Applying Key Operators in the Parallel Flux Class: Case Study ex5 (Part 2)

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

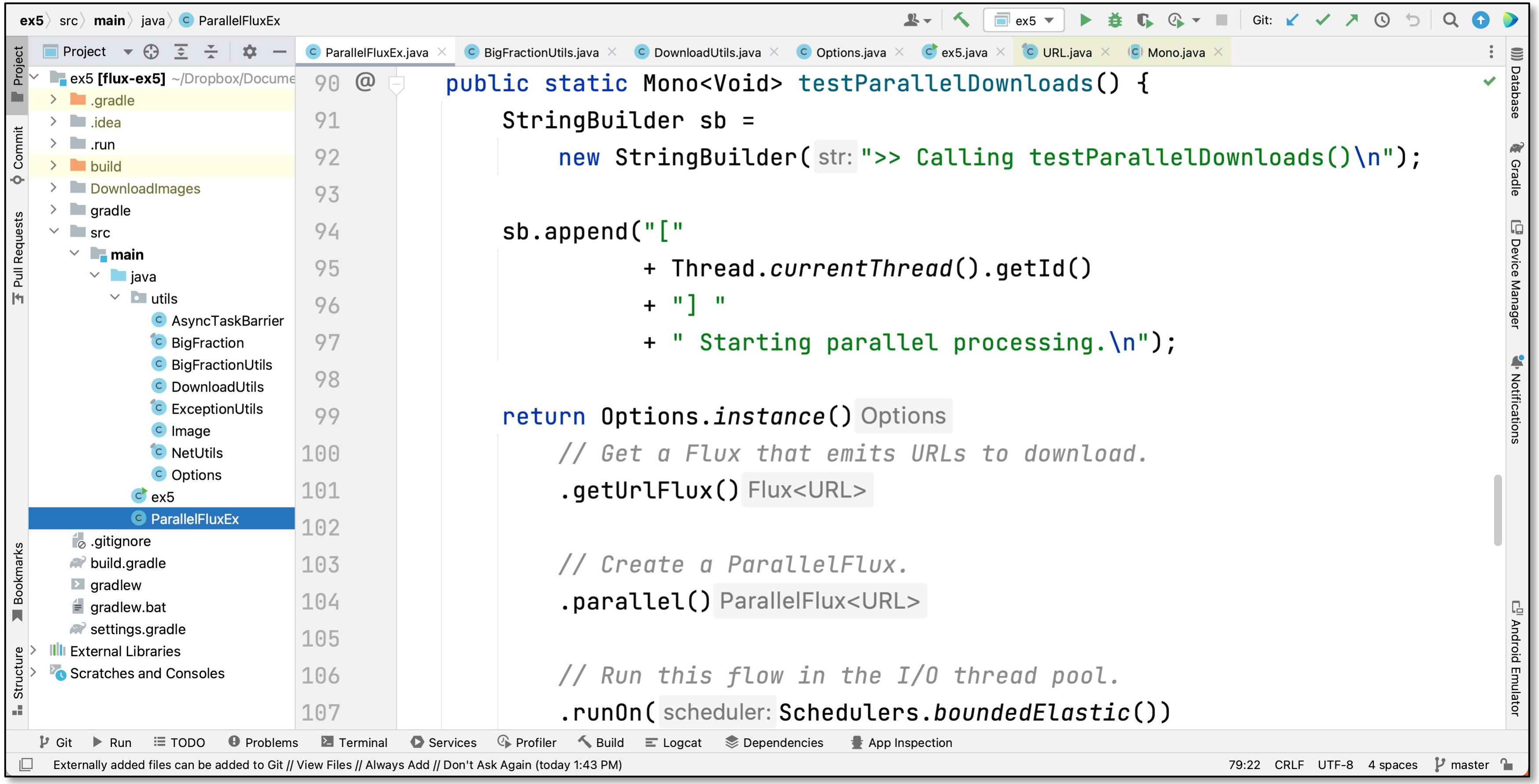
Learning Objectives in this Part of the Lesson

- Part 2 of case study ex5 shows how to apply Project Reactor features to download & store images from remote web servers by showcasing various operators, e.g.
  - Flux operators fromIterator(), parallel(), & collect()
  - ParallelFlux operators runOn(), map(), & sequential()
  - Mono operators doOnSuccess() & then()
  - The Schedulers.boundedElastic() thread pool

```
return Flux
  .fromIterable(getUrlList)

  .parallel()

  .runOn
     (Schedulers
       .boundedElastic())

  .map(downloadAndStoreImage)

  .sequential()

  .collectList()
```

Applying Key Operators in the ParallelFlux Class to ex5

Applying Key Operators in the ParallelFlux Class to ex5



```java
public static Mono<Void> testParallelDownloads() {
    StringBuilder sb =
        new StringBuilder(str: ">> Calling testParallelDownloads()\n");

    sb.append("["
              + Thread.currentThread().getId()
              + "] "
              + " Starting parallel processing.\n");

    return Options.instance() Options
        // Get a Flux that emits URLs to download.
        .getUrlFlux() Flux<URL>

        // Create a ParallelFlux.
        .parallel() ParallelFlux<URL>

        // Run this flow in the I/O thread pool.
        .runOn(scheduler: Schedulers.boundedElastic())
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/flux/ex5

End of Applying Key Operators in the ParallelFlux Class: Case Study ex5 (Part 2)