Apply Java Parallelism to Case Study Apps

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

- Know how Java’s functional programming features are applied in several example case study apps
- Know how Java’s functional programming features are applied in several example case study apps
- These apps are available in open-source form from my GitHub repo!
Summary of Example Case Study Apps
Summary of Example Case Study Apps

- ThreadJoinTest case study shows how Java functional programming features are applied to search for a list of phrases in the complete works of William Shakespeare.

The Complete Works of William Shakespeare

Welcome to the Web's first edition of the Complete Works of William Shakespeare. This site has offered Shakespeare's plays and poetry to the Internet community since 1993.
Summary of Example Case Study Apps

- SimpleSearchStream case study shows how Java sequential streams are applied to find words in song lyrics

```
inputString
    .stream()
    .map(this::searchForWord)
    .filter(not(SearchResults::isEmpty))
    .collect(toList())
```

Let's start at the very beginning...

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream)
Summary of Example Case Study Apps

- SearchStreamGang case study applies Java sequential/parallel streams & regular expressions to find phrases in the complete works of William Shakespeare

Summary of Example Case Study Apps

- ImageCounter case study applies completable futures to recursively crawl web pages & count the # of images in parallel via asynchronous reactive programming.

The root folder can either reside locally (filesystem-based) or be accessed remotely (web-based).

Summary of Example Case Study Apps

- ImageStreamGang shows how the StreamGang framework can be combined with Java streams & completable futures to download, filter, store, & display images in parallel.

```
List of URLs to Download

map(this::checkUrlCachedAsync)

map(this::downloadImageAsync)

flatMap(this::applyFiltersAsync)

collect(toFuture())

thenAccept(this::log)

---

Parallel Streams

filter(not(this::urlCached))

map(this::downloadImage)

flatMap(this::applyFilters)

collect(toList())
```

End of Apply Java Parallelism to Case Study Apps