

JAVA INSIDE

TP 04 - Performance, CallSite et JMH

- MethodHandle
- MutableCallSite

Lien GitHub : <https://github.com/MelissaDaCosta/java-inside>

Mélissa DA COSTA

Test-driven development (TDD)

Méthode de développement qui consiste à écrire des tests avant le code source.

Les tests écrits ne peuvent donc pas encore être ok.

=> Spécifier le comportement souhaité du logiciel en fonction de son utilisation.

Tests de performance avec JMH

JMH est un framework java pour créer et lancer des benchmark (des essais).

Exemple d'un benchmark :

```
@Benchmark  
public void simple_logger() {  
    Foo.LOGGER.log("test");  
}
```

Logger

Un logger permet de se connecter à une application.

Dans ce TP : un logger exécute une méthode donnée en argument via l'interface fonctionnelle Consumer sur une String.

```
public interface Logger {  
public void log(String message);  
public static Logger of(Class<?> declaringClass, Consumer<?  
super String> consumer) {..} ... }
```

Plus une méthode qui crée la MethodHandle de Consumer :

```
private static MethodHandle  
createLoggingMethodHandle(Class<?> declaringClass,  
Consumer<? super String> consumer)
```

Résultat JMH : avec une lambda

La méthode of utilisait une classe anonyme pour créer un Logger :

Result simple_logger :

(min, avg, max) = (4,310, 4,391, 4,645), stdev = 0,080

En créant la méthode fastOf qui utilise une lambda au lieu de la classe anonyme :

Result fast_logger :

(min, avg, max) = (0,572, 0,605, 0,842), stdev = 0,068

Explications des résultats

L'opération fastOf est nettement plus rapide. Elle à le même coût que si on ne faisait aucune opération car avec la lambda c'est comme ci elle remplissait le code par « rien ».

Dans le cas ou la lambda est considéré comme une constante, alors toutes les variables sont considérées comme des constantes

Résultat JMH : Créer et désactiver

Avec la nouvelle méthode ***createLoggingMethodHandle*** qui utilise ***guardWithTest***.

Voici le benchmarck qui crée un logger et le désactive :

@Benchmark

```
public void create_and_disable_logger() {  
    Foo.LOGGER.log("test");  
    Logger.enable(Foo.class, false);  
}
```

Voici les résultats de ce benchmark :

Result "fr.umlv.javainside.lab4.LoggerBenchMark.create_and_disable_logger":

157,773 ±(99.9%) 28,997 ns/op [Average]

(min, avg, max) = (135,455, 157,773, 212,947), stdev = 27,124

CI (99.9%): [128,776, 186,770] (assumes normal distribution)

Le coût est énorme.