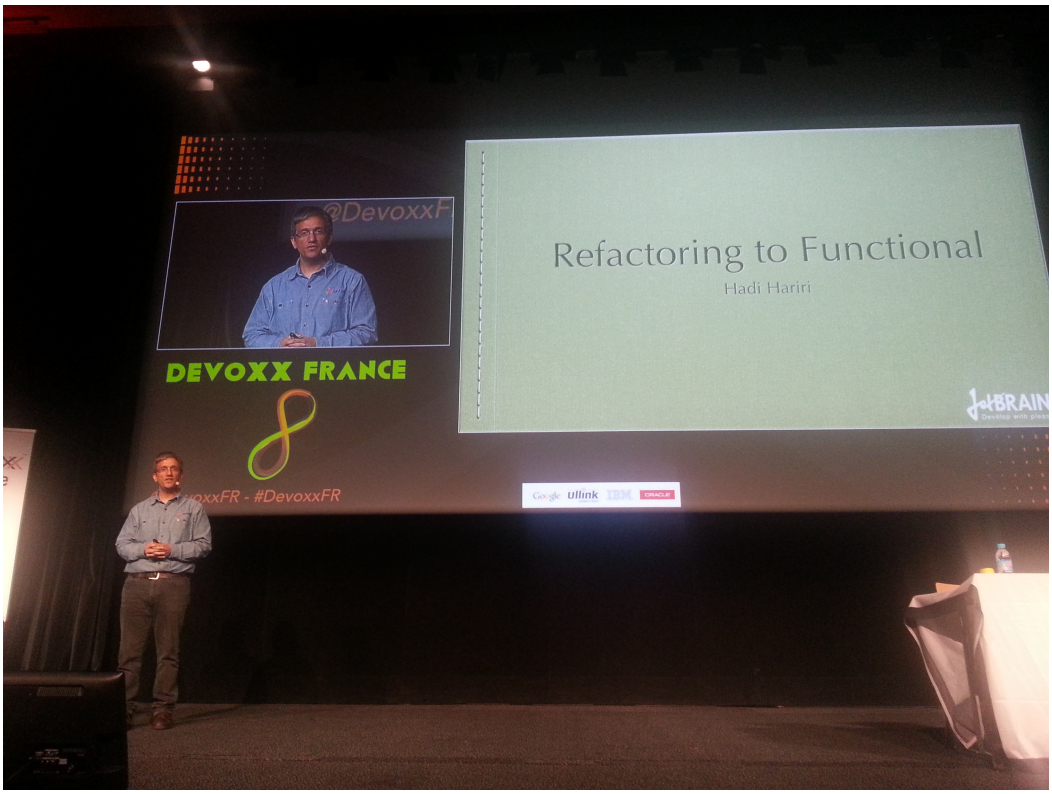
Refactoring to Functional

Date : 9 avril 2015
Format : Conférence
Speaker : Hadi Hariri, Jetbrains.



Hadi commence par une introduction sur la programmation fonctionnelle.
Pourquoi refactorer du code en fonctionnel ?

1. Ecrire moins de code : plus besoin de trouver des noms de variables intermédiaires
2. Ecrire du code expressif
3. Ecrire du code correct
4. Ecrire du code performant

Peut-on utiliser n'importe quel language ?
Non, car un language fonctionnel a certaines caractéristiques : immutabilité, higher-order functions, lambdas, top-level functions. Java et C# ne les ont pas toute.
Dans ses exemples, Hadi va utiliser Kotlin, le langage développé par Jetbrains. Mais il précise qu'on peut utiliser d'autres langages.

Une fonction pure a les caractéristiques suivantes :

- Son exécution n'a pas de side effect
- Le même entrée entraine la même sortie

Comment refactorer du code comportant des boucles, des compteurs … .

1. Retirer les états
2. Utiliser des sucres syntaxiques comme foreach
3. Utiliser des filtres en lui passant une condition ou les map pour changer le type de donnée retourné

La programmation fonctionnelle peut s'appliquer à plusieurs niveaux :

- Un service de recherche de clients par prédicats
- Templates patterns : moins de boilerplate code en passant les fonctions différenciantes en paramètres
- Ne pas oublier de fermer ses ressources à la fin d'un traitement
- Strategy Pattern : exemple avec différents algorithmes de code. Ce qui les différencie, ce n'est pas les données mais le comportement (l'algorithme).

Comment injecter les dépendances ?
Trop de dépendances cassent le principe de séparation des préoccupations.
Réponse : on peut passer le comportement en paramètres des méthodes.

Hadi met en garde quant à la lisibilité du code fonctionnel. On peut également enchainer les appels de fonctions. Il propose d'utiliser des fonctions intermédiaires.

Currying : fonction prenant N paramètres et retournant une fonction prenant N-1 paramètres. Ce sont des fonctions partielles.

What about data ?

Fold : fonction qui prend un accumulateur en paramètre
Exemple d'implémentation du calcul du max d'une liste.

What about performance ?

Exemple d'une suite de fibonacci implémentée avec un algorithme récursif et une approche itérative. Possibilité d'utiliser la Memoization en fonctionnel pour avoir les mêmes performances qu'en itératif.

Utilisation du tailRecursive pour ne pas faire tomber la pile d'appel.

Livre recommandé : Functionnal Programming in Java