Léna Francisco Do Carmo 71702495

Léia Lebail 71702346

Langages pour la JVM : Kotlin vs. Scala vs. Ocamljava vs. Clojure

Rapport d'exposé

Programmation Comparée

Nous allons aujourd'hui aborder le sujet des JVM, et plus particulièrement de quatre langages pouvant les utiliser : Kotlin, Scala, OCamljava et Clojure. Pour cela, nous allons donc en premier lieu vous présenter ce qu'est une JVM et quelle est son utilité, puis vous introduire ces quatre langages abordés juste avant et enfin réaliser des comparaisons entre ces langages.

La JVM

La **machine virtuelle Java** (**JVM**) est un environnement d'exécution pour des applications Java et un appareil informatique fictif qui permet d'exécuter des programmes compilés sous forme de bytecode Java. Le bytecode est un code binaire qui permet un traitement plus rapide que le code source Java.
On l'utilise dans le but de fournir un environnement de programmation indépendant de la plate-forme. Elle fait abstraction du matériel et éventuellement du système d'exploitation. La JVM permet donc à un programme de s'exécuter de la même manière peu importe la plate-forme.

Le compilateur Java faisant peu d'optimisations, c'est à la JVM de les faire, et pour ce faire c'est Sun qui décrit les optimisations, fonctions et propriétés que doivent respecter toutes les JVM, comme par exemple le compilateur à la volée ou plus surprenant, le ramasse-miettes qui n'est pas obligatoire mais fortement recommandé.
Il existe plusieurs JVM pour plusieurs projets et objectifs. Celle que nous utilisons couramment est HotSpot la JVM de chez Oracle, mais il existe aussi JamaicaVM, une machine virtuelle pour gérer les contraintes de temps réel de l'électronique embarqué ou encore la machine virtuelle de chez Google, Dalvik, pour Android.

Les langages

Kotlin est un langage qui a été conçu pour faire à la fois de l'orienté objet et du fonctionnel avec un typage statique (comme java) principalement par une équipe de chez JetBrains et dont la première version est apparue en 2011. Kotlin permet de compiler pour plusieurs machines virtuelles notamment Java ou Javascript. Ce langage tire son nom de l'île de Kotline située à proximité de St. Petersbourg d'où était basée l'équipe de chez JetBrains.

Kotlin est le second langage pris en charge par Android et est maintenant le premier langage recommandé par Google pour Android.

Scala est un langage de programmation multi-paradigme conçu à l'École polytechnique fédérale de Lausanne (EPFL) par Martin Odersky en 2004. Ce langage a été conçu pour exprimer les modèles de programmation courants dans une forme plus concise que les autres.
Scala intègre les paradigmes de programmation orientée objet et de programmation fonctionnelle, avec un typage statique. Il concilie ainsi ces deux paradigmes habituellement opposés (à de rares exceptions près, tel que le langage OCaml) et offre au développeur la possibilité de choisir le paradigme le plus approprié à son problème.

Xavier Clerc, à travers le compilateur **OcamlJava**, a voulu permettre une intégration transparente d'Ocaml et de Java dans le but de pouvoir utiliser les librairies externes Java qui sont assez pauvres en Ocaml. Le projet a vu le jour en 2013 et n'est plus maintenu depuis 2015. OcamlJava permet également la programmation simultanée dans un seul processus Ocaml.

Clojure a été conçu par Rich Hickey et publié pour sa première version en 2007. C'est un langage de programmation fonctionnel compilé, multi-plateforme et a été conçu dans l'optique d'avoir des programmes plus simples, rapides, sûrs et facilement distribuables. Clojure est un dialecte de Lisp, une ancienne famille de langages de programmation impératifs et fonctionnels créé en 1958.

Les comparaisons

Kotlin & Java

Kotlin permet d'utiliser des co-routines, qui appartiennent à la librairie kotlinx.coroutines et qui sont des threads allégés et facile d'utilisation qui sont asynchrones et non liés à un thread (on peut trouver la documentation en suivant ce lien : https://kotlinlang.org/docs/coroutines-guide.html). Il dispose d'héritage multiple, de fonctions d'extensions qui permettent de rajouter des fonctions à une classe à l'extérieur de celle-ci, ou encore de null safety avec l'utilisation d'un ? qui signifie "cette variable est potentiellement nulle" et qui implique l'utilisation de !, ce qui permet de vérifier qu'une variable est non nulle avant de l'utiliser.
En revanche Java permet l'encapsulation ce qui permet de gérer l'accessibilité aux variables et fonctions, les membres statiques qui n'existent pas en Kotlin ou encore permet l'utilisation de l'opérateur ternaire.
Kotlin est également plus concis (dû également à l'absence d'encapsulation) et est réputé plus simple d'apprentissage pour les débutants.

Scala & Java

En comparaison à du code Java, celui de Scala est plus concis.

Java a été conçu pour réaliser de la programmation orientée objet alors que Scala est conçu pour faire de l'orientée objet et du fonctionnel.
Un des avantages non négligeables de Java par rapport à Scala est que Java est rétrocompatible : Java garantit une rétrocompatibilité dans ses changements de versions alors que la communauté autour de Scala a fait le choix de s'autoriser quelques changements. Aujourd'hui, en Scala, il existe deux compilateurs. Le premier est maintenu par la société Typesafe, le second par la communauté Typelevel.

OcamlJava & Java

Les différences entre OcamlJava et Java sont les mêmes qu'entre Ocaml et Java.
Ocaml possède le filtrage par motif, la syntaxe est plus simple et possède une inférence de types.
Java en revanche est plus documenté et possède plus de librairies (une des raisons principales de la création d'OcamlJava).

Clojure & Java

Clojure est conçu pour la programmation fonctionnelle contrairement à Java.
Clojure possède des structures immuables, ce qui signifie que chaque opération que le développeur fait ne changera pas ses valeurs. Au lieu de les modifier, Clojure créé une nouvelle structure dans la forme d'un arbre qui contient la nouvelle donnée ajoutée. Cela est utile dans le cas où nous avons beaucoup de threads : au lieu de synchroniser et changer les variables dans chaque thread, Clojure va créer de nouvelles structures.
Grâce à cela, Clojure simplifie multithreading et améliore la concurrence, ce qui donne encore une fois un code plus concit que Java.
Cependant, un des avantages de Java est sécurisé dans le sens où Java est le premier langage de programmation qui inclus la sécurité comme une part de son design. Java réduit les risques en évitant d'utiliser des pointeurs explicites. Un pointeur peut causer des accès non autorisés dans la mémoire car il stocke l'adresse mémoire dans une autre valeur. Il y a également un gestionnaire de sécurité en Java pour chaque application qui définit les règles d'accès des classes.

Kotlin & Scala

Comme vu précédemment lors de nos comparaisons de Kotlin et Scala à Java, ces deux langages ont des similitudes telles que la concision du code ou l'absence d'encapsulation. Scala est néanmoins plus populaire notamment parce qu'ils proposent plus de facilité du coté fonctionnel (et pas seulement orienté objet) comme le filtrage par motif.
Kotlin en revanche a un typage plus simple que celui de Scala (qui ressemble au typage de Java) et a pour avantage de pouvoir utiliser des variables null sans lever d'exceptions (null safety).
Il est en revanche assez simple d'utiliser du code scala à partir de code kotlin, il suffit de faire attention à bien avoir compiler le code scala en bytecode (et donc obtenir un fichier .class) pour que kotlin le trouve et l'utilise.

OcamlJava & Clojure

Clojure est utilisé pour des applications webs tandis qu'OcamlJava a été conçu dans l'optique de réaliser des preuves mathématiques.
Clojure est basé sur Lisp (comme dit plus haut) et donc possède une communauté plus développée que OcamlJava, qui n'a pas été maintenu depuis 2015 et possède très peu de documentation.
Cependant, comme désavantage pour Clojure, nous savons que le lancement des applications Clojure est connu pour être lent.

Pour conclure

La machine virtuelle Java est un outil très pratique qui nous permet d'utiliser le langage de notre choix tout en pouvant bénéficier des avantages de Java par ses très nombreuses librairies bien documentées qui n'existent pas dans d'autres langages. Il n'y a pas de "langage parfait" meilleur que les autres à utiliser absolument en priorité : chacun de ces langages a ses avantages et inconvénients, le tout dépend donc de son utilisateur, ses préférences et ses besoins.

Sources

Kotlin
Kotlin JVM
Scala
Scala VS Java
OcamlJava
Clojure
Clojure VS Java