The Reactive QuoteServices App Case Study:

Overview

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

• Understand how various concurrency & persistency frameworks are applied in a case study using Spring WebFlux to provide two different quote services

See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3
Overview of the Reactive Quote Services App Case Study
Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices.

Also shows how to use the Eureka discovery service.
Overview of the Reactive QuoteServices App Case Study

• This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices

Microservice-based Quotes App

All microservices register with the Eureka discovery service at startup time

- Gateway
- Handey Application
- Eureka
- Zippy Application

→ → → →
Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices

See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3/client
Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices.

The API gateway receives client HTTP requests & uses the Eureka discovery service to route them to the designated microservice.

Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices.

The Eureka discovery service enables the API gateway to find & communicate with the back-end microservices without hard-coding ports & hostnames.

Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices.

This microservice uses R2DBC to respond with quotes when the API gateway forwards it HTTP requests.

See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3/handeymicroservice
Overview of the Reactive QuoteServices App Case Study

- This case study shows how Spring WebFlux can send/receive HTTP GET/POST requests asynchronously to/from an API gateway & multiple microservices.

      QuoteDriver

      This microservice uses the JPA to respond with quotes when the API gateway forwards it HTTP requests.

      Microservice-based Quotes App

      Gateway
      Handey Application
      Eureka
      Zippy Application

Structure of the Reactive Quote Services App Project
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages

```java
// Code snippets for gateway, common, client, eureka, and zippymicroservice
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3](https://github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3)
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - eureka
    - Contains the “app” entry point for the Eureka discovery service

The QuoteServices App project source code is organized into several modules & packages:

- **eureka**
  - eureka
- **resources**
  - Define the port number listened on by the Eureka discovery service & other properties
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - gateway
    - Contains the “app” entry points & the controller
      - The gateway is largely programmed declaratively

The QuoteServices App project source code is organized into several modules & packages:

- gateway
  - gateway
- resources
  - Specifies the port number exposed by the API gateway &
  - Configures the gateway to use the Eureka discovery service
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - zippymicroservice
    - microservice
      - Contains the “app” entry points & the controller for a JPA database
      - Returns reactive types, however
The QuoteServices App project source code is organized into several modules & packages:

- zippymicroservice
  - microservice
  - repository
  - Implements the JPA database
  - repository
  - Does not return reactive types
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • zippymicroservice
    • microservice
    • repository
  • resources
    • Defines various application properties
      • e.g., microservice name, Eureka client configuration, schema definitions & data for Zippy quotes
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - handeymicroservice
    - microservice
      - Contains the “app” entry points & the controller for an R2DBC database
      - Returns reactive types
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • handeymicroservice
    • microservice
    • repository
      • Implements the R2DBC database repository
      • Returns reactive types
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • handeymicroservice
    • microservice
    • repository
  • resources
    • Defines various application properties
      • e.g., microservice name, Eureka client configuration, schema definitions & data for Handey quotes
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - common
    - common
    - Classes shared by the zippy & handey microservices
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • common
    • common
  • utils
    • Helper classes that are reused by other projects
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • client
    • QuoteDriver
      • This test driver causes the client to asynchronously send/receive requests/responses to/from the microservices running on the server & displays results
Structure of the Reactive QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • client
    • QuoteDriver
  • client
    • Sends HTTP GET/POST requests to the microservices using reactive types
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
  - client
  - common
    - Helper classes that are specific to this client driver
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
  - client
  - common
  - utils
    - Helper classes that are reused by other projects
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
  - client
  - common
  - utils
  - resources
    - Defines various application properties
      - e.g., disable/enable logging & sets the client driver name & port number

```
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
  - client
  - common
  - utils
  - resources
    - Defines various application properties
      - e.g., disable/enable logging & sets the client driver name & port number
```

- Defines various application properties
  - e.g., disable/enable logging & sets the client driver name & port number
End of the Reactive QuoteServices App Case Study: Overview