Event-Driven Microservices with Jakarta EE

David R. Heffelfinger, Ensode Technology, LLC, http://www.ensode.com
@ensode

Ondro Mihályi, Payara Services Ltd, http://www.payara.fish
@omihalyi

About the speakers

- **David R. Heffelfinger**
  - Independent consultant based in Fairfax, VA
  - Java Champion
  - Apache NetBeans committer
  - Author of several books on Java, Java EE and related technologies
  - Named one of 39 Java experts you should follow on Twitter

About the speakers (cont’d)

- **Ondro Mihályi**
  - Java EE / Jakarta EE developer and trainer
  - Core member of Payara and MicroProfile opensource projects
  - Czech JUG leader

Session Outline

- Event-Driven Architecture Design Patterns
- Java Message Service (JMS)
- Message Brokers
- Java Connector Architecture (JCA)
- Demo
- Summary

Event Driven Architecture Design Patterns

- Database per service
- Saga
- Event Sourcing

Database Per Service Pattern

- Microservices need to be loosely coupled
- Not a good idea to share persistent data between microservices
- For a true microservice architecture, each microservice must have its own database
- Database could be a separate instance of an RDBMS or NoSQL database, a separate schema, or private tables in a schema.

Saga Design Pattern

- One database per service, transactions to each database are independent
- Local transactions updating all relevant databases not possible
- Two-phase commit is not possible
- Rollbacks become a challenge

Saga Design Pattern (cont’d)

- Distributed transactions implemented as local transactions for each independent database
- For rollbacks, a series of compensating transactions to undo the initial transactions are initiated

Event Sourcing Design Pattern

- Changes to our application state are stored as a sequence of events
- When using the Saga design pattern, each microservice stores an event into an event or message store

Java Message Service (JMS)

- Standard Java / Jakarta EE API to implement messaging functionality
- Allows applications to interact with messaging systems such as message brokers or Message Oriented Middleware (MOM)
- Keeps our code independent of the underlying message broker

Message Brokers

- There is no requirement to implement microservices as RESTful web services
- Message brokers are common for Event-Driven architectures
- Microservices generating events post messages to a JMS topic
- Microservices needing to react to events listen to the corresponding JMS topics

Message Brokers and JCA

- Typical message brokers include products such as Apache ActiveMQ and IBM Websphere MQ
- Most vendors provide connectors that allow Java code to interact with their product
- These connectors are typically developed via the Java Connector Architecture
- Most vendors provide a JCA Resource Archive (RAR file)

Cloud and Non-Traditional Message Brokers

- Most cloud providers supply their own message broker
  - Amazon Web Services provides Amazon Simple Queue Service
  - Azure provides the Azure Service Bus
- Specialized, Non-Traditional Message Brokers
  - MQ Telemetry Transport (MQTT)
    - Low bandwidth or unreliable networks
  - Apache Kafka
    - High volume of messages

Cloud and Non-Traditional Message Brokers (cont'd)

- Cloud and non-traditional message brokers provide their own, custom Java APIs (not JMS)
- Coding against these APIs ties the Java application to a particular message broker

Payara Cloud Connectors

- Cloud connectors implemented as JCA RAR files
- Can be used to interact with many popular cloud and non-traditional message brokers
  - Apache Kafka
  - Amazon SQS
  - MQTT
  - Azure Service Bus

Payara Cloud Connectors (cont'd)

- Available on Maven central

DEMO

Summary

- Event-Driven microservices help build resilient systems
- Database per service, Saga, and Event Sourcing design patterns help implement event-driven microservices
- Messaging / JMS is typically used to develop event-driven microservices
- Payara Cloud Connectors allow Java applications to use JMS against cloud and non traditional messaging systems

payara®

Additional Resources

- Event Driven Microservices with Payara Micro: https://info.payara.fish/event-driven-microservices-with-payara-micro
- GitHub repository: https://github.com/payara/Payara-Examples

Questions?