Java CompletableFuture Futures ImageStreamGang

Example: StreamOfFuturesCollector

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 the design of the Java completable future version of ImageStreamGang
- Know how to apply completable futures to ImageStreamGang, e.g.
  - Factory methods
  - Completion stage methods
  - Arbitrary-arity methods
  - Wrap the allOf() method to work with the Java streams framework
Implementing the Class StreamOfFuturesCollector
Implementing the Class StreamOfFuturesCollector

StreamOfFuturesCollector wraps allOf() to work with the Java streams framework
StreamOfFuturesCollector is a non-concurrent collector (supports parallel & sequential streams)

Implementing the Class `StreamOfFuturesCollector`

- `StreamOfFuturesCollector` wraps `allOf()` to work with the Java streams framework
- Converts a *stream* of completable futures into a *single* completable future that’s triggered when *all* futures in the stream complete

```java
<<Java Interface>>
Collector<T,A,R>

- supplier():Supplier<A>
- accumulator():BiConsumer<A,T>
- combiner():BinaryOperator<A>
- finisher():Function<A,R>
- characteristics():Set<Characteristics>
```

```java
<<Java Class>>
StreamOfFuturesCollector<T>

- StreamOfFuturesCollector()
- supplier():Supplier<List<CompletableFuture<T>>>
- accumulator():BiConsumer<List<CompletableFuture<T>>,CompletableFuture<T>>
- combiner():BinaryOperator<List<CompletableFuture<T>>>
- finisher():Function<List<CompletableFuture<T>>,CompletableFuture<Stream<T>>>
- characteristics():Set
- toFuture():Collector<CompletableFuture<T>,?,CompletableFuture<Stream<T>>>
```
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector wraps allOf() to work with the Java streams framework
- Converts a stream of completable futures into a single completable future that’s triggered when all futures in the stream complete
- Implements the Collector interface

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector wraps allOf() to work with the Java streams framework
- Converts a stream of completable futures into a single completable future that’s triggered when all futures in the stream complete
- Implements the Collector interface

A collector accumulates input stream elements into a mutable result container
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector wraps allOf() to work with the Java streams framework.

StreamOfFuturesCollector provides a powerful wrapper for some complex code!
Implementing the Class `StreamOfFuturesCollector`

- `StreamOfFuturesCollector` implements all methods in the `Collector` interface.

```java
public class StreamOfFuturesCollector<T>
    implements Collector<CompletableFuture<T>,
                        List<CompletableFuture<T>>,
                        CompletableFuture<Stream<T>>> {
  ...
```

Implements a custom collector

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html)
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> implements Collector<CompletableFuture<T>,
List<CompletableFuture<T>>,
CompletableFuture<Stream<T>>> {
...
```

*The type of input elements in the stream*
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T>
    implements Collector<CompletableFuture<T>,
                      List<CompletableFuture<T>>,
                      CompletableFuture<Stream<T>>> {
    ...
```

The mutable result container type
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T>
    implements Collector<CompletableFuture<T>,
    List<CompletableFuture<T>>,
    CompletableFuture<Stream<T>>> {
    ...
```

The result type of final output of the collector
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T>
    implements Collector<CompletableFuture<T>,
                        List<CompletableFuture<T>>,
                        CompletableFuture<Stream<T>>> {
    ...
```

The Stream<T> parameter differs from the List<T> parameter applied by the previous FuturesCollector
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T>
    implements Collector<CompletableFuture<T>,
                        List<CompletableFuture<T>>,
                        CompletableFuture<Stream<T>>> {
    public Supplier<List<CompletableFuture<T>>> supplier() {
        return ArrayList::new;
    }

    public BiConsumer<List<CompletableFuture<T>>,
                      CompletableFuture<T>> accumulator() {
        return List::add;
    }
...
```

This factory method returns a supplier used by the Java streams collector framework to create a new mutable array list container.
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
class StreamOfFuturesCollector<T> implements Collector<CompletableFuture<T>,
    List<CompletableFuture<T>>,
    CompletableFuture<Stream<T>>> {
    public Supplier<List<CompletableFuture<T>>> supplier() {
        return ArrayList::new;
    }
    public BiConsumer<List<CompletableFuture<T>>,
        CompletableFuture<T>> accumulator() {
        return List::add;
    }
    ...
}
```

This mutable result container stores a list of completable futures of type T
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> implements Collector<CompletableFuture<T>,
    List<CompletableFuture<T>>,
    CompletableFuture<Stream<T>>> {

    public Supplier<List<CompletableFuture<T>>> supplier() {
        return ArrayList::new;
    }

    public BiConsumer<List<CompletableFuture<T>>,
        CompletableFuture<T>> accumulator() {
        return List::add;
    }

    // ...}
```

This factory method returns a bi-consumer used by the Java streams collector framework to add a new completable future into the mutable array list container

This method is only ever called in a single thread (so no locks are needed)
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ... public BinaryOperator<List<CompletableFuture<T>>> combiner() {
        return (List<CompletableFuture<T>> one,
                List<CompletableFuture<T>> another) -> {
            one.addAll(another);
            return one;
        }
    }
    ...}
```

This factory method returns a binary operator that merges two partial
array list results into a single array list (only relevant for parallel streams)

This method is only ever called in a single thread (so no locks are needed)
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0])).thenApply(v -> futures.stream()
            .map(CompletableFuture::join));
    }
    ...
}
```

This factory method returns a function used by the Java streams collector framework to transform the array list mutable result container to the completable future result type.
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0]))
            .thenApply(v -> futures.stream()
                .map(CompletableFuture::join));
    }
    ...
}
```

Reference to the mutable result container, which is an ArrayList
Implementing the Class `StreamOfFuturesCollector`

- `StreamOfFuturesCollector` implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...

    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0]))
            .thenApply(v -> futures.stream()
                .map(CompletableFuture::join));
    }

    ...
```

Convert the list of futures to an array of futures & pass to `allOf()` to obtain a future that will complete when all futures complete.
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0]))
            .thenApply(v -> futures.stream()
                .map(CompletableFuture::join));
    }
    ...
}

When all futures have completed get a single future to a stream of joined elements of type T
```
Implementing the Class `StreamOfFuturesCollector`

- `StreamOfFuturesCollector` implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0])).thenApply(v -> futures.stream().map(CompletableFuture::join));
    }
    ...
}
```

`Convert the array list of futures into a stream of futures`
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0])).thenApply(v -> futures.stream().map(CompletableFuture::join));
    }
    ...

    This call to join() will never block!
```
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...
    public Function<List<CompletableFuture<T>>, CompletableFuture<Stream<T>>> finisher() {
        return futures -> CompletableFuture.allOf(futures.toArray(new CompletableFuture[0])).thenApply(v -> futures.stream().map(CompletableFuture::join));
    }
    ...
```

Return future to stream of elements of T since no terminal operation after map()
Implementing the Class StreamOfFuturesCollector

- toFuture() returns a future to a stream of futures to images that are being downloaded, filtered, & stored

```java
void processStream() {
    List<URL> urls = getInput();

    CompletableFuture<Stream<Image>> resultsFuture = urls
        .stream()
        .map(this::checkUrlCachedAsync)
        .map(this::downloadImageAsync)
        .flatMap(this::applyFiltersAsync)
        .collect(toFuture())
        .thenApply(stream ->
            log(stream.flatMap(Optional::stream), urls.size())
        ).join();
```

Provides a single means to await completion of a set of futures before continuing with the program.
Implementing the Class StreamOfFuturesCollector

- `toFuture()` returns a future to a stream of futures to images that are being downloaded, filtered, & stored

```java
def processStream() {
    List<URL> urls = getInput();
    CompletableFuture<Stream<Image>> resultsFuture = urls
        .stream()
        .map(this::checkUrlCachedAsync)
        .map(this::downloadImageAsync)
        .flatMap(this::applyFiltersAsync)
        .collect(toFuture())
        .thenApply(stream -> log(stream.flatMap(Optional::stream), urls.size()))
}
```

*thenApply() is called only after the future returned from collect() completes*
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
...
    public Set characteristics() { 
        return Collections.singleton(Characteristics.UNORDERED);
    }
}
```

*Returns a set indicating the characteristics of the StreamOfFutureCollector class*

```java
public static <T> Collector<CompletableFuture<T>, ?, CompletableFuture<Stream<T>>>
toFuture() {
    return new StreamOfFuturesCollector<>();
}
```

StreamOfFuturesCollector is thus a *non-concurrent* collector
Implementing the Class StreamOfFuturesCollector

- StreamOfFuturesCollector implements all methods in the Collector interface

```java
public class StreamOfFuturesCollector<T> {
    ...  
    public Set characteristics() {
        return Collections.singleton(Characteristics.UNORDERED);
    }
}
```

This static factory method creates a new StreamOfFuturesCollector

```java
public static <T> Collector<CompletableFuture<T>, ?, CompletableFuture<Stream<T>>> toFuture() {
    return new StreamOfFuturesCollector<>();
}
```
End of Java Completable Futures ImageStreamGang Example: StreamOf FuturesCollector