Closing Remarks



José Paumard PHD, Java Champion, JavaOne RockStar

@JosePaumard https://github.com/JosePaumard

Agenda



Understanding how the API is organized

- what model for your tasks
- tasks chaining 1 1 and P 1
- exception handling

Performances

Creating a Task

Creating a CompletableFuture

Task	Example	Pattern	Executor
Supplier	<pre>() -> getWeather()</pre>	<pre>CompletableFuture<weather> cf = CompletaleFuture .supplyAsync(supplier)</weather></pre>	yes
Runnable	<pre>() -> logger.info("Done")</pre>	<pre>CompletableFuture<void> cf = CompletableFuture .runAsync(runnable)</void></pre>	yes

Creating a CompletableFuture

Task	Pattern	Executor
Completes when all completes	CompletableFuture <void> cf = CompletableFuture .allOf()</void>	no
Completes on one of the fastest	CompletableFuture <object> cf = CompletableFuture .anyOf()</object>	no

Modeling Tasks

CompletableFuture Supported Tasks

Task	Example	Method	CF method
Runnable	<pre>() -> logger.info("System OK")</pre>	void run()	thenRun()
Consumer	<pre>n -> logger.info(n + " users read")</pre>	void accept(Long)	thenAccept()
Function	<pre>id -> readUserFromDB(id)</pre>	User apply(Long)	thenApply()

Chaining tasks

1-1 CompletableFuture Chaining

	Parameter type	Async	Executor
thenRun()	Runnable	yes	yes
thenAccept()	Consumer <t></t>	yes	yes
thenApply()	Function <t, u=""></t,>	yes	yes
thenCompose()	Function <t, completionstage<u="">></t,>	yes	yes

2 – 1 CompletableFuture Chaining

	Parameter type	Async	Executor
runAfterBoth()	Runnable	yes	yes
thenAcceptBoth()	BiConsumer <t, u=""></t,>	yes	yes
thenCombine()	BiFunction <t, u,="" v=""></t,>	yes	yes

2 – 1 CompletableFuture Chaining

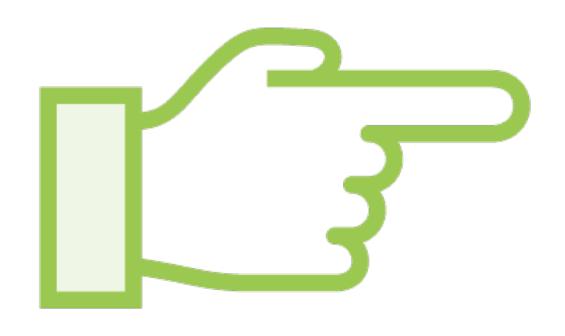
	Parameter type	Async	Executor
runAfterEither()	Runnable	yes	yes
acceptEither()	Consumer <t></t>	yes	yes
applyToEither()	Function <t, v=""></t,>	yes	yes

Exception Handling

Exception Handling

	Parameter type	Can recover	Async
exceptionally()	Function <throwable, t=""></throwable,>	yes	yes
handle()	BiFunction <t, throwable,="" u=""></t,>	yes	yes
whenComplete()	BiConsumer <t, throwable=""></t,>	no	yes

Performances



Identify in-memory vs. I/O computation

Use asynchronous calls for long running tasks

Use executors with the right number of threads

Avoid moving data from one thread to the other

https://jdk.java.net/loom

Course Wrap Up



What did you learn?

The CompletionStage API

Create efficient asynchronous pipelines

How to model a task, how to chain them

How to recover from exceptions

Up Next: Design Efficient Applications!