Spring 5

Project Reactor

Geoff Filippi / @geofffilippi

DJUG

Geoff Filippi

Senior Architect

DISH Network

- Satellite Pay TV provider
- Sling TV
- Wireless
  - Spectrum
  - Narrow Band Internet of Things (NB-IoT) Network
  - 5G

Formerly:

Oildex

A cloud service company for oil and gas

- 2
  years

Formerly:

Time Warner Cable

- 12
  years

Experience

- Microservices
- Domain-Driven Design
- Event-Sourced Systems
- Security

Experience

- Worked on streaming media (Voice over IP), 6 years
- 5 million phone customers

Experience

- Worked on video and streaming video, 4 years

Projects

twctv.com

- HTML5 Video streaming website
- HTML5 Video streaming Set-Top Box (STB) web application

Oildex Projects

- Rewrite 10+-year-old apps
- Angular 1.4/Angular 2
- Scala/Play microservices

We will cover

- Reactive
- Project Reactor
- Spring Framework 5
- Reactive Streams

Reactive

Reactive Manifesto

- Responsive
- Resilient
- Elastic
- Message Driven

Responsive

- Respond in a timely manner

Resilient

- Stay responsive, even during failure
- Replication
- Isolation
- Delegation
- Contain failures
    - Circuit Breaker

Elastic

- Responsive under varying workload
- Scales resources depending on load
- Cost effective

Message Driven

- Rely on asynchronous message passing
- Loosely-coupled components
- Failures are also messages
- Uses message queues
- Backpressure
- Non-blocking communication

Reactive Concepts

- Synchronous vs. Asynchronous
- Blocking vs. Non-Blocking

Synchronous

- Client makes a request and waits for server to complete
- Server starts a thread per request
- Blocks the thread that it runs on
- Lots of concurrent requests use lots of threads
- Familiar Java APIs

Asynchronous

- Client makes a request and does not wait for server to complete
- Server handles all requests on the same thread
- Does not block the thread that it runs on
- Run in an event loop in a single thread
- New Java APIs

Blocking vs. Non-blocking

Blocking

- Related to synchronous
- Thread execution is postponed
- Caller waits for the resource to become available

Non-blocking

- Related to asynchronous
- Thread execution is not postponed
  - Inform the caller that resource is not immediately available
  - Allows the caller to do other work
- Notify the caller when results are ready

Back Pressure

- Mechanism to prevent a message producer from overwhelming a consumer
- Consumer signals producer to pause
- Should propogate through responsive systems

Async Java Features

`Future`

- Java 7
- `.get()` Blocks
- No methods to combine
- No methods to handle errors

`CompletableFuture`

- Java 8
- `CompletionStage<T>` interface
- Equivalent to JavaScript `Promise`
- Single Value
- Part of `java.util.concurrent`

Project Reactor

Project Reactor

- Reactive Core
- Typed Sequences
- Non-Blocking IO
- Efficient Message Passing
- Async
- Based on Reactive Streams

Mono

- 0 or 1 async result
- Reactive equivalent of `CompletableFuture`

Flux

- Stream of results
- Reactive equivalent of `Stream`

Spring Framework 5

- Project Reactor
- WebFlux

Spring Boot 2

- Spring 5
- `spring-boot-starter-webflux`

WebFlux

- Netty
- No Servlet APIs

Reactive Spring Data

- Cassandra
- MongoDB
- Couchbase
- Redis

Reactive Spring Data

- No JPA
- JDBC is a fully-blocking API
- Asychronous Database Access API (ADBA)
  - Proposal
- Relational Databases are bottlenecks in reactive systems

Spring Boot 2

- `spring-boot-starter-data-mongo-reactive`

Reactive Streams

Reactive Streams

- Java 9
- Standard for Asynchronous Stream Processing
  - `java.util.concurrent.Flow`
- Non-blocking back pressure

Reactive Streams Java Specification

- `Publisher`
- `Subscriber`
- `Subscription`
- `Processor`

Reactive Streams Implementations for Java

- RxJava
- Reactor
- Akka Streams
- Ratpack
- Vert.x

Reactive Streams for Java

- 1.8
  - `org/reactivestreams`
- 1.9
  - `java.util.concurrent.Flow`

Demo

- Spring Initializr
- Josh Long - FluxFlix Service
    - Application

Questions?

References

Reactive References

- The Reactive Manifesto
- Notes on Reactive Programming Part I: The Reactive Landscape
- Notes on Reactive Programming Part II: Writing Some Code
- Advanced Reactive Java - Operator-fusion (Part 1)
- Understanding Reactive types
- Design Principles behind Akka Streams

Spring Boot 2 References

- Spring Boot 2
- What's new in Spring Boot 2
- Josh Long - Flux-Flix Service
- Spring Initializr

Spring 5 References

- Spring Framework 5
- Spring WebFlux
- WebFlux framework
- Reactive Spring 5 and Application Design Impact
- SampleWebFluxApplication
- Reacting to Spring Framework 5.0
- Going reactive with Spring Data
- Spring Messaging with RabbitMQ
- Doing Reactive Programming with Spring 5

Spring Integrations References

- Reactive Streams With Spring Data and MongoDB
- MongoDB Reactive Streams Java Driver
- Project Kafka Reference Guide
- Reactor Kafka Reference Guide
- Kafka 1.0 Documentation
- Reactive Kafka
- RabbitMQ
- RabbitMQ Node Tutorial
- Reactor RabbitMQ Reference Guide
- Spring Cloud Stream Reference Guide

Project Reactor References (Continued)

- Project Reactor
- Intro To Reactor Core
- Reactor by Example
- Flux
- Mono
- Reactor 3 Reference Guide
- SampleConsumer
- Fetch first element which matches criteria

Reactive Streams References

- What Are Reactive Streams in Java?
- Reactive Streams
- Java 9 Reactive Streams

Videos

- DevOneConf 2018 - Juergen Hoeller and Josh Long - Reactive Spring
- Under the Hood of Reactive Data Access - Mark Paluch
- Webinar: Upgrading to Spring Boot 2.0 - Phil Webb
- Spring 5 Playlist