Build streaming integrations with relational databases using the JDBC Alpakka Connector

IBM Code Tech Talk

February 07, 2018

Build streaming integrations with relational databases using the JDBC Alpakka Connector
More Data

- Rather than acting on data *at rest*, modern software increasingly
  *operates* on data in *near real-time*.
- Shortened *time-frames* for putting changes into *production*
- *New business models* evolve from existing ones
- *New questions* need to be answered by existing applications
More Systems

CENTRALISED MAINFRAME

DECENTRALIZED APPSERVERS

DISTRIBUTED CLOUD
Today’s Reality

MORE USERS + MORE DATA + MORE SYSTEMS = MORE CONNECTIONS
Scaling the most common programming task…
...to work in a data intensive, world.
Stream all the things!

map, flatMap, filter, groupBy
Plus Flow control

- Data stream
  - Data Protection
  - Validate
  - Enrich
  - Storage

- Data
  - Back Pressure

- Audit
  - Overflow Protection

- Audit
  - Enrich

- External Data Source
Reactive Streams to the rescue

Reactive Streams is an initiative to provide a standard for asynchronous stream processing with non-blocking back pressure. This encompasses efforts aimed at runtime environments (JVM and JavaScript) as well as network protocols.

http://www.reactive-streams.org
Towards reusable shapes for integration

Source<Integer> source = Source.range(0, 20000000);

Flow<Integer, String> flow = Flow.fromFunction(
    (Integer n) -> n.toString());

Sink<String> sink = Sink.foreach(str ->
    System.out.println(str));
Reactive Enterprise Integration - Alpakka Project

- AMQP Connector
- Apache Geode connector
- AWS DynamoDB Connector
- AWS Kinesis Connector
- AWS Lambda Connector
- AWS S3 Connector
- AWS SNS Connector
- AWS SQS Connector
- Azure Storage Queue Connector
- Cassandra Connector
- Elasticsearch Connector
- File Connectors
- FTP Connector
- Google Cloud Pub/Sub
- HBase connector
- IronMQ Connector
- JMS Connector
- MongoDB Connector
- MQTT Connector
- Server-sent Events (SSE) Connector
- Slick (JDBC) Connector
- Spring Web
- Unix Domain Socket Connector
- File IO
- Azure
- Camel
- Eventuate
- FS2
- HTTP Client
- MongoDB
- Kafka
- TCP

https://developer.lightbend.com/docs/alpakka/current/
Moving from ETL/Batch style applications...
...to Reactive Microservices.
An example .. Finally :-D

https://github.com/myfear/alpakka-jdbc
<dependency>
    <groupId>com.lightbend.akka</groupId>
    <artifactId>akka-stream-alpakka-slick_${scala.version}</artifactId>
    <version>${akka.alpakka.version}</version>
</dependency>

<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-stream_${scala.version}</artifactId>
    <version>${akka.version}</version>
</dependency>

<dependency>
    <groupId>com.typesafe.akka</groupId>
    <artifactId>akka-http_${scala.version}</artifactId>
    <version>${akka.http.version}</version>
</dependency>

<scala.version>2.12</scala.version>
<akka.version>2.5.9</akka.version>
<akka.http.version>10.0.10</akka.http.version>
<akka.alpakka.version>0.16</akka.alpakka.version>
mvn compile exec:java

http://localhost:8080/

http://localhost:8080/more

http://localhost:8080/
Code walk through
Next Steps! Play with Akka and Alpakka

Project Site:
http://www.lightbend.com/akka

GitHub Repo:
https://github.com/akka

Documentation:
https://developer.lightbend.com/docs/alpakka/current/

Example:
https://github.com/myfear/alpakka-jdbc
Written for architects and developers that must quickly gain a fundamental understanding of microservice-based architectures, this free O’Reilly report explores the journey from SOA to microservices, discusses approaches to dismantling your monolith, and reviews the key tenets of a Reactive microservice:

- Isolate all the Things
- Act Autonomously
- Do One Thing, and Do It Well
- Own Your State, Exclusively
- Embrace Asynchronous Message-Passing
- Stay Mobile, but Addressable
- Collaborate as Systems to Solve Problems

The detailed example in this report is based on Lagom, a new framework that helps you follow the requirements for building distributed, reactive systems.

- Get an overview of the Reactive Programming model and basic requirements for developing reactive microservices
- Learn how to create base services, expose endpoints, and then connect them with a simple, web-based user interface
- Understand how to deal with persistence, state, and clients
- Use integration technologies to start a successful migration away from legacy systems

• Understand the challenges of starting a greenfield development vs tearing apart an existing brownfield application into services

• Examine your business domain to see if microservices would be a good fit

• Explore best practices for automation, high availability, data separation, and performance

• Align your development teams around business capabilities and responsibilities

• Inspect design patterns such as aggregator, proxy, pipeline, or shared resources to model service interactions
