CompletionStage<T> Quick Reference

Jesper de Jong - http://www.jesperdj.com

	Method group	Arguments	Return type	Notes
<U>	thenApply(Async)	Function<? super T, ? extends U>	CompletionStage<U>	Apply a function to the result of the previous stage
	thenAccept(Async)	Consumer<? super T>	CompletionStage<Void>	Let a consumer deal with the result of the previous stage
	thenRun(Async)	Runnable	CompletionStage<Void>	Run an action without argument and return value
<U, V>	thenCombine(Async)	CompletionStage<? extends U>, BiFunction<? super T, ? super U, ? extends V>	CompletionStage<V>	Apply a function to the results of both of two previous stages
<U>	thenAcceptBoth(Async)	CompletionStage<? extends U>, BiConsumer<? super T, ? super U>	CompletionStage<Void>	Let a consumer deal with the results of both of two previous stages
	runAfterBoth(Async)	CompletionStage<?>, Runnable	CompletionStage<Void>	Run an action when both of two previous stages complete
<U>	applyToEither(Async)	CompletionStage<? extends T>, Function<? super T, U>	CompletionStage<U>	Apply a function to the result of either of two previous stages
	acceptEither(Async)	CompletionStage<? extends T>, Consumer<? super T>	CompletionStage<Void>	Let a consumer deal with the result of either of two previous stages
	runAfterEither(Async)	CompletionStage<?>, Runnable	CompletionStage<Void>	Run an action when either of two previous stages completes
<U>	thenCompose(Async)	Function<? super T, ? extends CompletionStage<U>>	CompletionStage<U>	Like thenApply, except that the function returns CompletionStage<U> instead of U
	whenComplete(Async)	BiConsumer<? super T, ? super Throwable>	CompletionStage<T>	Handle the outcome of a stage, whether it's a result value or an exception
<U>	handle(Async)	BiFunction<? super T, ? super Throwable, ? extends U>	CompletionStage<U>	Handle the outcome of a stage and return a new value
	exceptionally	Function<Throwable, ? extends T>	CompletionStage<T>	When an exception happened, replace the exception with a result value
	toCompletableFuture		CompletableFuture<T>	Convert the CompletionStage to a CompletableFuture