Java Completable Futures ImageStreamGang Example: Applying Factory Methods

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

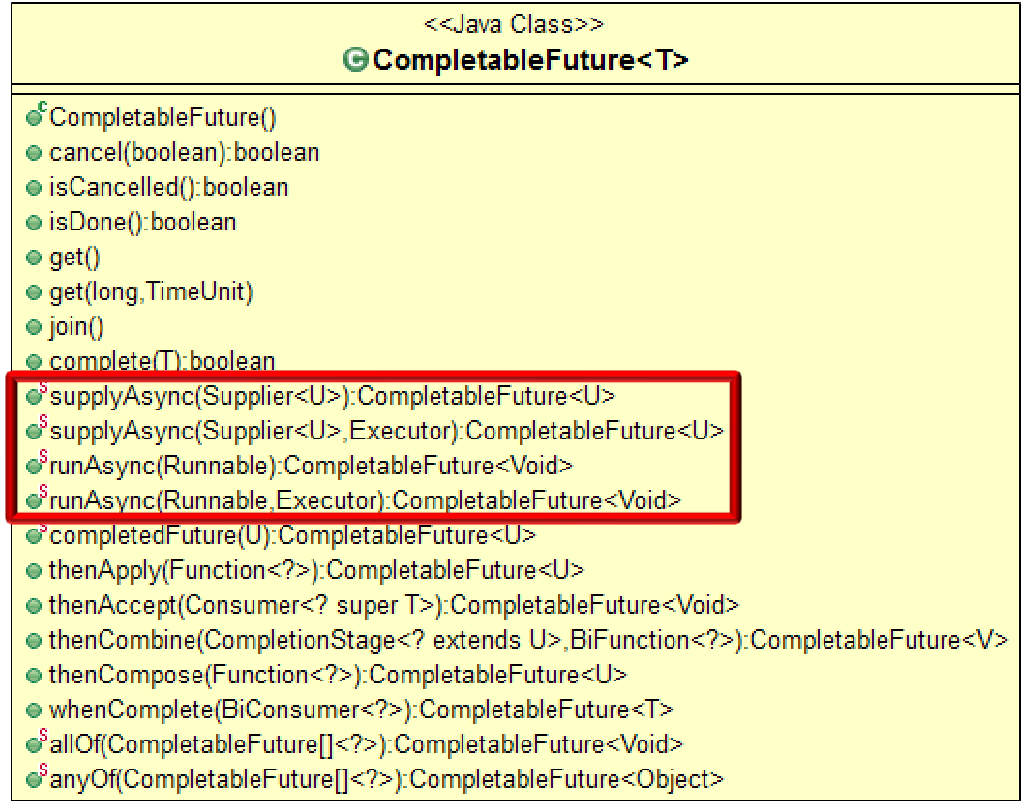
Professor of Computer Science

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
    - supplyAsync()



```
<<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 Factory Methods in ImageStreamGang

Applying Factory Methods in ImageStreamGang

- Initiate an async check to see if images are cached locally

map() calls the behavior checkUrlCachedAsync()

```
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 Factory Methods in ImageStreamGang

- Initiate an async check to see if images are cached locally

Asynchronously check if a URL is already downloaded

```
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 Factory Methods in ImageStreamGang

- Initiate an async check to see if images are cached locally

```
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();
```

Returns a stream of completable futures to optional URLs, which have a value if the URL is not cached or are empty if it is cached

Later behaviors simply ignore "empty" optional URL values

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                 Optional.ofNullable(urlCached(url)
                   ? null
                   : url),
                 getExecutor());
}
```

See imagestreamgang/streams/ImageStreamCompletableFutureBase.java

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                  Optional.ofNullable(urlCached(url)
                    ? null
                    : url),
                  getExecutor());
}
```

This factory method registers an action that runs asynchronously

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

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                 Optional.ofNullable(urlCached(url)
                   ? null
                   : url),
                 getExecutor());
}
```

supplyAsync() runs action in a worker thread from the common fork-join pool

```
void initiateStream() {
  // Set the executor to the common fork-join pool.
  setExecutor(ForkJoinPool.commonPool());
  ...
}
```

See dzone.com/articles/common-fork-join-pool-and-streams

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                 Optional.ofNullable(urlCached(url)
                   ? null
                   : url),
                 getExecutor());
}
```

ofNullable() is a factory method that returns an optional URL, which has a value if the URL is not cached or is empty if it is already cached

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

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                Optional.ofNullable(urlCached(url)
                  ? null
                  : url),
                getExecutor());
}
```

Returns true if the image has already been filtered before

```
boolean urlCached(URL url) {
  return mFilters.stream()
                 .filter(filter -> urlCached(url, filter.getName()))
                 .count() > 0;
}
```

See imagestreamgang/streams/ImageStreamGang.java

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                 Optional.ofNullable(urlCached(url)
                   ? null
                   : url),
                 getExecutor());
}
```

Returns true if image file already exists

```
boolean urlCached(URL url, String filterName) {
  File file = new File(getPath(), filterName);
  File imageFile = new File(file, getNameForUrl(url));
  return !imageFile.createNewFile();
}
```

See imagestreamgang/streams/ImageStreamGang.java

Applying Factory Methods in ImageStreamGang

- checkUrlCachedAsync() uses the supplyAsync() factory method internally

```
CompletableFuture<Optional<URL>> checkUrlCachedAsync(URL url) {
  return CompletableFuture
    .supplyAsync(() ->
                 Optional.ofNullable(urlCached(url)
                   ? null
                   : url),
                 getExecutor());
}
```

```
boolean urlCached(URL url, String filterName) {
  File file = new File(getPath(), filterName);
  File imageFile = new File(file, getNameForUrl(url));
  return !imageFile.createNewFile();
}
```

There are clearly better ways of implementing an image cache!

End of Java Completable Futures ImageStreamGang Example: Applying Factory Methods