Understanding Method Groupings in the Java Completable Futures API (Part 2)

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

Arity is the # of params accepted by a method

See en.wikipedia.org/wiki/Arity
Other completable future features are more advanced

- Factory methods
- Completion stage methods
- “Arbitrary-arity” methods
- Process futures in bulk by combine multiple futures into a single future

<table>
<thead>
<tr>
<th>Grouping the Java Completable Future API</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Java Class Diagram" /></td>
</tr>
</tbody>
</table>

```java
class CompletableFuture<T> {  
  CompletableFuture()  
  cancel(boolean): boolean  
  isCancelled(): boolean  
  isDone(): boolean  
  get()  
  get(long, TimeUnit)  
  join()  
  complete(T): boolean  
  supplyAsync(Supplier<U>): CompletableFuture<U>  
  runAsync(Runnable): CompletableFuture<Void>  
  completedFuture(U): CompletableFuture<U>  
  thenApply(Function<?>) : CompletableFuture<U>  
  thenAccept(Consumer<? super T>): CompletableFuture<Void>  
  thenCombine(CompletionStage<? extends U>, BiFunction<?>) : CompletableFuture<V>  
  thenCompose(Function<?>) : CompletableFuture<U>  
  whenComplete(BiConsumer<?>) : CompletableFuture<T>  
  allOf(CompletableFuture[]): CompletableFuture<Void>  
  anyOf(CompletableFuture[]): CompletableFuture<Object>  
}  ```
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
- Process futures in bulk by combine multiple futures into a single future
  - Single future triggered when all complete
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
  - Process futures in bulk by combine multiple futures into a single future
    - Single future triggered when *all* complete
    - Single future triggered when *first* completes
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
- Process futures in bulk by combine multiple futures into a single future
  - Single future triggered when *all* complete
  - Single future triggered when *first* completes

Help make programs more *responsive* by not blocking caller code
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
- Process futures in bulk by combine multiple futures into a single future
  - Single future triggered when \( \textit{all} \) complete
  - Single future triggered when \( \textit{first} \) completes

Complicated to program directly & are best served via defining a wrapper!
Other completable future features are more advanced:

- Factory methods
- Completion stage methods
- "Arbitrary-arity" methods
- Exception methods
Other completable future features are more advanced
- Factory methods
- Completion stage methods
- “Arbitrary-arity” methods
- Exception methods
- Handle exceptional conditions at runtime
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
- Exception methods
  - Handle exceptional conditions at runtime
  - These methods are essential since async exceptions are different than sync exceptions

See mincong.io/2020/05/30/exception-handling-in-completable-future
Grouping the Java Completable Future API

- Other completable future features are more advanced
  - Factory methods
  - Completion stage methods
  - “Arbitrary-arity” methods
  - Exception methods
    - Handle exceptional conditions at runtime

Help make programs more resilient by handling erroneous computations gracefully
Grouping the Java Completable Future API

- All methods are implemented internally via a message-passing framework

Ensures loose coupling, isolation, & location transparency between components
All methods are implemented internally via a message-passing framework.

Various Java thread pools are used to process the messages.

End of Understanding Method Groupings in the Java Completable Futures API (Part 2)