de l'organisation Jakarta Project.

L'objectif recherché est de produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

II. OBJECTIF DE MAVEN

L'objectif principal de Maven est de permettre à un développeur de comprendre l'état complet d'un effort de développement dans les plus brefs délais. Afin d'atteindre cet objectif, Maven tente de traiter plusieurs domaines de préoccupation:

1. Rendre le processus de construction facile

Bien que l'utilisation de Maven n'élimine pas la nécessité de connaître les mécanismes sous-jacents, Maven fournit beaucoup de protection contre les détails.

2. Fournir un système de construction uniforme

Maven permet à un projet de se construire en utilisant son modèle d'objet de projet (POM) et un ensemble de plugins qui sont partagés par tous les projets utilisant Maven, fournissant un système de construction uniforme. Une fois que vous vous êtes familiarisé avec la façon dont un projet Maven se construit, vous savez automatiquement comment tous les projets Maven se construisent, ce qui vous fait gagner énormément de temps lorsque vous essayez de naviguer dans de nombreux projets.

3. Fournir des informations de qualité sur les projets

Maven fournit de nombreuses informations utiles sur le projet qui sont en partie extraites de votre POM et en partie générées à partir des sources de votre projet. Par exemple, Maven peut fournir:

- Document de journal des modifications créé directement à partir du contrôle de code source

- Sources croisées
- Liste des listes de diffusion gérées par le projet
- Liste des dépendances
- Rapports de tests unitaires, y compris la couverture

Au fur et à mesure que Maven s'améliore, l'ensemble des informations fournies s'améliorera, ce qui sera transparent pour les utilisateurs de Maven.

D'autres produits peuvent également fournir des plugins Maven pour permettre leur ensemble d'informations sur le projet en plus de certaines des informations standard fournies par Maven, toutes toujours basées sur le POM.

4. ***Fournir des directives pour le développement des meilleures pratiques***

Maven vise à rassembler les principes actuels pour le développement des meilleures pratiques et à faciliter l'orientation d'un projet dans cette direction.

Par exemple, la spécification, l'exécution et la génération de rapports sur les tests unitaires font partie du cycle de construction normal à l'aide de Maven. Les meilleures pratiques actuelles de tests unitaires ont été utilisées comme lignes directrices:

- Conserver le code source du test dans une arborescence source distincte mais parallèle
- Utilisation des conventions de dénomination des cas de test pour localiser et exécuter des tests
- Ayant les cas de test configuré leur environnement au lieu de compter sur la personnalisation de la construction pour la préparation des tests

Maven vise également à aider au flux de travail du projet, comme la gestion des versions et des problèmes.

Maven suggère également quelques directives sur la façon de mettre en page la structure de répertoire de votre projet. Une fois que vous avez appris la disposition, vous pouvez facilement naviguer dans n'importe quel autre projet qui utilise Maven et les mêmes valeurs par défaut.

5. ***Permettre une migration transparente vers de nouvelles fonctionnalités***

Maven offre aux clients Maven un moyen simple de mettre à jour leurs installations afin qu'ils puissent profiter des modifications apportées à Maven lui-même.

L'installation de plugins nouveaux ou mis à jour de tiers ou de Maven lui-même est devenue triviale pour cette raison.

III. CONVENTION ET CYCLE DE VIE

1. *Convention*

Maven impose une arborescence et un nommage des fichiers du projet selon le concept de Convention plutôt que configuration. Ces conventions permettent de réduire la configuration des projets, tant qu'un projet suit les conventions. Si un projet a besoin de s'écarter de la convention, le développeur le précise dans la configuration du projet.

Voici une liste non exhaustive des répertoires d'un projet Maven :

- `/src` : les sources du projet
- `/src/main` : code source et fichiers source principaux
- `/src/main/java` : code source
- `/src/main/resources` : fichiers de ressources (images, fichiers annexes, etc.)
- `/src/main/webapp` : webapp du projet
- `/src/test` : fichiers de test
- `/src/test/java` : code source de test
- `/src/test/resources` : fichiers de ressources de test
- `/src/site` : informations sur le projet et/ou les rapports générés suite aux traitements effectués
- `/target` : fichiers résultat, les binaires (du code et des tests), les packages générés et les résultats des tests

2. *Cycle de vie*

Les principales finalités du cycle de vie d'un projet Maven sont:

- `compile`
- `test`
- `package`
- `install`
- `deploy`

L'idée est que, pour n'importe quel but, tous les buts en amont doivent être exécutés sauf s'ils ont déjà été exécutés avec succès et qu'aucun changement n'a été

fait dans le projet depuis. Par exemple, quand on exécute `mvn install`, Maven va vérifier que `mvn package` s'est terminé avec succès (le jar existe dans `target/`), auquel cas cela ne sera pas ré-exécuté.

D'autres buts sont exécutables en dehors du cycle de vie et ne font pas partie du cycle de vie par défaut de Maven (ils ne sont pas indispensables). Voici les principaux :

- `clean`
- `assembly:assembly`
- `site`
- `site-deploy`
- etc.

Ils peuvent néanmoins être rajoutés au cycle de vie via le POM.

IV. CONCLUSION

En conclusion, Maven est devenu l'outil de build incontournable. Il est de plus en plus utilisé dans le monde de l'entreprise et il permet également de définir un projet avec des conventions.