The QuoteServices App Case Study: Eureka Microservice Structure & Functionality

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

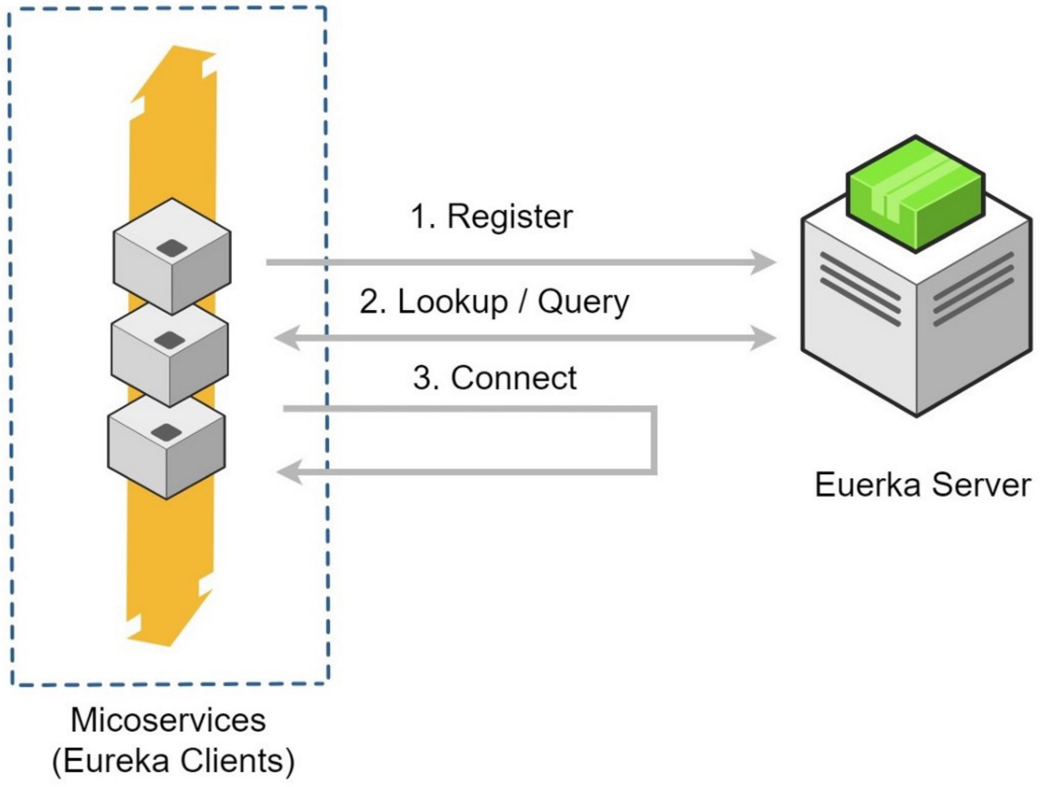
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

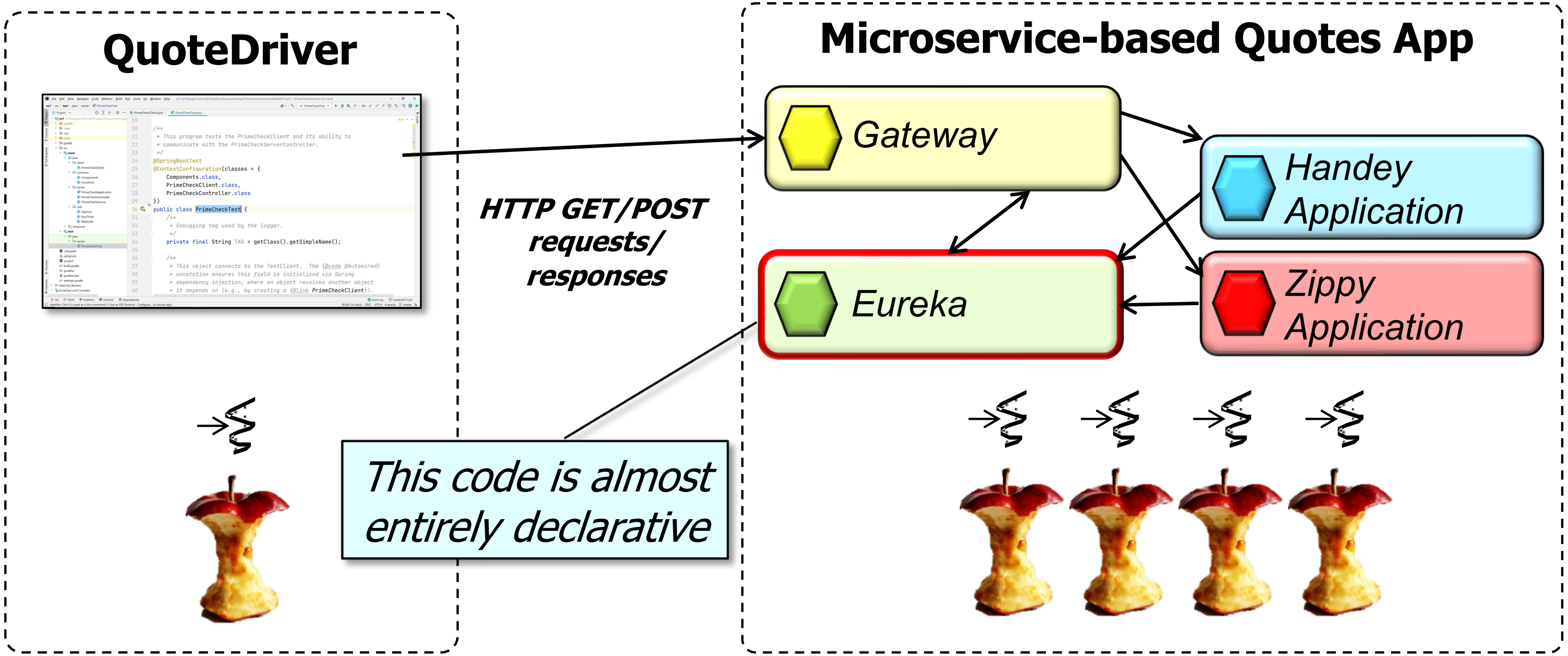
- Understand the structure & functionality of the Eureka discovery service



See www.baeldung.com/spring-cloud-netflix-eureka

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the Eureka discovery service & its microservice implementation in the QuoteServices app


QuoteDriver
Microservice-based Quotes App
Gateway
Handey Application
Eureka
Zippy Application
HTTP GET/POST requests/ responses
This code is almost entirely declarative

```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * Debugging tag used by the logger.
     */
    private final String TAG = getClass().getSimpleName();

    /**
     * This object connects to the TestClient.  The {@code @Autowired}
     * annotation ensures this field is initialized via Spring
     * dependency injection, where an object receives another object
     * it depends on (e.g., by creating a {@link PrimeCheckClient}).
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3/eureka

Overview of the Eureka Discovery Service

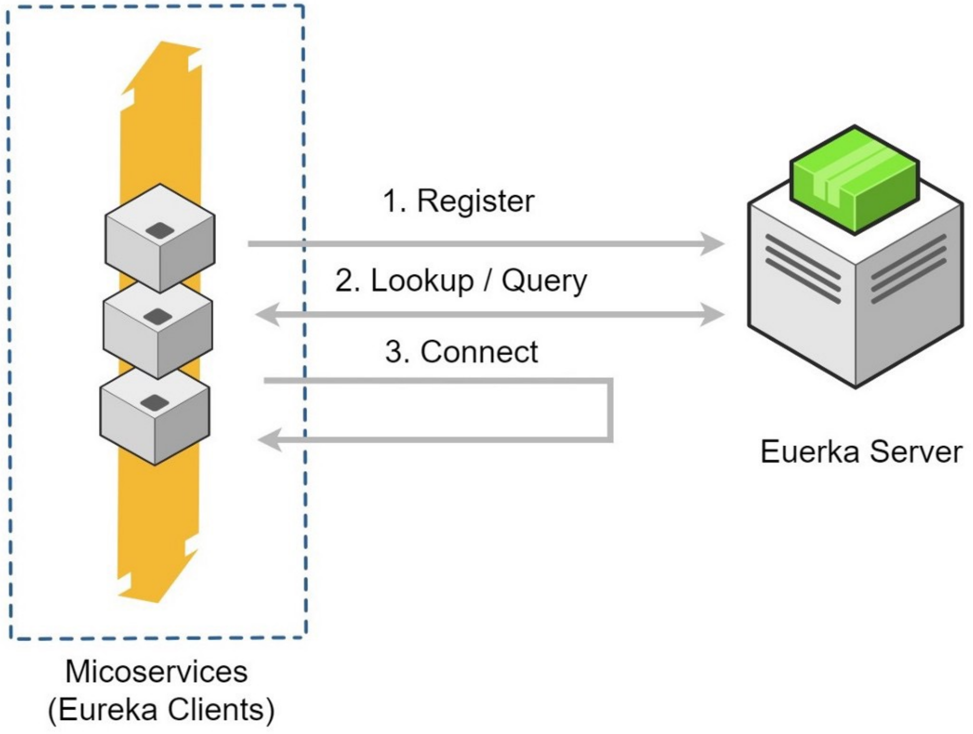
Overview of the Eureka Discovery Service

- Eureka is a discovery service

See en.wikipedia.org/wiki/Eureka_(word)

Overview of the Eureka Discovery Service

- Eureka is a discovery service
  - It enables an API gateway to find & communicate with back -end microservices



See www.baeldung.com/spring-cloud-netflix-eureka

Overview of the Eureka Discovery Service

- Eureka is a discovery service
  - It enables an API gateway to find & communicate with back -end microservices
    - *Without* hard-coding ports & hostnames

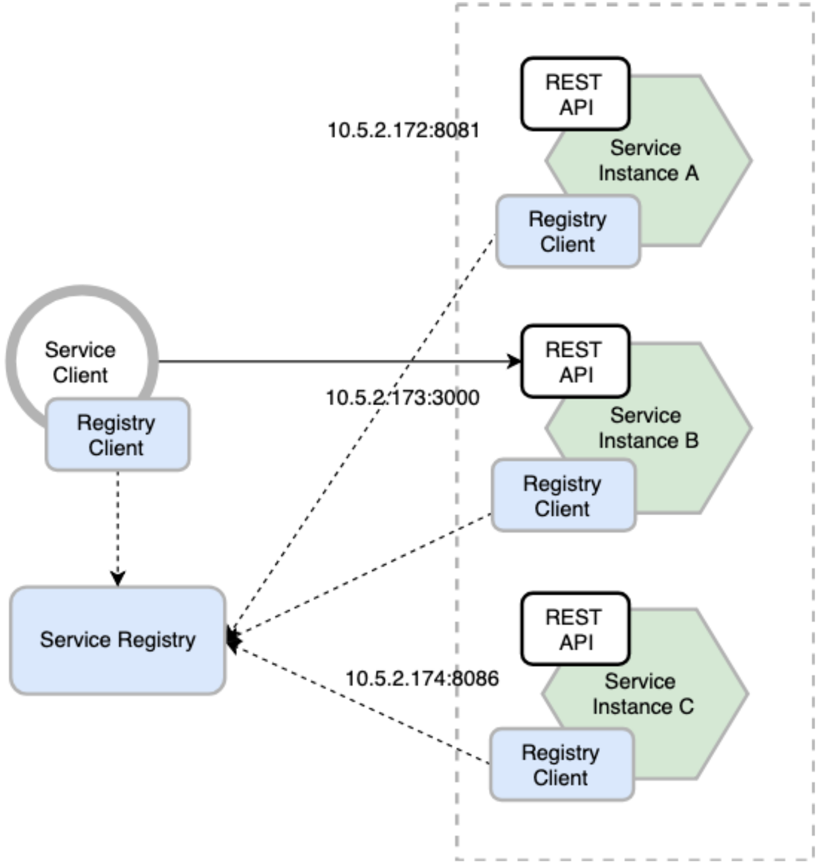
See www.dev-garden.org/2011/08/20/six-things-you-should-never-hardcode

Overview of the Eureka Discovery Service

- A Eureka microservice must start prior to microservices that use it

Overview of the Eureka Discovery Service

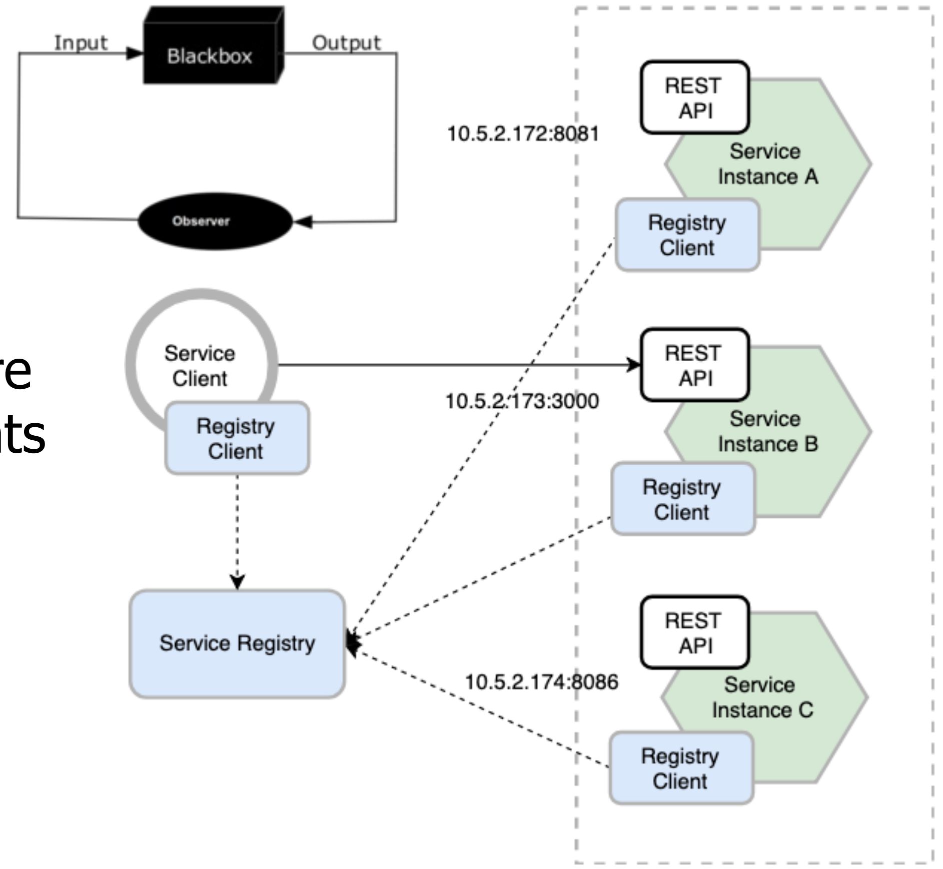
- A Eureka microservice must start prior to microservices that use it
  - It implements a "server-side" discovery pattern



See microservices.io/patterns/server-side-discovery.html

Overview of the Eureka Discovery Service

- A Eureka microservice must start prior to microservices that use it
  - It implements a "server-side" discovery pattern
    - i.e., clients need not be aware how back-end microservices are deployed in server environments

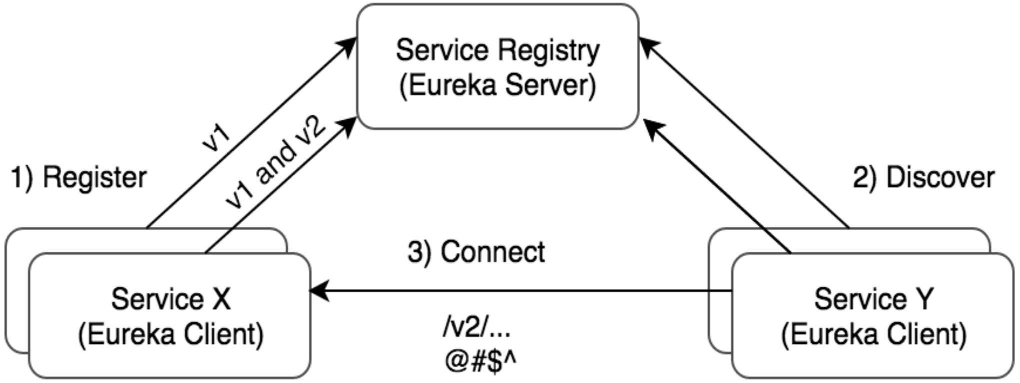


See microservices.io/patterns/server-side-discovery.html

Structure & Functionality of the Eureka Microservice

Structure & Functionality of the Eureka Microservice

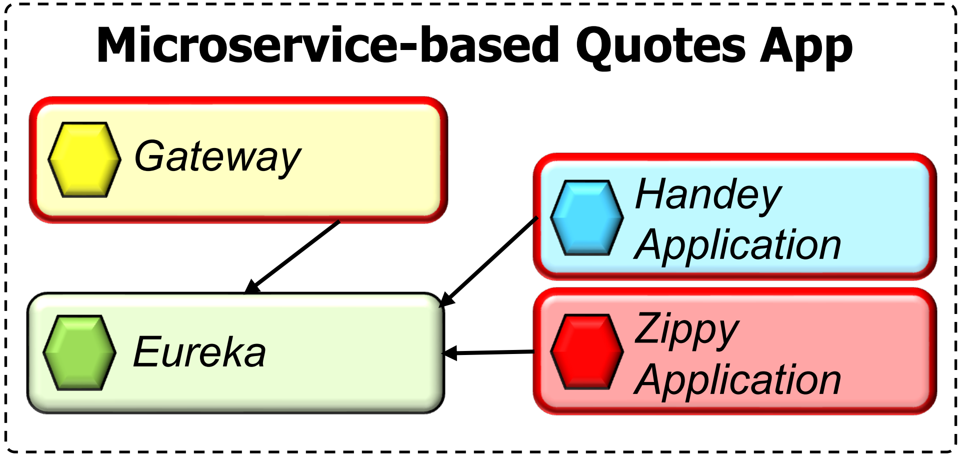
- Back-end microservices interact with the Eureka service registry when they start up



See microservices.io/patterns/service-registry.html

Structure & Functionality of the Eureka Microservice

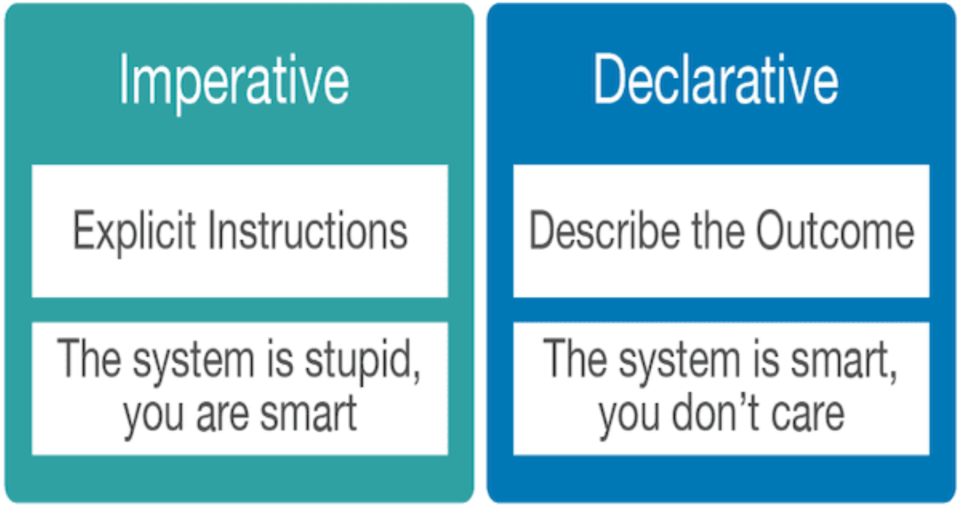
- Back-end microservices interact with the Eureka service registry when they start up
  - e.g., the Gateway, Handey, & Zippy microservices all interact with the Eureka microservice when launched



See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3

Structure & Functionality of the Eureka Microservice

- The QuotesServices Eureka micro-service is configured declaratively


Imperative
Explicit Instructions
The system is stupid, you are smart
Declarative
Describe the Outcome
The system is smart, you don't care

See www.educative.io/blog/declarative-vs-imperative-programming

Structure & Functionality of the Eureka Microservice

- The QuotesServices Eureka micro-service is configured declaratively, e.g.,
  - Via Spring annotations

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication
{ ... }
```

This annotation makes a Spring Boot app act as a Eureka Server

See WebFlux/ex3/eureka/src/main/java/edu/vandy/quoteservices/eureka/EurekaApplication.java

Structure & Functionality of the Eureka Microservice

- The QuotesServices Eureka micro-service is configured declaratively, e.g.,
  - Via Spring annotations
  - Via YAML property files

```
server:
  port: 8761
spring:
  application:
    name: eureka

eureka:
  instance:
    hostname: localhost
    prefer-ip-address: true
  client:
    register-with-eureka: false
    fetch-registry: false
    serviceUrl:
      defaultZone: ...
```

This YAML data file configures a Eureka server microservice that listens on port 8791

See WebFlux/ex3/eureka/src/main/resources/application.yml

Structure & Functionality of the Eureka Microservice

- The QuotesServices Eureka micro-service is configured declaratively, e.g.,
  - Via Spring annotations
  - Via YAML property files

```
server:
  port: 8761
spring:
  application:
    name: eureka

eureka:
  instance:
    hostname: localhost
    prefer-ip-address: true
  client:
    register-with-eureka: false
    fetch-registry: false
    serviceUrl:
      defaultZone: ...
```

Configure this version of Eureka to run on the localhost & use IP addresses instead of host names

Structure & Functionality of the Eureka Microservice

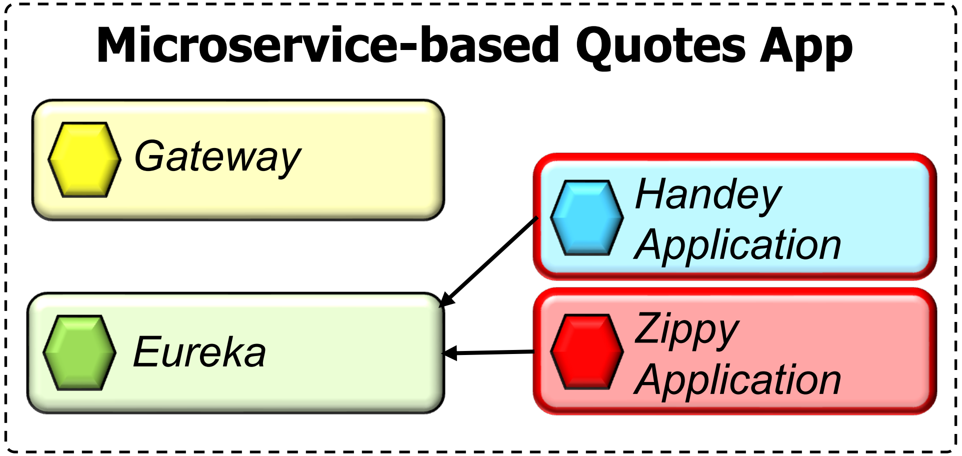
- The QuotesServices Eureka micro-service is configured declaratively, e.g.,
  - Via Spring annotations
  - Via YAML property files

```
server:
  port: 8761
spring:
  application:
    name: eureka

eureka:
  instance:
    hostname: localhost
    prefer-ip-address: true
  client:
    register-with-eureka: false
    fetch-registry: false
    serviceUrl:
      defaultZone: ...
```

Don't register this version of Eureka with any other versions

Structure & Functionality of the Eureka Microservice

- Zippy & Handey microservices also use YAML property files to register themselves with Eureka



Structure & Functionality of the Eureka Microservice

- Zippy & Handey microservices also use YAML property files to register themselves with Eureka, e.g.,

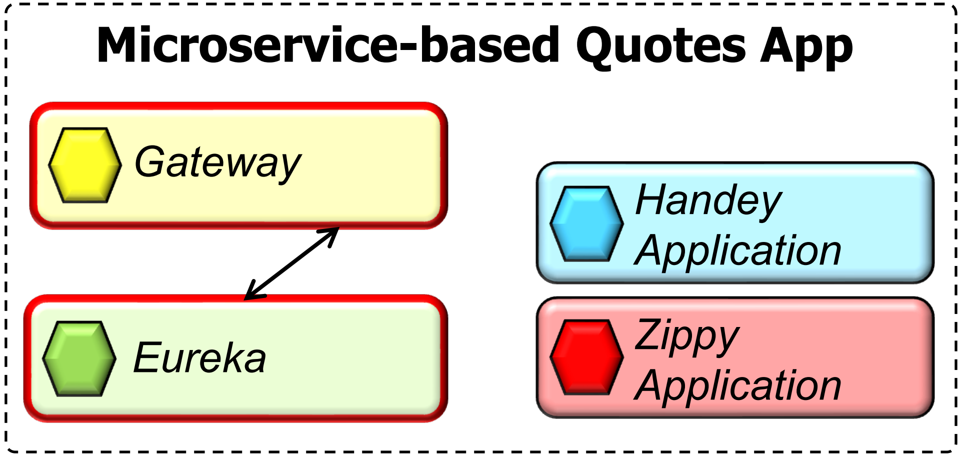
```
# Zippy microservice profile.
server:
  port: 0

# Eureka client properties
eureka:
  client:
    enabled: true

spring:
  application:
    name: zippy
```

This YAML file registers the Zippy microservice with Eureka

See [WebFlux/ex3/zippymicroservice/src/main/resources/application.yml]()

Structure & Functionality of the Eureka Microservice

- The QuoteService API gateway uses Eureka to locate other microservices it encapsulates by name



Structure & Functionality of the Eureka Microservice

- The QuoteService API gateway uses Eureka to locate other microservices it encapsulates by name

```
spring:
  application:
    name: gateway
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          ...
eureka:
  client:
    enabled: true
    register-with-eureka: false
    fetch-registry: false
    serviceUrl: ...
```

This YAML file connects the API Gateway with a discovery service so it automatically creates routes

See WebFlux/ex3/gateway/src/main/resources/application.yml

Structure & Functionality of the Eureka Microservice

- The QuoteService API gateway uses Eureka to locate other microservices it encapsulates by name

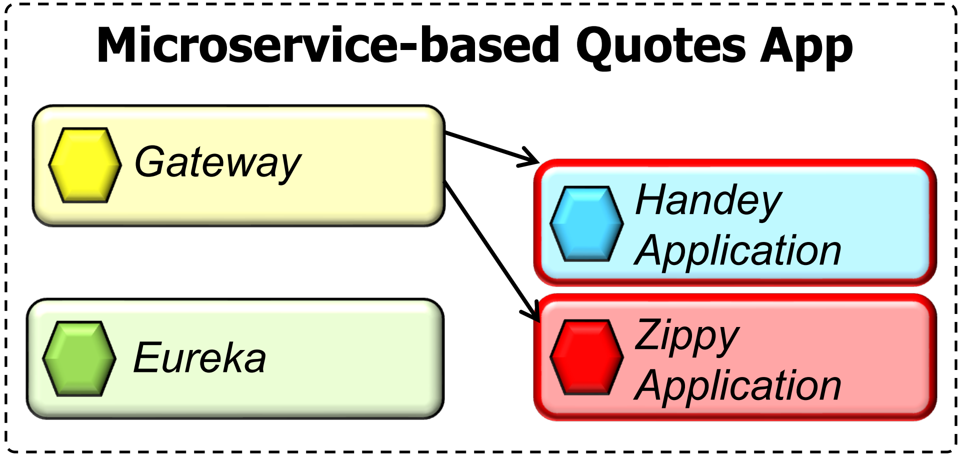
```
spring:
  application:
    name: gateway
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          ...
eureka:
  client:
    enabled: true
    register-with-eureka: false
    fetch-registry: false
    serviceUrl: ...
```

The API Gateway acts as a consumer of the Eureka registry & can discover other registered microservices but doesn't participate in the registry itself

See WebFlux/ex3/gateway/src/main/resources/application.yml

Structure & Functionality of the Eureka Microservice

- The QuoteService API gateway uses Eureka to locate other microservices it encapsulates by name
  - It then transparently forwards HTTP requests to the designated microservice



End of the QuoteServices App Case Study: Eureka MicroService Structure & Functionality