Understand the Java CompletableFuture ImageStream

Gang Case Study: 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()`
Applying Factory Methods in ImageStreamGang
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 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();

    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());

    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.

- Initiate an async check to see if images are cached locally.
• checkUrlCachedAsync() uses the supplyAsync() factory method internally

```java
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

```java
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](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

```java
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

```java
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

```java
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](https://docs.oracle.com/javase/8/docs/api/java/util/Optional.html#ofNullable)
checkUrlCachedAsync() uses the supplyAsync() factory method internally

```java
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

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

See [imagestreamgangstreamsImageStreamGang.java](imagestreamgangstreamsImageStreamGang.java)
Applying Factory Methods in ImageStreamGang

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

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

Returns true if image file already exists

```java
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

```java
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 Understand the Java CompletableFuture ImageStreamGang Case Study: Applying Factory Methods