Controlling What Thread Can Execute a Task

José Paumard
PHD, Java Champion, JavaOne RockStar

@JosePaumard https://github.com/JosePaumard

Agenda

You have a total control on the threads that are executing your tasks

What are the threads used by default?

How can you control these threads?

Default Threads

Asynchronous tasks are executed
in the Common Fork / Join pool

The Common Fork / Join pool is a pool of threads

The number of threads is the number of cores on your machine

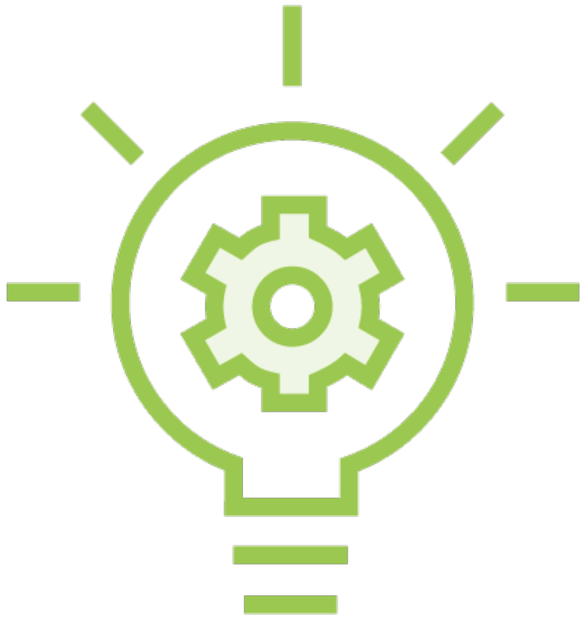
Each thread has its own queue of task

Each queue can steal tasks from another queue

Is this what you need?

Asynchronous programming is mostly used for I/O operations

Where you may need more threads than your number of cores



What about having the following?

- a pool of threads for your DB operations

- another one for your disk operations

- a last one for your in-memory operations

And what about GUI operations that needs to operate in a special thread?

You can specify an Executor for all the CompetableFuture operations

Either you pass an executor

Or you call a method ending with async that takes an executor

By default,
a task is executed in the same thread
as the one that created it

If the method name ends with ASYNC
then it may be executed
in a different thread

```
CompletableFuture<Quotation> quotationCF =
        CompletableFuture.supplyAsync(
            () -> getQuotation());

CompletableFuture<EmailInfos> infosCF =
        quotationCF.thenApply(
            quotation -> email(quotation));

CompletableFuture<Boolean> doneCF =
        infosCF.thenApply(
            emailInfos -> writeToDB(emailInfos));

doneCF.thenApply(done -> updateGUI(done));
```

```
CompletableFuture<Quotation> quotationCF =
        CompletableFuture.supplyAsync(
            () -> getQuotation());

CompletableFuture<EmailInfos> infosCF =
        quotationCF.thenApplyAsync(
            quotation -> email(quotation));

CompletableFuture<Boolean> doneCF =
        infosCF.thenApplyAsync(
            emailInfos -> writeToDB(emailInfos));

doneCF.thenApplyAsync(done -> updateGUI(done));
```

```
Executor executor = Executors.newFixedThreadPool(4);

CompletableFuture<Quotation> quotationCF =
        CompletableFuture.supplyAsync(
            () -> getQuotation(), executor);

CompletableFuture<EmailInfos> infosCF =
        quotationCF.thenApplyAsync(
            quotation -> email(quotation), executor);

CompletableFuture<Boolean> doneCF =
        infosCF.thenApplyAsync(
            emailInfos -> writeToDB(emailInfos), executor);

doneCF.thenApplyAsync(done -> updateGUI(done),
                      SwingUtilities::invokeLater);
```

```
doneCF.thenApplyAsync(done -> updateGUI(done),
                      SwingUtilities::invokeLater);
```

```
public interface Executor {

    void execute(Runnable task);
}
```

Demo

Let us see some code!

Check what thread is executing what task

And provide a specific executor to run some tasks

Module Wrap Up

What did you learn?

What thread is executing what task

How to provide a thread pool to execute a specific task

Up Next: Reporting and Recovering from Errors