Java CompletableFutures ImageStreamGang Example: Applying Completion Stage Methods (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

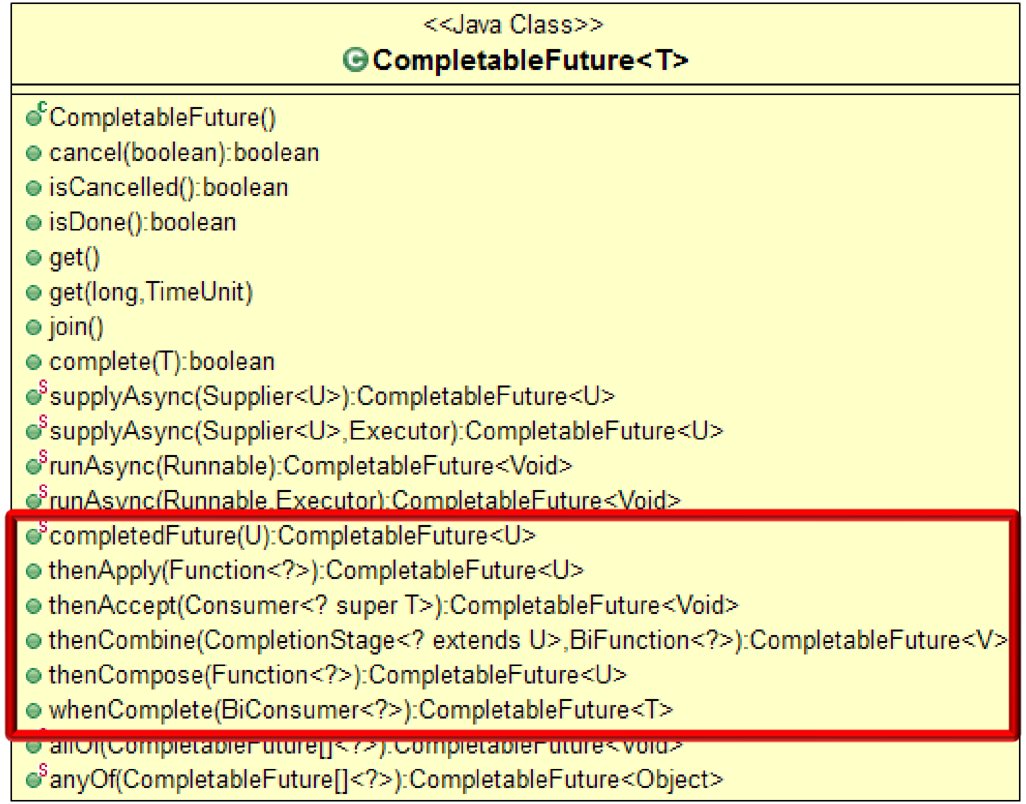
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand 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
    - downloadImageAsync()


<<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>
supplyAsync(Supplier<U>,Executor):CompletableFuture<U>
runAsync(Runnable):CompletableFuture<Void>
runAsync(Runnable,Executor):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>

Applying Completion Stage Methods in DownloadImageAsync

Applying Completion Stage Methods in DownloadImageAsync

- Asynchronously download an image at each given URL

```
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();
```

map() calls the behavior downloadImageAsync()

Applying Completion Stage Methods in DownloadImageAsync

- Asynchronously download an image at each given URL

```
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();
```

Asynchronously downloads an image & stores it in memory

Later behaviors simply ignore "empty" optional images

Applying Completion Stage Methods in DownloadImageAsync

- Asynchronously download an image at each given URL

Returns a stream of futures to optional images, which have a value if the image is being downloaded or are empty if it is already cached

```
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();
```

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                    (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

Asynchronously download image when future completes

See [imagestreamgang/streams/ImageStreamCompletableFuture1.java]

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                      (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

This completion stage method registers an action that's not executed immediately, but only after urlFuture completes

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#thenApplyAsync

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                  (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

If a url is present, then download it when urlFuture completes & return an optional describing the result; otherwise return an empty optional

See docs.oracle.com/javase/8/docs/api/java/util/Optional.html#map

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                  (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

Asynchronously run blockingDownload() if urlOpt is non-empty

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                    (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

Use the common fork-join pool & its ManagedBlocker mechanism

See earlier lesson on "*The Java Fork-Join Pool: The ManagedBlocker Interface*"

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
Image blockingDownload(URL url) {
        return BlockingTask
          .callInManagedBlock(() ->
                                      downloadImage(url));
}
```

Transform a URL into an Image by downloading each image via the URL

See imagestreamgang/streams/ImageStreamGang.java

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
Image blockingDownload(URL url) {
          return BlockingTask
             .callInManagedBlock(() ->
                                 downloadImage(url));
}
```

BlockingTask.callInManagedBlock() wraps the ManagedBlocker interface, which auto-expands the common fork-join pool to handle the blocking image download

See imagestreamgang/utils/BlockingTask.java

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
Image blockingDownload(URL url) {
        return BlockingTask
          .callInManagedBlock(() ->
                                   downloadImage(url));
}
```

This blocking call actually downloads the image at the given url

```
Image downloadImage(URL url) {
   return new Image(url,
                    NetUtils.downloadContent(url));
}
```

See imagestreamgang/streams/ImageStreamGang.java

Applying Completion Stage Methods in DownloadImageAsync

- downloadImageAsync() uses the thenApplyAsync() method internally

```
CompletableFuture<Optional<Image>> downloadImageAsync
                  (CompletableFuture<Optional<URL>> urlFuture) {
  return urlFuture
    .thenApplyAsync(urlOpt ->
                    urlOpt
                    .map(this::blockingDownload),
                    getExecutor());
}
```

Returns a future to an optional image that will complete when the image finishes downloading

End of Applying Completion Stage Methods in Image StreamGang (Part 1)