Understanding Method Groupings in the Java Completable Futures API (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Recognize how Java completable futures overcome limitations with Java futures
• Understand how methods are grouped in the Java completable future API
Learning Objectives in this Part of the Lesson

• Recognize how Java completable futures overcome limitations with Java futures
• Understand how methods are grouped in the Java completable future API
Birds-Eye View of the Java Completable Future API
The entire completable future framework resides in 1 public class with 60+ methods!!!
This framework implements the Façade pattern. 

Provide a unified interface to a set of interfaces in a sub-system that makes the subsystem easier to use.

See en.wikipedia.org/wiki/Facade_pattern
Grouping the Java Completable Future API
Grouping the Java Completable Future API

- Given the large # of methods in this API it helps to have a “birds-eye” view

See [en.wikipedia.org/wiki/Earthrise](en.wikipedia.org/wiki/Earthrise)
Grouping the Java Completable Future API

- Some completable future features are basic.
Grouping the Java Completable Future API

- Some completable future features are basic
- e.g., the Java Future API + some simple enhancements

Only slightly better than the conventional Future interface
Other completable future features are more advanced
Grouping the Java Completable Future API

- Other completable future features are more advanced
- Factory methods

See en.wikipedia.org/wiki/Factory_method_pattern
Other completable future features are more advanced
- Factory methods
  - Initiate async computations without explicit thread use
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
    - Initiate async computations without explicit thread use
  - Both one-way & two-way

![Diagram of CompletableFuture API with factory methods highlighted]

- supplyAsync(Supplier<U>): CompletableFuture<U>
- supplyAsync(Supplier<U>, Executor): CompletableFuture<U>
- runAsync(Runnable): CompletableFuture<Void>
- runAsync(Runnable, Executor): CompletableFuture<Void>
- completedFuture(U): CompletableFuture<U>
- thenApply(Function<? super T>): CompletableFuture<U>
- thenAccept(Consumer<? super T>): CompletableFuture<Void>
- thenCombine(CompletionStage<? extends U, BiFunction<?, > void>: CompletableFuture<V>
- thenCompose(Function<? super T>): CompletableFuture<U>
- whenComplete(BiConsumer<? super U>): CompletableFuture<Void>
- allOf(CompletableFuture[]): CompletableFuture<T>
- anyOf(CompletableFuture[]): CompletableFuture<Object>
Grouping the Java Completable Future API

- Other completable future features are more advanced
- Factory methods
  - Initiate async computations without explicit thread use
- Both one-way & two-way

```
CompletableFuture()
cancel(boolean): boolean
iscancelled(): boolean
isdone(): boolean
get()
get(long, TimeUnit)
join()
complete(T): boolean

supplyAsync(Supplier<T>): CompletableFuture<T>
supplyAsync(Supplier<T>, Executor): CompletableFuture<T>
runAsync(Runnable): CompletableFuture<Void>
runAsync(Runnable, Executor): CompletableFuture<Void>
completedFuture(T): CompletableFuture<T>
thenApply(Function<? super T>: CompletableFuture<T>
thenAccept(Consumer<? super T>: CompletableFuture<T>
thenCombine(CompletionStage<? extends U>, BiFunction<>: CompletableFuture<U>
thenCompose(Function<? super T>: CompletableFuture<T>
whenComplete(Consumer<? super T>: CompletableFuture<T>
allOf(CompletableFuture[]): CompletableFuture<Void>
anyOf(CompletableFuture[]): CompletableFuture<Object>
```

Help make programs more *elastic* by leveraging a pool of worker threads.
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - Chain actions that process results of async operations

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
    - Chain actions that process results of async operations
    - Can trigger actions based on 1 or more prior operations

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
    - Chain actions that process results of async operations
    - Can trigger actions based on 1 or more prior operations

Help make programs more *responsive* by not blocking caller code.
End of Understanding Method Groupings in the Java Completable Futures API (Part 1)