Motivating the Need for Java 8 Completable Futures (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

• Motivate the need for Java futures
• Motivate the need for Java 8 completable futures
Motivating the Need for Completable Futures
Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures
Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures
  - **Pros**
    - May leverage inherent parallelism more effectively with fewer threads

See Lesson 2.3 on the Java Executor Framework
Motivating the Need for Completable Futures

• Pros & cons of asynchronous calls with Java futures
  
  • **Pros**
    
    • May leverage inherent parallelism more effectively with fewer threads, e.g.,
    
    • Queue async computations for execution in a pool of threads

Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures
  - **Pros**
    - May leverage inherent parallelism more effectively with fewer threads, e.g.,
      - Queue async computations for execution in a pool of threads
    - Automatically tune variable number of threads based on the workload

Motivating the Need for Completable Futures

- **Pros & cons of asynchronous calls with Java futures**
  - **Pros**
    - May leverage inherent parallelism more effectively with fewer threads, e.g.,
      - Queue async computations for execution in a pool of threads
      - Automatically tune variable number of threads based on the workload
      - Queue of futures can be triggered to get the results

See [github.com/douglas craig schmidt/LiveLessons/tree/master/SearchTaskGang](https://github.com/douglas craig schmidt/LiveLessons/tree/master/SearchTaskGang)
Motivating the Need for Completable Futures

- **Pros & cons of asynchronous calls with Java futures**
  - **Pros**
    - May leverage inherent parallelism more effectively with fewer threads
    - Can lock until the result of an async two-way task is available
Motivating the Need for Completable Futures

• Pros & cons of asynchronous calls with Java futures
  • Pros
    • May leverage inherent parallelism more effectively with fewer threads
    • Can lock until the result of an async two-way task is available
    • Can be canceled & tested to see if a task is done

<<Java Interface>>

- Future<V>
  - cancel(boolean): boolean
  - isCancelled(): boolean
  - isDone(): boolean
  - get()
  - get(long, TimeUnit)
Motivating the Need for Completable Futures

• Pros & cons of asynchronous calls with Java futures
  • Pros
  • Cons
    • Limited feature set
Motivating the Need for Completable Futures

Pros & cons of asynchronous calls with Java futures

Pros

Cons

• Limited feature set
  • Cannot be completed explicitly
    • e.g., additional mechanisms like FutureTask are needed

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/FutureTask.html
Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures
  - **Pros**
  - **Cons**
    - Limited feature set
      - *Cannot* be completed explicitly
      - *Cannot* be chained together fluently to handle async results
Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures

- **Pros**

- **Cons**

  - Limited feature set
    - *Cannot* be completed explicitly
    - *Cannot* be chained together fluently to handle async results
    - *Cannot* be triggered reactively/efficiently as a *collection* of futures w/out undue overhead
Motivating the Need for Completable Futures

- Pros & cons of asynchronous calls with Java futures

  - **Pros**
  
  - **Cons**
    - Limited feature set
      - *Cannot* be completed explicitly
      - *Cannot* be chained together fluently to handle async results
      - *Cannot* be triggered reactively/efficiently as a collection of futures w/out undue overhead

It’s awkward & inefficient to try & “compose” multiple futures
End of Motivating the Need for Java 8 Completable Futures (Part 2)