Reactive Programming & Java 8
Completable Futures

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 Lesson

• Understand the relationship between reactive programming & Java 8 completable futures
Reactive Programming &
Java 8 Completable Futures
Reactive Programming & Java 8 Completable Futures

- Reactive programming is an asynchronous programming paradigm concerned with processing data streams & propagation of changes

See [en.wikipedia.org/wiki/Reactive_programming](en.wikipedia.org/wiki/Reactive_programming)
Reactive Programming & Java 8 Completable Futures

- Reactive programming is based on four key principles

- Responsive
- Resilient
- Elastic
- Message-driven

See [www.reactivemanifesto.org](http://www.reactivemanifesto.org)
Reactive Programming & Java 8 Completable Futures

- Reactive programming is based on four key principles, e.g.
  - **Responsive**
    - Provide rapid & consistent response times

*Establish reliable upper bounds to deliver consistent quality of service & prevent delays*

See [en.wikipedia.org/wiki/Responsiveness](en.wikipedia.org/wiki/Responsiveness)
Reactive Programming & Java 8 Completable Futures

- Reactive programming is based on four key principles, e.g.
  - Responsive
  - Resilient
    - The system remains responsive, even in the face of failure

*Failure of some operations should not bring the entire system down*

See [en.wikipedia.org/wiki/Resilience_(network)](en.wikipedia.org/wiki/Resilience_(network))
Reactive Programming & Java 8 Completable Futures

- Reactive programming is based on four key principles, e.g.
  - Responsive
  - Resilient
  - Elastic
    - A system should remain responsive, even under varying workload

It should be possible to "auto-scale" performance

See [en.wikipedia.org/wiki/Autoscaling](en.wikipedia.org/wiki/Autoscaling)
Reactive Programming & Java 8 Completable Futures

- Reactive programming is based on four key principles, e.g.
  - Responsive
  - Resilient
  - Elastic
  - Message-driven

This principle is an “implementation detail” wrt the others..

- Asynchronous message-passing ensures loose coupling, isolation, & location transparency between components

See en.wikipedia.org/wiki/Message-oriented_middleware
Reactive Programming & Java 8 Completable Futures

Java 8 completable futures map onto key reactive programming principles, e.g.

**Responsive**
- Avoid blocking in user code
- Blocking underutilizes cores, impedes inherent parallelism, & complicates program structure

CALLER

```
searchForWord_1
```

```
return result_1
```

CALLEE

```
searchForWord_2
```

```
return result_2
```

```
searchForWord_3
```

```
return return_3
```

Reactive Programming & Java 8 Completable Futures

• Java 8 completable futures map onto key reactive programming principles, e.g.

  • **Responsive**
    • Avoid blocking in user code
    • Blocking underutilizes cores, impedes inherent parallelism, & complicates program structure

  **Factory, completion stage, & arbitrary-arity methods avoid blocking threads**

**Completion stage methods**

**Factory methods**

**Arbitrary-arity methods**

**Basic methods**
Reactive Programming & Java 8 Completable Futures

- Java 8 completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
    - Avoid blocking in user code
    - Avoid changing threads
    - Incurs excessive overhead wrt synchronization, context switching, & memory/cache management

See [gee.cs.oswego.edu/dl/papers/fj.pdf](gee.cs.oswego.edu/dl/papers/fj.pdf)
Java 8 completable futures map onto key reactive programming principles, e.g.:

- **Responsive**
  - Avoid blocking in user code
  - Avoid changing threads
  - Incurs excessive overhead wrt synchronization, context switching, & memory/cache management

See gee.cs.oswego.edu/dl/papers/fj.pdf
Reactive Programming & Java 8 Completable Futures

- Java 8 completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
  - **Resilient**
    - Exception methods help systems be resilient to crippling failures

However, completable futures are localized to a single process, *not* a cluster!
Java 8 completable futures map onto key reactive programming principles, e.g.:

- **Responsive**
- **Resilient**
- **Elastic**
  - Async computations can run scalably in a pool of threads

Can be a fork-join pool or a custom thread pool

Java 8 completable futures map onto key reactive programming principles, e.g.:

- Responsive
- Resilient
- Elastic
- Message-driven

The Java fork-join pool passes messages between threads in the pool internally.

See [en.wikipedia.org/wiki/Work_stealing](en.wikipedia.org/wiki/Work_stealing)
Java 9 support reactive programming via "Reactive Streams" & the Flow API

Class Flow

java.lang.Object
da.util.concurrent.Flow

public final class Flow
extends Object

Interrelated interfaces and static methods for establishing flow-controlled components in which Publishers produce items consumed by one or more Subscribers, each managed by a Subscription.

These interfaces correspond to the reactive-streams specification. They apply in both concurrent and distributed asynchronous settings: All (seven) methods are defined in void "one-way" message style. Communication relies on a simple form of flow control (method Flow.Subscription.request(long)) that can be used to avoid resource management problems that may otherwise occur in "push" based systems.

See community.oracle.com/docs/DOC-1006738
Java 9 support reactive programming via “Reactive Streams” & the Flow API

- Adds support for stream-oriented pub/sub patterns

See javasampleapproach.com/java/java-9/java-9-flow-api-example-publisher-and-subscriber
Reactive Programming & Java 8 Completable Futures

- Java 9 support reactive programming via “Reactive Streams” & the Flow API
- Adds support for stream-oriented pub/sub patterns

- Combines two patterns
  - *Iterator*, which applies a pull model where apps pulls items from a source
  - *Observer*, which applies a push model that reacts when item is pushed from a source to a subscriber

See [www.journaldev.com/20723/java-9-reactive-streams](http://www.journaldev.com/20723/java-9-reactive-streams)
Reactive Programming & Java 8 Completable Futures

- Java 9 support reactive programming via “Reactive Streams” & the Flow API
- Adds support for stream-oriented pub/sub patterns
- Combines two patterns
- Intended as an interoperable foundation for other reactive programming frameworks

See www.baeldung.com/java-9-reactive-streams
Comparing reactive programming with other Java programming paradigms

- Reactive Programming
  - Reactive Streams (& Streams + CompletableFutures)
  - Completable Futures

- Streams
- Objects

- Synchronous
- Asynchronous

- Single value
- Multiple values
End of Reactive Programming & Java 8 Completable Futures