Thème de Veille : Java Developpement Kit 8

Introduction :

Le 18 Mars 2014, oracle mettait à la disposition des développeurs du monde la version 8 de Java. Cette version a pour objectif outre de moderniser un peu plus le langage, mais aussi pouvoir répondre aux besoins de plus en plus précis des développeurs notamment sur les technologies lié au cloud.
Mais aussi oracle souhaitait par cette nouvelle version accroitre la sécurité de leur langage qui pour la version 7 s'été traduit par de nombreuse mise à jour.

Nouveauté de la version 8

Java 8 apporte des nouveautés dans le langage:

- les interfaces fonctionnelles : connue sous le nom : Single Abstract Method interfaces, cette nouveauté introduit les interfaces qui ne possèdent uniquement que une seule méthode d'instance abstraite. Avec Java 8, elles portent donc le nom d 'interface fonctionnelle et offre la possibilité d'annoter l'interface par @funtionalInterface . Et si une Interface est annotée et qu'elle possède plus d'une Méthode d'instance abstraite, une erreur lors de la compilation sera produite.

- Les fonctions lambdas : Il s'agit de la nouveauté la plus importante de java 8, cette fonctionnalité permet d'apporter la puissance de la programmation fonctionnelle dans Java. Une expression lambda peut être assimilée à une fonction anonyme, ayant potentiellement accès au contexte (variables locales et/ou d'instance) du code appelant. Ces "fonctions anonymes" peuvent être affectées dans une interface fonctionnelle. Le code de l'expression lambda servira ainsi d'implémentation pour la méthode abstraite de l'interface. On peut donc les utiliser avec n'importe quel code Java utilisant une telle interface, à condition que les signatures de la méthode correspondent à celle de l'expression lambdas.

  Exemple: Sur une fonciton Trier

```
1 Arrays.sort(testStrings, new Comparator<String>() {
2     @Override
3     public int compare(String s1, String s2) {
4         return(s1.length() - s2.length());
5     }
6 });
```

En utilisant les lambdas, la nouvelle écriture sera :
Code :

```
1 // Forme longue :
2 Arrays.sort(testStrings, (String s1, String s2) -> { return s1.length() - s2.length(); });
3
4 // Forme courte (possible uniquement s'il n'y a qu'une instruction) :
5 Arrays.sort(testStrings, (String s1, String s2) -> s1.length() - s2.length());
6
7 // Forme courte avec type implicite des paramètres
8 // (le type est déduit par le compilateur via l'inférence)
9 Arrays.sort(testStrings, (s1, s2) -> s1.length() - s2.length());
```

- La référence de méthode: une référence de méthode est utilisée pour définir une méthode en tant qu'implémentation de la méthode abstraite d'une interface fonctionnelle. La notation utilise le nom de la classe ou une instance de la classe, suivi de l'opérateur « :: » et du nom de la méthode à référencer. Le type des paramètres sera déduit du contexte selon l'interface fonctionnelle vers laquelle on affecte la référence.

- Méthodes par défaut (Defender Methods) : cette fonctionnalité permet de proposer une implémentation dite par "défaut" aux méthodes déclarées dans les interfaces. Par conséquent, depuis Java 8, une interface Java contient du code. L'avantage est de pouvoir faire évoluer les interfaces sans avoir à tout casser.

  représentation du code:

```
interface Person {
    void sayGoodBye();
    default void sayHello() {
        System.out.println("Hello there!");
    }
}
```

- Méthodes static dans les interfaces : Java 8 propose également la possibilité de créer des méthodes statiques depuis une interface Java.

L'autre point est la mise à jour de l'API:

Stream et parallèles streams sur les collections : Java 8 apporte également la notion de Stream, qui représente un flux de données que l'on peut manipuler à la volée. L'utilisation d'un Stream se compose de trois parties :

- La création du flux à partir d'une source de donnée qui peut être très variée (un tableau, une collection, un flux d'entrée/sortie, des données générés à la volée, etc.)*;
- Des opérations intermédiaires, qui permettent de transformer le flux en un autre flux (en filtrant des données ou en les transformant par exemple)*;
- Une opération terminale, qui permet d'obtenir un résultat ou d'effectuer une opération spécifique.

On peut citer la mise a jour de l'API java.time qui gère date, heure et calendrier.

Java 8 offre la possibilité de répéter plusieurs fois une annotations de même type pour un élément donné d'un programme.

```
@Schedule(dayOfMonth="last")
@Schedule(dayOfWeek="Fri", hour="23")
public void doPeriodicCleanup() { ... }
```

Et enfin un nouveau moteur javascript : Nashorn , moteur proposé avec le JDK.

En conclusion:

La version 8 de java que Oracle à lancer officiellement au mois de Mars montre que le langage se veut plus moderne dans ses fonctionnalités et surtout essaie de simplifier ces fonctions pour rendre le code plus efficace et de faire du langage un outil adapté au Cloud.

Sources:

Developpez.com:
http://www.developpez.com/actu/68971/Java-8-est-disponible-la-plate-forme-se-met-aux-expressions-lambdas-tour-d-horizon-des-nouveautes/

Blogs Oracle:
https://blogs.oracle.com/java/entry/java_se_8_is_now

OpenJdk:
http://openjdk.java.net/projects/jdk8/features

PCInpact :
http://www.pcinpact.com/news/86565-java-8-debarque-enfin-avec-ses-expressions-lambda-et-sa-securite-renforcee.htm

Outil de veille:
RSSOwl