The 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 WebMVC to provide two different quote services.

See github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex4
Overview of the Quote Services App Case Study
Overview of the QuoteServices App Case Study

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

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

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

All microservices register with the Eureka discovery service at startup time
Overview of the QuoteServices App Case Study

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

Overview of the QuoteServices App Case Study

- This case study shows how Spring WebMVC can send & receive HTTP GET requests synchronously 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.

Microservice-based Quotes App:
- Gateway
- Handey Application
- Eureka
- Zippy Application

See `github.com/douglascraigschmidt/LiveLessons/tree/master/WebMVC/ex4/gateway`
Overview of the QuoteServices App Case Study

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

Microservice-based Quotes App

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

Overview of the QuoteServices App Case Study

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

The microservices respond with quotes when the API gateway forwards them HTTP requests.

See WebMVC/ex4/microservices
Structure of the Quote
Services App Project
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages.
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - eureka
      - Contains the “app” entry point
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - eureka
  - resources
    - Define the port number listened on by the Eureka discovery service & other properties
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - gateway
      - Contains the “app” entry points & the controller
Structure of the QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • main
    • gateway
  • resources
    • Specifies the port numbers & microservices exposed by the API gateway
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - microservices
      - Contains the “app” entry points & the controller
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - microservices
  - common
    - Consolidates various project-specific helper classes
Structure of the QuoteServices App Project

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

  - main
    - microservices
    - common
  - utils
    - Consolidaes various reusable helper classes
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - main
    - microservices
    - common
    - utils
  - resources
    - Defines various application properties
      - e.g., microservice names & port numbers, schema definitions for quotes
Structure of the QuoteServices App Project

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

- **client**
  - QuoteDriver
- **client**
  - Sends HTTP GET requests to the microservices
Structure of the QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - `client`
    - `QuoteDriver`
  - `client`
  - `utils`
    - Consolidates various reusable helper classes
Structure of the QuoteServices App Project

• The QuoteServices App project source code is organized into several modules & packages
  • client
    • QuoteDriver
  • client
  • utils
  • resources
    • Defines various application properties
      • e.g., disable/enable logging & sets the client driver port number
End of the QuoteServices App Case Study: Overview