JAVA INSIDE
TP 06 - Continuation

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

Mélissa DA COSTA

Continuation VS Thread

CONTINUATION	THREAD
Exécute Runnable une seule fois	
	Contient une continuation Il peut en exécuter plusieurs
Scheduling : C'est le code que l'on écrit dans l'application	Scheduling géré par l'OS
Concurrence coopérative car pas d'interruption	Concurrence : Non coopérative / racy / compétitif

Exemple de création d'une continuation

```
var continuationScope = new
ContinuationScope("hello1");
var continuation = new
Continuation(continuationScope, ()→{

System.out.println("hello continuation");
});

continuation.run();
```

Continuation.yield()

L’appel à yield sauvegarde la pile d’exécution et redonne la main au thread ayant appelé la continuation.

```
var continuation = new
Continuation(continuationScope, ()->{
Continuation.yield(continuationScope);
System.out.println("hello continuation");
});

continuation.run();
```

Cette exemple n’affiche rien.

Continuation.run()

L’appel à run() replace le bout de la pile d’exécution.

```
var continuation = new
Continuation(continuationScope, ()->{
Continuation.yield(continuationScope);
System.out.println("hello continuation");
});

continuation.run();
continuation.run();
```

Affiche « hello continuation »

Le premier run() lance la continuation.

Le deuxième run() reprend l’exécution de la continuation après l’appel à yield().

Continuation.getCurrentContinuation(scope)

A l'intérieur d'un runnable on peut demander la continuation courante.

Il y peut y avoir plusieurs continuation les unes dans les autres avec des scopes différents.

Si yield : dans qu'elle scope va-t-il ?

Donc on doit lui donner

Scheduler

Permet de planifier l'exécution des continuations. Ici, il exécute la dernière continuation enregistrée.

```
public void enqueue(ContinuationScope scope) {
    if(Continuation.getCurrentContinuation(scope) == null) {
        throw new IllegalStateException();
}
    this.deque.offer(Continuation.getCurrentContinuation(scope));
    Continuation.yield(scope);
}

public void runLoop() {
    while(!deque.isEmpty()) {
        Continuation continuation = null;
        continuation = deque.pollLast();
        continuation.run();
    }
}
```