GRADLE

GRADLE

1. Definition du GRADLE

1.1 Introduction

ANT et Maven ont partagé un succès considérable sur le marché Java. ANT était le premier outil de construction publié en 2000 et il est développé sur la base de l'idée de la programmation procédurale. Plus tard, il a été amélioré avec une capacité à accepter des plug-ins et la gestion de la dépendance sur le réseau à l'aide sur Apache-Ivy. Le principal inconvénient était XML comme format pour écrire créer des scripts. XML étant hiérarchique est pas bon pour la programmation procédurale et tend à devenir ingérable grand.

Maven a été introduit en 2004. Il est livré avec beaucoup d'amélioration que ANT. Il change sa structure et il continue en utilisant XML pour les spécifications de construction par écrit. Maven repose sur les conventions et est en mesure de télécharger les dépendances sur le réseau. Le principal avantage de Maven est son cycle de vie. En suivant le même cycle de vie pour de multiples projets en continu, cela vient un coût de flexibilité. Maven est également confronté à des problèmes dans la gestion des dépendances. Il ne gère pas bien les conflits entre les versions de la même bibliothèque, et complexes personnalisé scripts de construction sont en fait plus difficile à écrire dans Maven que dans ANT.

Enfin, Gradle est entré en scène en 2012. Gradle porte certaines fonctionnalités efficaces à la fois les outils.

La réalisation d'une application est constituée de nombreuses étapes. Parmi celles-ci, l'étape de construction, aussi appelée étape de build, est la plus importante. Elle permet de transformer un code source, issu d'une représentation humaine, dans un ensemble de code exécutable constituant une représentation machine. Il n'est pas rare que cette étape soit complexe tant par sa mise en oeuvre que son adéquation aux besoins changeants du projet. Il est alors nécessaire d'outiller cette construction afin de répondre au plus près aux besoins du projet.

De nombreux outils de build existent pour répondre à cette problématique. Certains de ces outils, implantés depuis des années, dominent le marché comme Ant et Maven. Néanmoins, malgré leur omniprésence, nombreux sont les développeurs qui restent bloqués et souffrent d'un manque de fonctionnalités. Gradle, né de l'expérience des utilisateurs acquise au cours ces différentes années, tente de répondre aux besoins particuliers de build des applications d'une entreprise.

Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala,Groovy .

Gradle allie les atouts de Apache Maven et Apache Ant : il allie l'utilisation de conventions à la manière de Maven (convention plutôt que configuration) avec la flexibilité de Ant pour décrire les tâches de constructions, avec une cohérence forte dans l'interface de programmation des tâches.

1.2 Un system de build de troisieme génération.

Gradle est un système de build de troisième génération proposant une approche flexible pour la construction de projets Java, Groovy et Scala et Java EE. Il utilise les meilleures fonctionnalités de chacun des outils existants comme une structure de développement cadrée, une ossature pour la construction de son projet, des conventions de noms, tout en proposant en permanence une personnalisation selon ses

besoins. En termes d'architecture technique, Gradle est un toolkit fondé sur les meilleures librairies du marché de l'intégration continue, comme le gestionnaire de dépendances Ivy, l'utilisation native de taches Ant et le langage Groovy pour décrire le build d'un projet.

A l'image de ses concurrents, Gradle apporte une approche déclarative dans la définition de la chaîne de build. Mais Gradle fournit un langage de spécification du domaine (DSL) en langage Groovy pour écrire un script Gradle. Cette DSL orchestre une API Gradle, et permet de fournir à l'écriture du build une grande richesse des éléments. Chaque script Gradle configure de manière sous-jacente un objet Gradle Projet sûr lequel le script utilise un ensemble de propriétés et de méthodes exposées par l'API.

Ce véritable langage de build vous offre une flexibilité d'expression répondant aux besoins de votre entreprise, comme la possibilité de pouvoir déclarer plusieurs répertoires de sources par projet, de configurer les dépendances très finement, de mettre en place plusieurs projets par répertoires de sources ou encore de générer autant d'artefacts que nécessaire pour votre projet.

1.3 Gradle permet de résoudre les problèmes rencontrés aven Ant ou Maven.

Au-delà d'être un compétiteur face aux outils de build existants, Gradle fait office de fédérateur. En effet, l'outil peut s'insérer aisément dans des infrastructures utilisant Ant ou Maven, ceci afin d'enrichir les fonctionnalités existantes, et offre une panoplie de stratégies pour résoudre les problèmes rencontrés avec Ant ou Maven.

Dans le cadre d'Ant, Gradle possède un module d'import permettant de convertir toutes les cibles d'un script Ant en tâches Gradle. Celui-ci peut ensuite rajouter du comportement avant ou après l'exécution de la cible Ant. Cette fusion native entre Ant et Gradle permet également de faire référence à une tache Gradle depuis un script Ant importé. Cette mutualisation donne la possibilité ainsi d'utiliser Gradle comme outil complémentaire à votre infrastructure Ant sans un besoin de migration.

Dans le cadre de Maven, Gradle est capable de collecter des artefacts dans des dépôts Maven à travers la librairie Ivy et d'écrire dans des dépôts Maven via l'utilisation native du module Maven inclus dans la distribution Gradle. Cette fonctionnalité permet d'utiliser Gradle pour ses chaines de build, et d'utiliser des dépôts d'entreprise de type Maven en standard dans la majorité des organisations. Ces dépôts, gérés par des outils gestionnaires comme JFrog Artifactory, rendent possible une centralisation du stockage des librairies utilitaires publiques issues de dépôts Internet avec les artefacts issus des projets, le tout avec des métadonnées de type uniforme.

Gradle augmente également les aspects de fiabilité et de performance d'une chaine de compilation en fournissant un support extrêmement puissant, pour un projet multi module, à travers des fonctionnalités de build incrémentales que ses compétiteurs ne possèdent pas.

Après deux ans d'existence, Gradle atteint un niveau de maturité lui permettant d'être adopté en masse et sans risques par les grands comptes. A ce jour, les entreprises ayant utilisé Gradle ont constaté une plus value immédiate, et l'ont implanté comme leur principal outil de build.

2 Caractéristiques de Gradle

Voici la liste des fonctionnalités qui Gradle fournit.

- **Déclarative construit et construit par convention - Gradle est disponible avec la langue séparée de domaine spécifique (DSL) fondée sur la langue Groovy.** Gradle fournit des éléments de langage déclaratif. Les éléments apportent également un soutien build par convention pour Java, Groovy, OSGi, Web et Scala.

- **Langue de la dépendance programmation basée - Le langage déclaratif se trouve au sommet d'un graphe de tâches à usage général, que vous pouvez entièrement l' effet de levier dans votre build.**
- **Structurez votre build - Gradle vous permet d'appliquer les principes de conception communs à votre construction.** Il vous donne une structure parfaite pour la construction, de sorte que vous pouvez concevoir bien structuré et facile à entretenir, construire compréhensible.
- **API profonde - L' utilisation de cette API, vous pouvez contrôler et personnaliser sa configuration et le comportement d'exécution à son noyau.**
- **Balances Gradle - Gradle peut facilement augmenter la productivité, du projet simple et unique construit à grande entreprise multi-projet construit.**
- **Multi-projet construit - Gradle supporte le multi-projet construit et construit aussi partielle.** Si vous construisez un sous-projet, Gradle prend soin de construire tous les sous-projets dont il dépend.
- **Différentes façons de gérer votre construit - Gradle soutient différentes stratégies pour gérer vos dépendances.**
- **Premier outil de l' intégration de la construction - Gradle soutient complètement les tâches ANT, Maven et Lvy infrastructure référentiel pour l' édition et dépendances récupération.** Il fournit également un convertisseur pour transformer un pom.xml Maven pour le script Gradle.
- **Facilité de la migration - Gradle peut facilement adapter à toute structure que vous avez.** Par conséquent, vous pouvez toujours développer votre Gradle construire dans la même branche où vous pouvez construire
- **Gradle Wrapper - Gradle Wrapper vous permet d'exécuter Gradle construit sur des machines où Gradle est pas installé.** Ceci est utile pour l'intégration continue des serveurs.
- **Open source gratuit - Gradle est un projet open source et sous licence Apache Software (ASL).**
- **Groovy - le script de construction de Gradle est écrit en Groovy.** Toute la conception de Gradle est orientée vers être utilisé comme une langue, non pas comme un cadre rigide. Groovy vous permet d'écrire votre propre script avec quelques abstractions. L'intégralité de l'API Gradle est conçu dans un langage Groovy.

3 Présentation

Gradle permet d'écrire des tâches de constructions dans un fichier de construction en utilisant le langage Groovy. Il est possible d'importer des tâches standards qui permettent de construire des programmes utilisant un ou plusieurs langages (Java, Groovy…) ou qui permettent d'exécuter des activités d'ingénierie logicielle telles qu'exécuter les tests unitaires, assurer la qualité du code (SonarQube, Checkstyle)…

3.1 Gradle reprend certaines des idées fortes de Maven :

- convention plutôt que configuration
- cycle de vie

- gestion des dépendances à la manière d'Apache Ivy ou Maven
- référentiel (ou entrepôts)

3.2 <u>Gradle présente les avantages suivants</u> :

- possibilité de scripter la construction en Groovy dans le fichier de construction ;
- possibilité de changer le comportement par défaut de certaines tâches ;
- une notation compacte pour décrire les dépendances ;
- un moteur de production pensé pour produire des projets multi-langages.

Gradle permet de construire sans effort des projets utilisant d'autres langages que Java. Migrer de Maven vers Gradle se fait très facilement pour un projet respectant les conventions Maven.

4 <u>les avantages et inconvénients des logiciels de gestion construction Gradle.</u>

<u>4.1 Qules sont les avantages et les inconvénients des logiciels de gestion construction Gradle sur Maven ?</u>

Permet de jeter un coup d'oeil en détails pourquoi android/google adopté gradle
Sont des points clés pour google à aller de l'avant avec gradle pour leur développement android

- Créer des APKs multiples pour votre application avec des caractéristiques différentes en utilisant le même projet. (créer plusieurs variantes d'une demande)
- Réutilisation de code et ressources.
- Personnaliser et configurer étendre le processus de génération.
- Intégration personnalisée avec leur propre Studio Android IDE.
- Outils de création simples à prendre en charge plusieurs langues.

Vous pouvez considérer au-dessus de points sont les avantages de gradle. gradle est DSL et par conséquent fournir beaucoup plus de souplesse pour définir votre logique.

Dans le même temps DSL rendrait peu difficile comme norme peut-être changer et vous pouvez finir par changer votre script de compilation.

Par ailleurs, avantages et inconvénients dépend de votre besoin et exigence.
Une caractéristique peut être bonne pour un seul système mais même caractéristique peut poser problème (pas très bon) pour l'autre système.

Jetons un coup d'oeil à Maven et Gradle outil de construction sur la base des points ci-dessus et leur donner score selon notre cas d'utilisation.
Voici la comparaison tabulaire pour chaque point entre Maven et Gradle.

Points	Maven	Gradle
Combien facile est la courbe d'apprentissage initiale	1) Maven XML est un outil basé, XML est très communément utilisé / connu. 2) Si le projet existant utilise maven alors les développeurs sont à l'aise avec le système.	1) Gradle est DSL système basé et doivent apprendre explicitement. 2) Les développeurs ont besoin d'apprendre un nouveau système dès le début s'ils ne sont pas familier
Quelle est la rapidité de la construction de chaque outil	Pris un rapport de zeroturnaround, ils ont fait l'analyse de détail de la vitesse des builds avec les deux outils construits et trouvé maven et gradle sont assez proches sur les timings de construction.	
Comment est-il complexe de créer et de maintenir le script de construction?	Les scripts de construction de Maven sont basés sur xml, qui a une structure prédéfinie et une seule façon d'écrire. Donc, il est plus standard et moins flexible.	Gradle a son propre DSL qui est introduit par Gradle Lui-même et étroitement lié à Gradle internes Mais son flexible et simple et court
Quelle est la qualité de la communauté et de la documentation pour chaque outil?	Maven est sur le marché depuis très longtemps, la documentation est bonne, beaucoup de ressources et d'aide disponibles dans la communauté ouverte et le forum.	Gradle est très récente. Il est open source mais toujours sous le contrôle de gradleware. Ils ont l'option pour le soutien commercial.

Choix

4.1.1 Gradle

Vous ne pouvez pas utiliser d'autres fonctionnalités qui est utilisé par google pour le développement android que ces fonctionnalités peu ou ne peu correspondent pas à vos besoins cependant gradle est essentiellement émerger comme outil populaire et vous pouvez certainement vouloir jeter un oeil à elle.

4.1.2 Maven

Maven est largement utilisé dans la plupart des entreprises et dispose d'un workflow robuste défini autour de lui. Les développeurs connaissent bien le système. Toutefois, cela peut ne pas être une option viable que le développement Android besoin d'utiliser gradle.

4.1.3 Maven and Gradle

Ce serait la seule option pour une entreprise avec une large gamme de zone de développement qui peut donner aux développeurs la flexibilité d'adopter l'outil selon leur choix dans un flux de travail donné.

4.2 Java Build Tools : Ant contre Maven Contre Gradle

4.2.1 Ant avec Ivy

Ant a été le premier parmi les outils de construction «modernes». Dans de nombreux aspects, il est similaire à Make. Il a été publié en 2000 et dans un court laps de temps est devenu l'outil de construction le plus populaire pour les projets Java. Il a une courbe d'apprentissage très faible permettant ainsi à quiconque de commencer à l'utiliser sans aucune préparation spéciale. Il est basé sur une idée de programmation procédurale.

Après sa première version, il a été amélioré avec la possibilité d'accepter les plug-ins.

Le principal inconvénient était XML comme format pour écrire des scripts de construction. XML, étant de nature hiérarchique, n'est pas un bon ajustement pour l'approche de programmation procédurale Ant utilise. Un autre problème avec Ant est que son XML tend à devenir très grand quand il est utilisé avec tous les projets, sauf très petits.

Plus tard, comme la gestion des dépendances sur le réseau est devenue un must, Ant a adopté Apache Ivy.

Principal avantage de Ant est son contrôle du processus de construction.

4.2.2 Maven

Maven a été libéré en 2004. Son but était d'améliorer certains des problèmes que les développeurs faisaient face en utilisant Ant.

Maven continue à utiliser XML comme format pour écrire la spécification de construction. Cependant, la structure est diamétralement différente. Bien que Ant requiert des développeurs d'écrire toutes les commandes qui mènent à l'exécution réussie de certaines tâches, Maven s'appuie sur des conventions et fournit les cibles (buts) disponibles qui peuvent être invoquées. Comme ajout supplémentaire, et probablement plus important, Maven a introduit la possibilité de télécharger des dépendances sur le réseau (plus tard adopté par Ant par Ivy). Cela a révolutionné la façon dont nous livrons le logiciel.
Cependant, Maven a ses propres problèmes. Gestion des dépendances ne gère pas bien les conflits entre les différentes versions de la même bibliothèque (quelque chose Ivy est beaucoup mieux à). XML comme format de configuration de construction est strictement structuré et hautement standardisé. La personnalisation des cibles (objectifs) est difficile. Puisque Maven est principalement axé sur la gestion des dépendances, les scripts de construction complexes et personnalisés sont plus difficiles à écrire dans Maven que dans Ant.
Maven configuration écrite en XML continue d'être gros et encombrant. Sur de plus grands projets, il peut avoir des centaines de lignes de code sans faire quoi que ce soit «extraordinaire». Le principal avantage de Maven est son cycle de vie. Tant que le projet est basé sur certaines normes, avec Maven on peut passer tout le cycle de vie avec une relative facilité. Cela entraîne un coût de flexibilité.
Dans l'intervalle, l'intérêt pour les DSL (Domain Specific Languages) a continué d'augmenter. L'idée est d'avoir des langages conçus pour résoudre des problèmes appartenant à un domaine spécifique. Dans le cas de builds, l'un des résultats de l'application de DSL est Gradle.

4.2.3 Gradle

Gradle combine de bonnes parties des deux outils et se construit sur eux avec DSL et d'autres améliorations. Il possède le pouvoir et la souplesse d'Ant avec le cycle de vie de Maven et sa facilité d'utilisation. Le résultat final est un outil qui a été publié en 2012 et a gagné beaucoup d'attention dans un court laps de temps. Par exemple, Google a adopté Gradle comme outil de construction par défaut pour le système d'exploitation Android.

Gradle n'utilise pas XML. Au lieu de cela, il avait son propre DSL basé sur Groovy (une des langues JVM). En conséquence, les scripts de génération de Gradle tendent à être beaucoup plus courts et plus clairs que ceux écrits pour Ant ou Maven. La quantité de code standard est beaucoup plus petite avec Gradle puisque son DSL est conçu pour résoudre un problème spécifique: déplacer le logiciel tout au long de son cycle de vie, de la compilation à l'analyse et au test statique jusqu'à l'emballage et au déploiement.

Initialement, Gradle a utilisé Apache Ivy pour sa gestion des dépendances. Plus tard, il a déménagé à son propre moteur de résolution de dépendance native.

L'effort de Gradle peut être résumé comme "la convention est bonne et la flexibilité".

4.2.4 Le choix du Gradle

Une des nouveautés annoncée à la Google I/O 2013 concernait la mise à disposition d’un nouveau système de build. Il était nécessaire de disposer d’un système commun facilitant la gestion de toutes les étapes de construction d’un projet. Plusieurs initiatives existaient à base de Maven, Ant... mais au final Google a décidé de s’appuyer sur Gradle

Gradle a été choisi pour plusieurs raisons

- il est plus souple dans son utilisation que Maven, qui impose une convention de développement plutôt adaptée au développement d'applications Java.
- il propose une gestion automatique des dépendances basée sur celle de Maven et/ou celle de Ivy tout en permettant de désactiver la résolution transitive.
- Un autre intérêt est d'avoir des fichiers de configuration en Groovy. La configuration se résume soit en des éléments déclaratifs se basant sur le DSL Gradle soit en des éléments de code Groovy
- son intégration commence à être correcte dans les IDE et notamment dans IntelliJ et AndroidStudio

Tout comme Maven, Gradle propose un processus basé sur des conventions. Si vous suivez le processus standard de construction vous n'avez pratiquement rien à paramétrer. Mais tout reste surchargeable pour que vous puissiez adapter vos spécificités.

5 Pourquoi Groovy?

L'API complète Gradle est conçu en utilisant un langage Groovy. Ceci est un avantage d'un DSL interne sur XML. Gradle est un outil de construction d'usage général à sa base; son objectif principal est des projets Java. Dans de tels projets, les membres de l'équipe seront très familiers avec Java et il est préférable que la construction devrait être aussi transparent que possible à tous les membres de l'équipe.

Langues comme Python, Groovy ou Ruby sont meilleurs pour le cadre de la construction. Pourquoi Groovy a été choisi est, car il offre de loin la plus grande transparence pour les personnes utilisant Java. La syntaxe de base de Groovy est identique à Java. Groovy fournit beaucoup plus en plus de cela.