Advanced Java CompletableFuture Features:

Two Stage Completion Methods (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

- Understand advanced features of completable futures, e.g.
  - Factory methods initiate async computations
- Completion stage methods chain together actions to perform async result processing & composition
  - Method grouping
  - Single stage methods
  - Two stage methods (and)

Completion stage methods

- Basic methods
- Arbitrary-arity methods
- Exception methods
- Factory methods
Methods Triggered by Completion of Both of Two Stages
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - thenCombine()

CompletableFuture<U> thenCombine(CompletionStage<? Extends U> other, 

{ ... }

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenCombine
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
- thenCombine()
  - Applies a bifunction action to two previous stages’ results

```java
CompletableFuture<U> thenCombine(
    CompletionStage<? Extends U> other,
    BiFunction<? super T,
             ? super U,
             ? extends V> fn)
{ ... }
```

See en.wikipedia.org/wiki/Logical_conjunction
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - thenCombine()
    - Applies a bifunction action to two previous stages’ results
      - Two futures are used here:
        - The future used to invoke thenCombine()
        - The ‘other’ future passed to thenCombine()

CompletableFuture\<U\> thenCombine
  (CompletionStage<\? Extends U\> other,
   BiFunction<\? super T,\n           \? super U,\n           \? extends V\> fn)
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - thenCombine()
    - Applies a bifunction action to two previous stages' results
  - Returns a future containing the result of the action

```java
CompletableFuture<? extends U> thenCombine
    (CompletionStage<? extends U> other,
     BiFunction<? super T, ? super U,
                 ? extends V> fn)
```

{ ... }
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
- `thenCombine()`
  - Applies a bifunction action to two previous stages’ results
- Returns a future containing the result of the action

```java
CompletableFuture<? extends U> thenCombine
  (CompletionStage<? Extends U> other,
{ ... }
```

`thenCombine()` essentially performs a “reduction”
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - `thenCombine()`
    - Applies a bifunction action to two previous stages’ results
    - Returns a future containing the result of the action
    - Used to “join” two paths of asynchronous execution

```java
CompletableFuture<BF> compF1 = CompletableFuture.supplyAsync(() ->
    /* multiply two BFs. */);

CompletableFuture<BF> compF2 = CompletableFuture.supplyAsync(() ->
    /* divide two BFs. */);

compF1.thenCombine(compF2, BigFraction::add)
    .thenAccept(System.out::println);
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex8)
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - thenCombine()
    - Applies a bifunction action to two previous stages’ results
    - Returns a future containing the result of the action
    - Used to “join” two paths of asynchronous execution

Asynchronously multiple & divide two big fractions

```java
CompletableFuture<BF> compF1 = CompletableFuture.supplyAsync(() -> /* multiply two BFs. */);
CompletableFuture<BF> compF2 = CompletableFuture.supplyAsync(() -> /* divide two BFs. */);
compF1.thenCombine(compF2, BigFraction::add)
    .thenAccept(System.out::println);
```
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - `thenCombine()`
    - Applies a bifunction action to two previous stages’ results
    - Returns a future containing the result of the action
    - Used to “join” two paths of asynchronous execution

`thenCombine()`’s action is triggered only after its two associated futures complete

```java
CompletableFuture<BF> compF1 = CompletableFuture.NOT_READY_SUPPLIER
                             .supplyAsync(() -> /* multiply two BFs. */);

CompletableFuture<BF> compF2 = CompletableFuture.NOT_READY_SUPPLIER
                             .supplyAsync(() -> /* divide two BFs. */);

compF1.thenCombine(compF2, BigFraction::add)    
       .thenAccept(System.out::println);
```
Methods Triggered by Completion of Both of Two Stages

- Methods triggered by completion of both of two previous stages
  - thenCombine()
    - Applies a bifunction action to two previous stages’ results
    - Returns a future containing the result of the action
    - Used to “join” two paths of asynchronous execution

```java
CompletableFuture<BF> compF1 = CompletableFuture.supplyAsync(() -> /* multiply two BFs. */);

CompletableFuture<BF> compF2 = CompletableFuture.supplyAsync(() -> /* divide two BFs. */);

compF1.thenCombine(compF2, BigFraction::add)
  .thenAccept(System.out::println);
```

Print out the results
End of Advanced Java
CompletableFuture Features:
Two Stage Completion Methods (Part 1)