The QuoteServices App Case Study: Gateway Microservice Structure & Functionality

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 the structure & functionality of the Gateway microservice implementation


This code is almost entirely declarative
Overview of the Spring Cloud API Gateway
Overview of the Spring Cloud API Gateway

• The Spring Cloud API Gateway consists of several components
Overview of the Spring Cloud API Gateway

- The Spring Cloud API Gateway consists of several components

- **Routes**
  - Consists of an ID, destination URI, collection of predicates, & a collection of filters

```yaml
routes:
  - id: handey
    uri: http://localhost:9100
    predicates:
      - Path= /handey/**
    filters:
      - StripPrefix=1 # ...

  - id: zippy
    uri: http://localhost:9101
    predicates:
      - Path= /zippy/**
    filters:
      - StripPrefix=1
```
Overview of the Spring Cloud API Gateway

• The Spring Cloud API Gateway consists of several components

• **Routes**
  • Consists of an ID, destination URI, collection of predicates, & a collection of filters
  • A route is matched if the aggregate predicate is true

```java
routes:
  - id: handey
    uri: http://localhost:9100
    predicates:
      - Path= /handey/**
    filters:
      - StripPrefix=1
  - id: zippy
    uri: http://localhost:9101
    predicates:
      - Path= /zippy/**
    filters:
      - StripPrefix=1
```
Overview of the Spring Cloud API Gateway

• The Spring Cloud API Gateway consists of several components

• **Routes**
  • Consists of an ID, destination URI, collection of predicates, & a collection of filters
  • A route is matched if the aggregate predicate is true
  • Routes can be created either programmatically (using Java) or declaratively (using YAML)

```
routes:
  - id: handey
    uri: http://localhost:9100
    predicates:
      - Path= /handey/**
    filters:
      - StripPrefix=1 # ...

  - id: zippy
    uri: http://localhost:9101
    predicates:
      - Path= /zippy/**
    filters:
      - StripPrefix=1
```
Overview of the Spring Cloud API Gateway

• The Spring Cloud API Gateway consists of several components

  • Routes

  • Predicates
    • Can match HTTP requests
    • e.g., headers, URLs, cookies, or parameters

      routes:
      - id: handey
        uri: http://localhost:9100
        predicates:
          - Path= /handey/**
        filters:
          - StripPrefix=1 # ...
      - id: zippy
        uri: http://localhost:9101
        predicates:
          - Path= /zippy/**
        filters:
          - StripPrefix=1

      Match “handey” or “zippy” route names in the path
Overview of the Spring Cloud API Gateway

- The Spring Cloud API Gateway consists of several components
  - Routes
  - Predicates
  - Filters

- Can modify the request or response as per requirements

```java
routes:
- id: handey
  uri: http://localhost:9100
  predicates:
    - Path= /handey/**
  filters:
    - StripPrefix=1

- id: zippy
  uri: http://localhost:9101
  predicates:
    - Path= /zippy/**
  filters:
    - StripPrefix=1
```

Remove the first path segment (e.g., “handey” or “zippy”) from URI before forwarding the request to the downstream microservice.
Structure & Functionality of the Gateway Microservice
Structure & Functionality of the Gateway Microservice

- The QuoteServices app uses a Spring Cloud API Gateway in conjunction with the Eureka server-side discovery service.
Structure & Functionality of the Gateway Microservice

- This API Gateway is configured largely using declarative YAML files

See [en.wikipedia.org/wiki/YAML](en.wikipedia.org/wiki/YAML)
Structure & Functionality of the Gateway Microservice

- This API Gateway is configured largely using declarative YAML files
  - application.yml

YAML data file configures a Gateway microservice & registers it with Eureka

```yaml
server:
  port: 8080

spring:
  profiles:
    active: path

application:
  name: gateway

... 

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    serviceUrl:
      defaultZone: 
        http://localhost:8761/eureka

See WebMVC/ex4/gateway/src/main/resources/application.yml
```
Structure & Functionality of the Gateway Microservice

- This API Gateway is configured largely using declarative YAML files
  - application.yml
  - application-path.yml

This YAML data file configures the routes to Quotes microservices that are handled automatically by the Gateway microservice.

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: handey
        uri: http://localhost:9100
        predicates:
          - Path= /handey/**
        filters:
          - StripPrefix=1  # ...
      - id: zippy
        uri: http://localhost:9101
        predicates:
          - Path= /zippy/**
        filters:
          - StripPrefix=1  # ...
```

See WebMVC/ex4/gateway/src/main/resources/application-path.yml
Structure & Functionality of the Gateway Microservice

- The Gateway uses the Eureka microservice to locate other microservices it encapsulates by name.
Structure & Functionality of the Gateway Microservice

- The Gateway uses the Eureka microservice to locate other microservices it encapsulates by name.

**Microservice-based Quotes App**

- Gateway
- Handey Application
- Eureka
- Zippy Application

**QuoteDriver**

1. HTTP GET request arrives at the Gateway
Structure & Functionality of the Gateway Microservice

• The Gateway uses the Eureka microservice to locate other microservices it encapsulates by name.

2. It identifies the microservice via the “route” & then uses Eureka to find the designated microservice.
The Gateway uses the Eureka microservice to locate other microservices it encapsulates by name.

3. The Gateway transparently routes the request to the designated microservice.
Structure & Functionality of the Gateway Microservice

- The Gateway uses the Eureka microservice to locate other microservices it encapsulates by name

4. The designated microservice’s controller & service perform the request & return the result back to the client (bypassing the Gateway)
End of the QuoteServices App Case Study: Gateway MicroService Structure & Functionality