The Reactive QuoteServices App Case Study: Overview

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

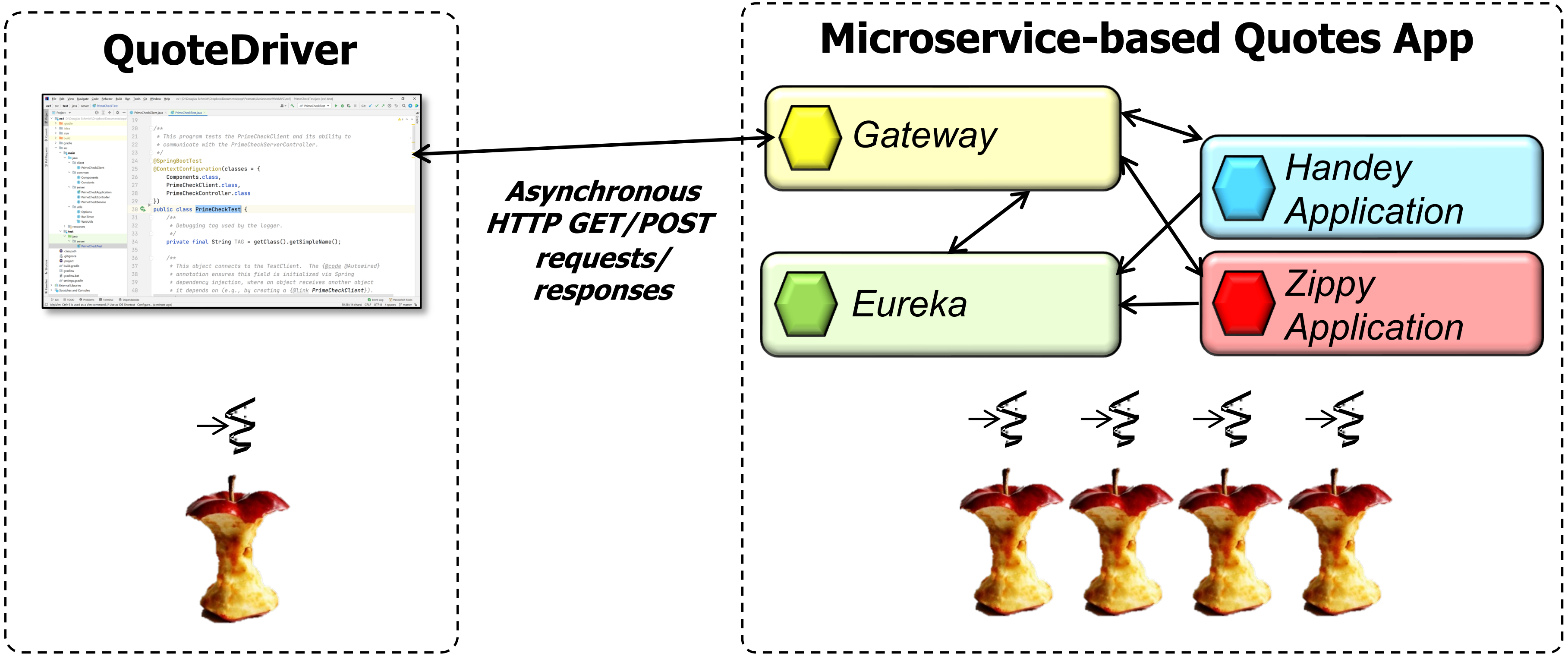
Professor of Computer Science

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


QuoteDriver
Asynchronous HTTP GET/POST requests/ responses
Microservice-based Quotes App
Gateway
Handey Application
Eureka
Zippy Application

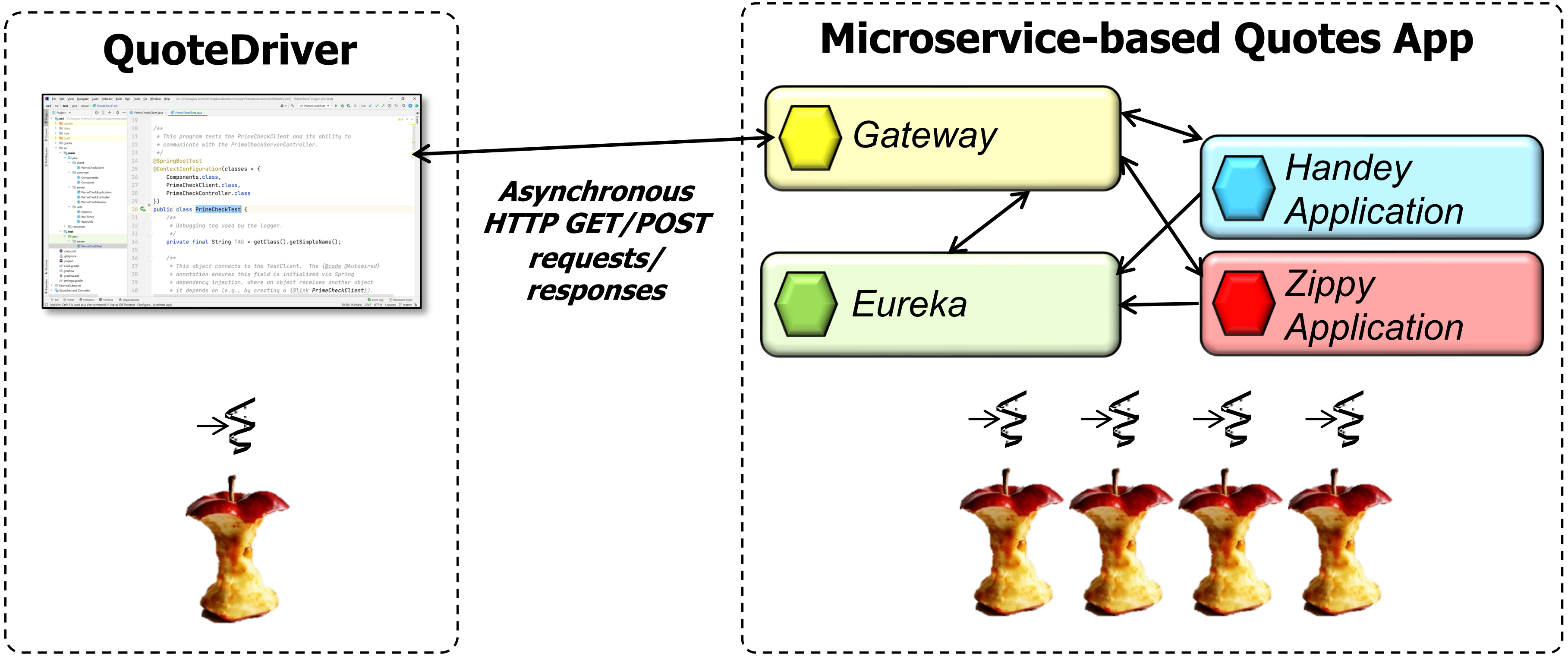
```java

/**
 * 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

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


QuoteDriver
Asynchronous HTTP GET/POST requests/ responses
Microservice-based Quotes App
Gateway
Handey Application
Eureka
Zippy Application

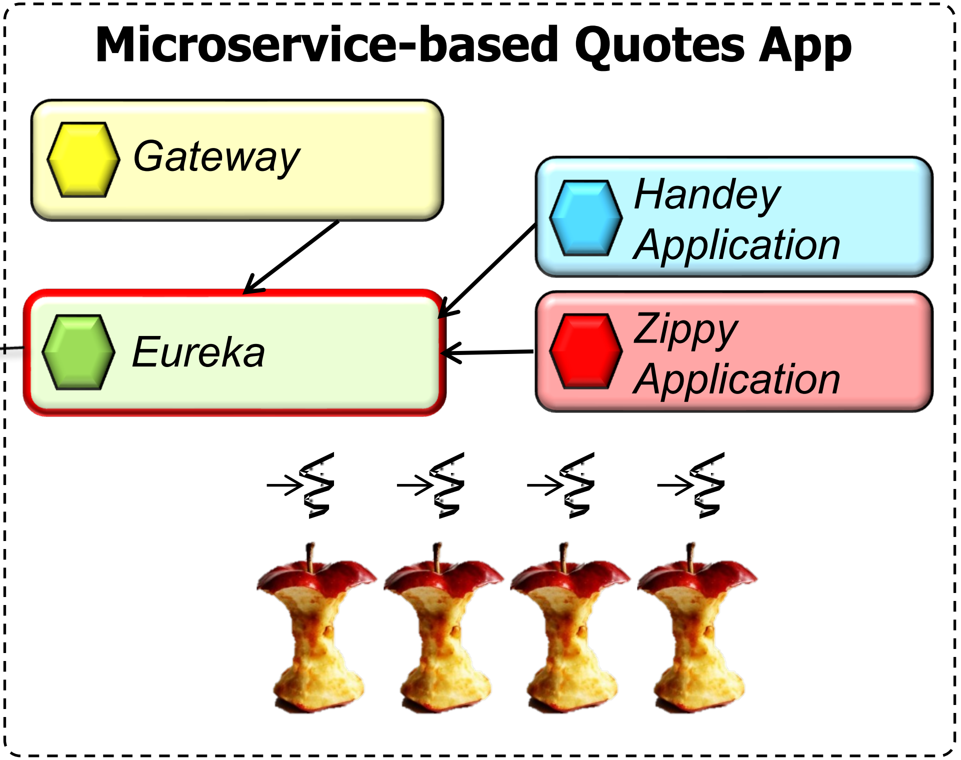
```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * 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}).
```

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

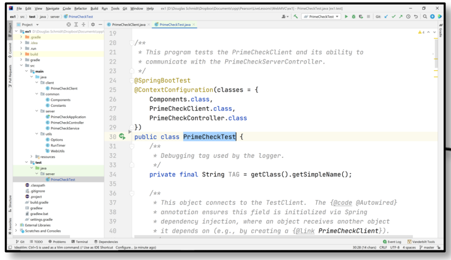
All microservices register with the Eureka discovery service at startup time



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



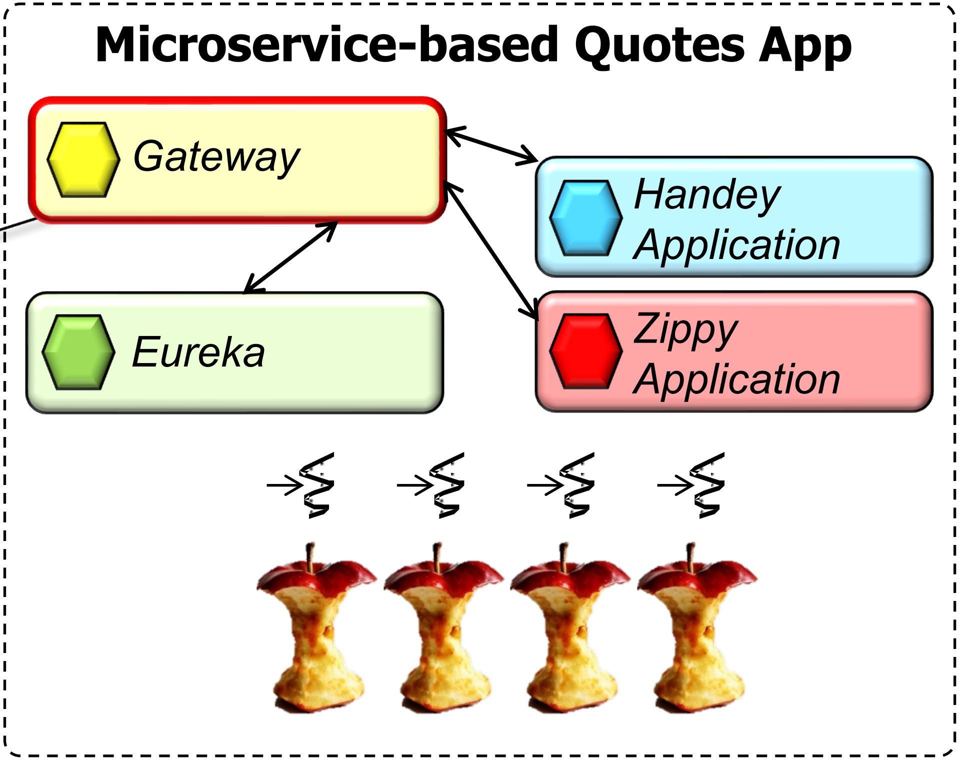
```
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * 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}).
```

The client sends requests to the API gateway (& only the API gateway)

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



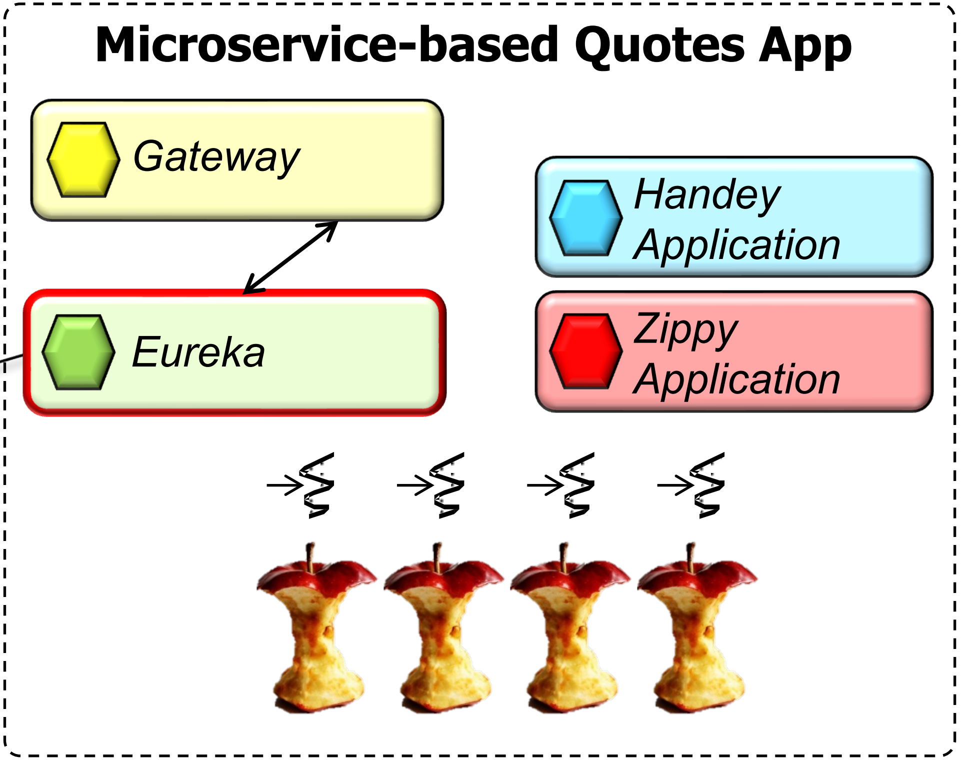
The API gateway receives client HTTP requests & uses the Eureka discovery service to route them to the designated microservice

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

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

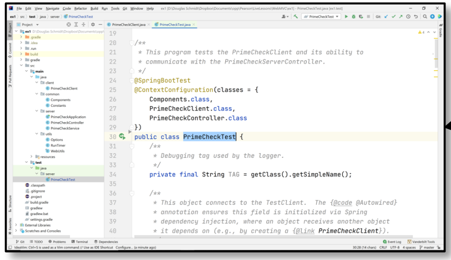


See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3/eureka

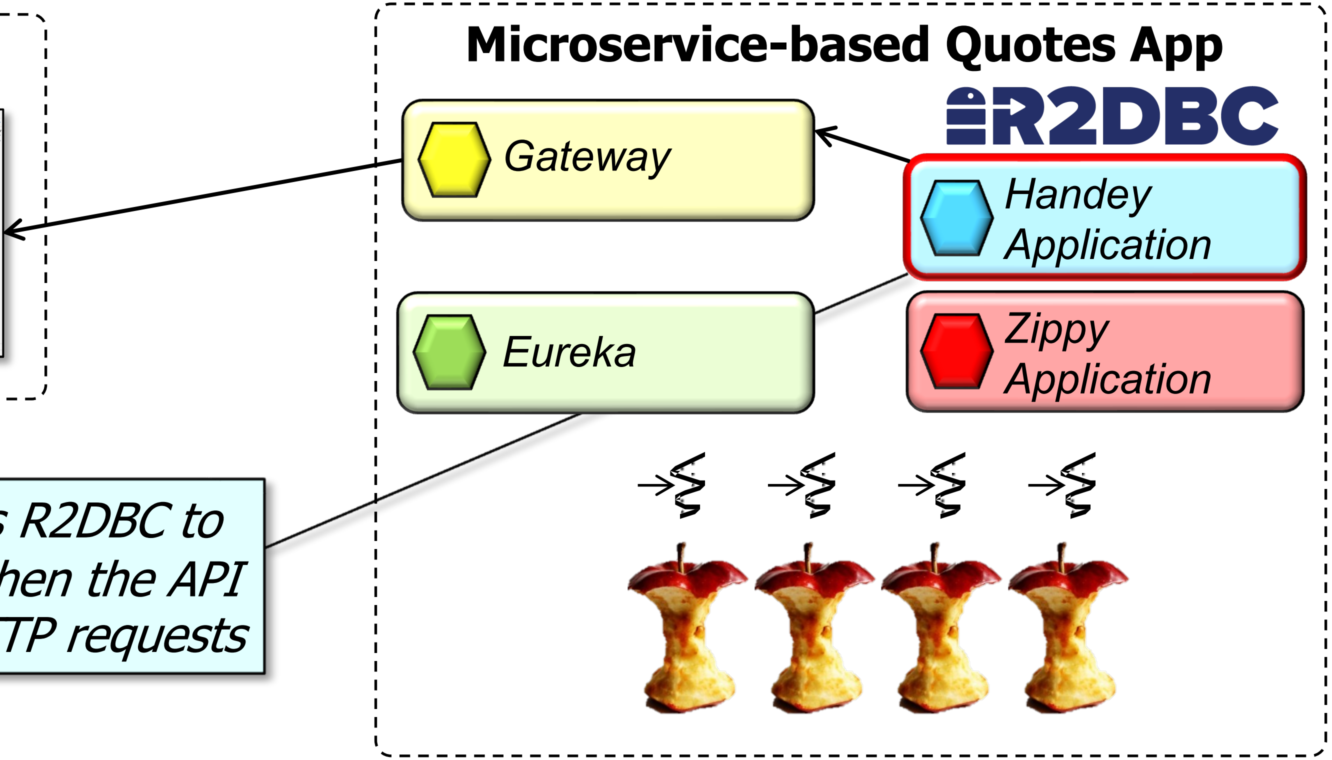
Overview of the Reactive QuoteServices App Case Study

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

QuoteDriver



```java
/**
 * 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}).
```



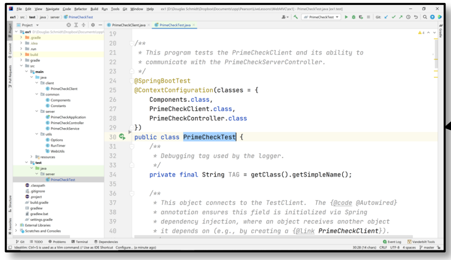
This microservice uses R2DBC to respond with quotes when the API gateway forwards it HTTP requests

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

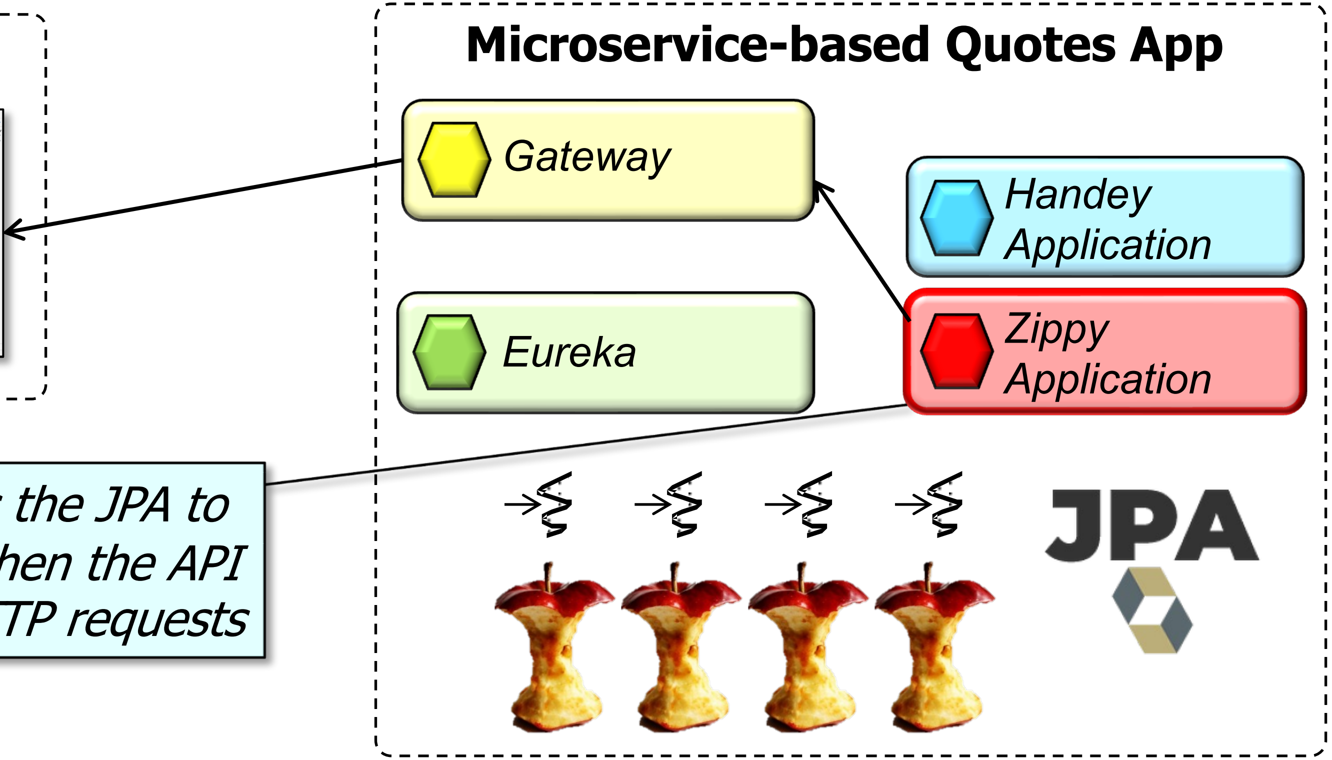
Overview of the Reactive QuoteServices App Case Study

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

QuoteDriver



```java
/**
 * 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}).
```



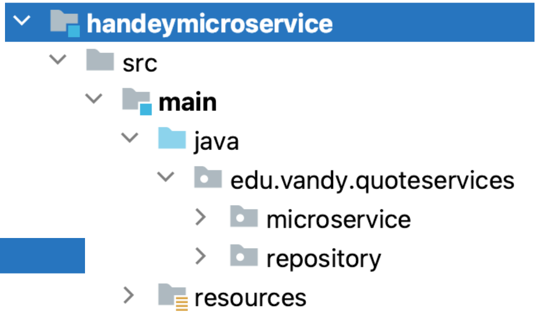
This microservice uses the JPA to respond with quotes when the API gateway forwards it HTTP requests

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

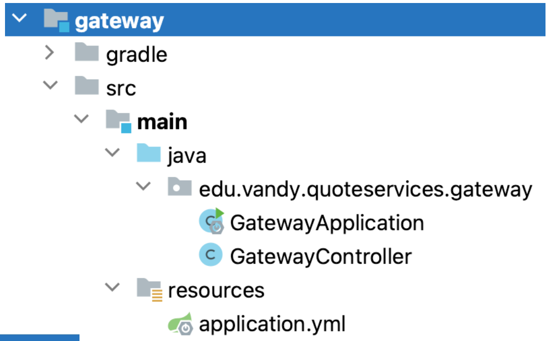
Structure of the Reactive Quote Services App Project

Structure of the Reactive QuoteServices App Project

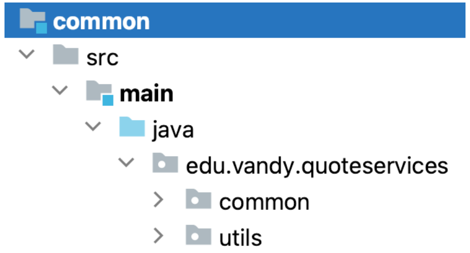
- The QuoteServices App project source code is organized into several modules & packages



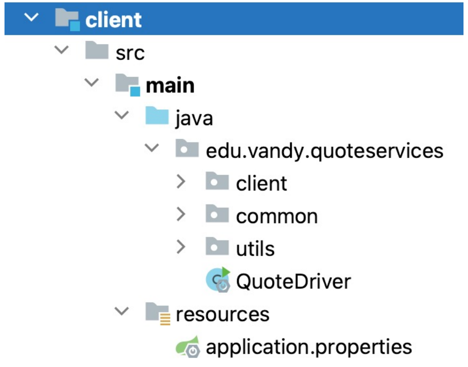
```
handeymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
          repository
      resources
```



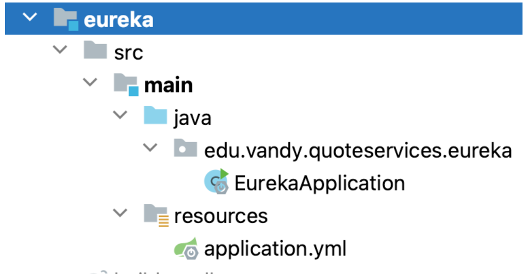
```
gateway
  gradle
  src
    main
      java
        edu.vandy.quoteservices.gateway
          GatewayApplication
          GatewayController
      resources
        application.yml
```



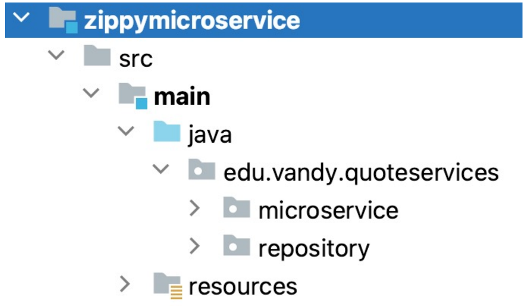
```
common
  src
    main
      java
        edu.vandy.quoteservices
          common
          utils
```



```
client
  src
    main
      java
        edu.vandy.quoteservices
          client
          common
          utils
          QuoteDriver
      resources
        application.properties
```



```
eureka
  src
    main
      java
        edu.vandy.quoteservices.eureka
          EurekaApplication
      resources
        application.yml
```

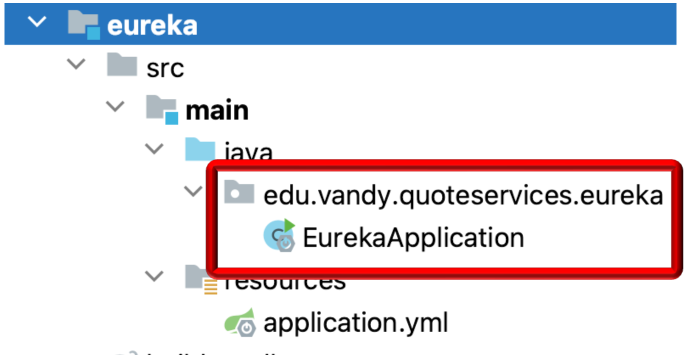


```
zippymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
          repository
      resources
```

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

Structure of the Reactive QuoteServices App Project

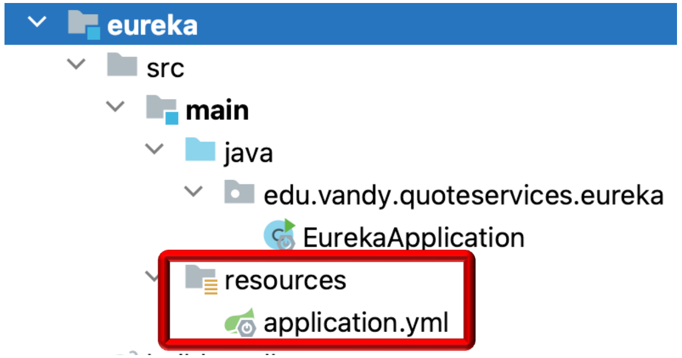
- The QuoteServices App project source code is organized into several modules & packages
  - eureka
    - eureka
      - Contains the "app" entry point for the Eureka discovery service



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

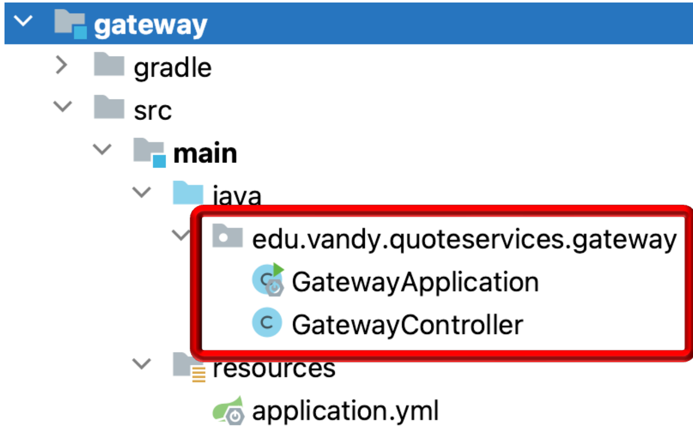
Structure of the Reactive QuoteServices App Project

- 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

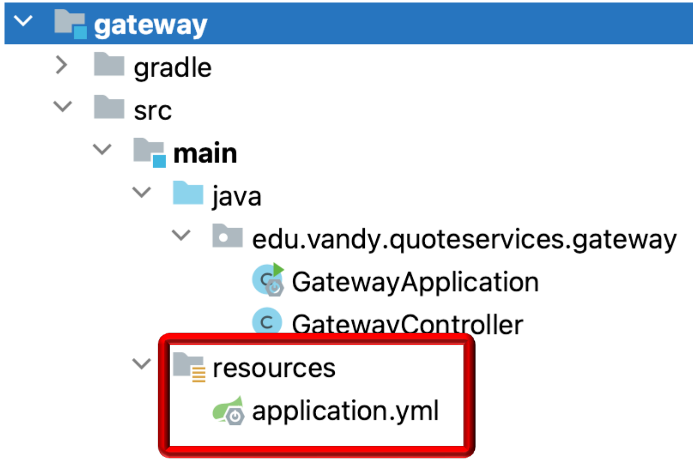
- The QuoteServices App project source code is organized into several modules & packages
  - gateway
    - gateway
      - Contains the "app" entry points & the controller
        - The gateway is largely programmed declaratively

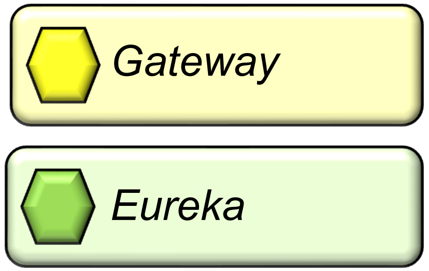


See github.com/douglascraigschmidt/LiveLessons/tree/master/WebFlux/ex3/gateway

Structure of the Reactive QuoteServices App Project

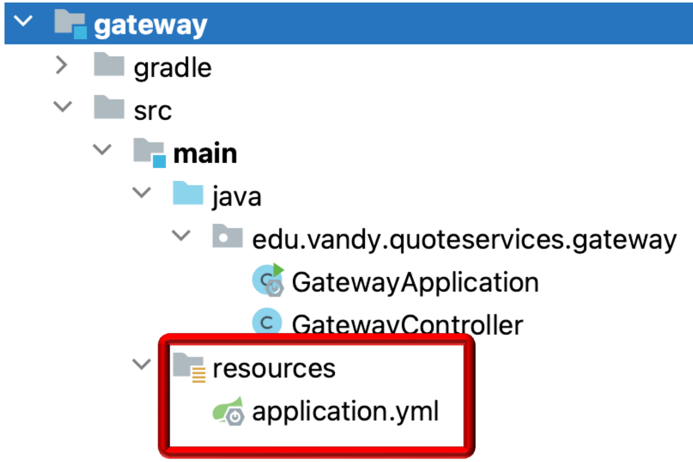
- The QuoteServices App project source code is organized into several modules & packages
  - gateway
    - gateway
    - resources
      - Configures the gateway to use the Eureka discovery service


```
gateway
  gradle
  src
    main
      java
        edu.vandy.quoteservices.gateway
          GatewayApplication
          GatewayController
      resources
        application.yml
```



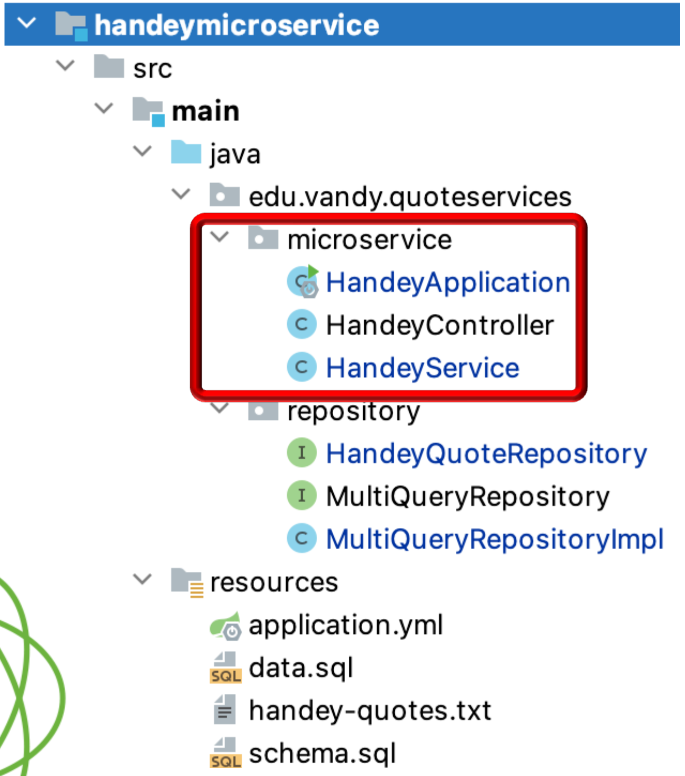
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - gateway
    - gateway
    - resources
      - Configures the gateway to use the Eureka discovery service
      - Specifies the port number exposed by the API gateway



Structure of the Reactive QuoteServices App Project

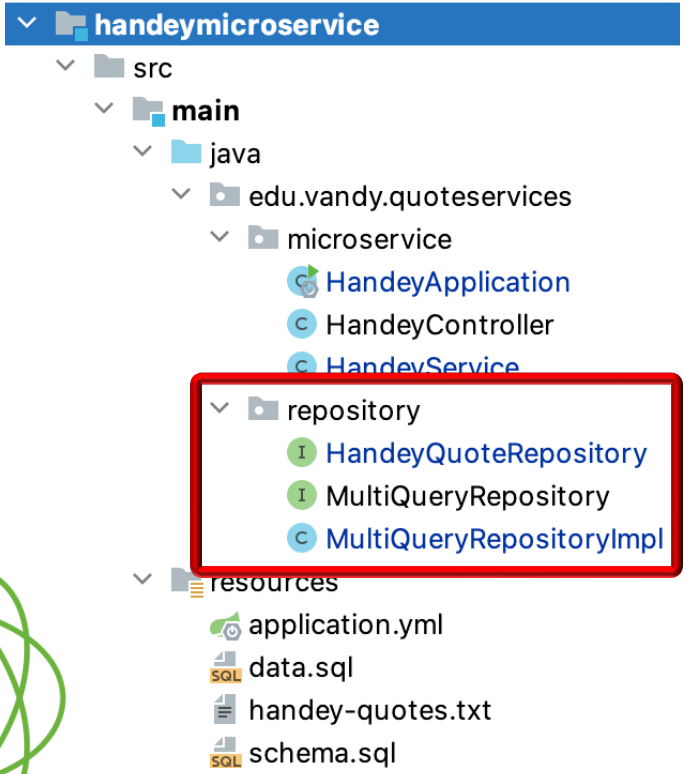
- The QuoteServices App project source code is organized into several modules & packages
  - handeymicroservice
    - microservice
      - Contains the "app" entry points & the controller for an R2DBC database
        - Returns reactive types



```
handeymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
            HandeyApplication
            HandeyController
            HandeyService
          repository
            HandeyQuoteRepository
            MultiQueryRepository
            MultiQueryRepositoryImpl
      resources
        application.yml
        data.sql
        handey-quotes.txt
        schema.sql
```

Structure of the Reactive QuoteServices App Project

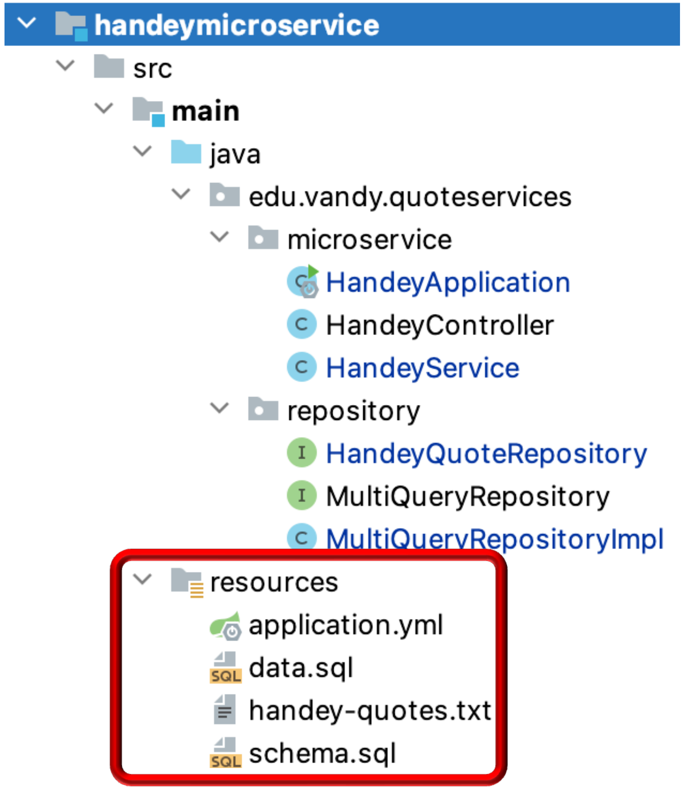
- The QuoteServices App project source code is organized into several modules & packages
  - handeymicroservice
    - microservice
    - repository
      - Implements the R2DBC database repository
        - Returns reactive types



```
handeymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
            HandeyApplication
            HandeyController
            HandeyService
          repository
            HandeyQuoteRepository
            MultiQueryRepository
            MultiQueryRepositoryImpl
      resources
        application.yml
        data.sql
        handey-quotes.txt
        schema.sql
```

Structure of the Reactive QuoteServices App Project

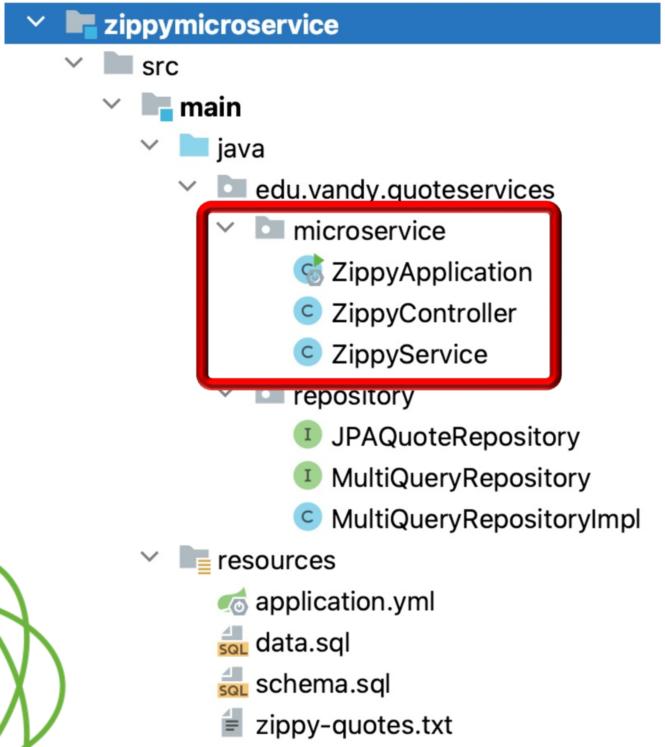
- The QuoteServices App project source code is organized into several modules & packages
  - handeymicroservice
    - microservice
    - repository
    - resources
      - Defines various application properties via YAML & SQL files
        - e.g., microservice name, Eureka client configuration, schema definitions & data for Handey quotes



```
handeymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
            HandeyApplication
            HandeyController
            HandeyService
          repository
            HandeyQuoteRepository
            MultiQueryRepository
            MultiQueryRepositoryImpl
      resources
        application.yml
        data.sql
        handey-quotes.txt
        schema.sql
```

Structure of the Reactive QuoteServices App Project

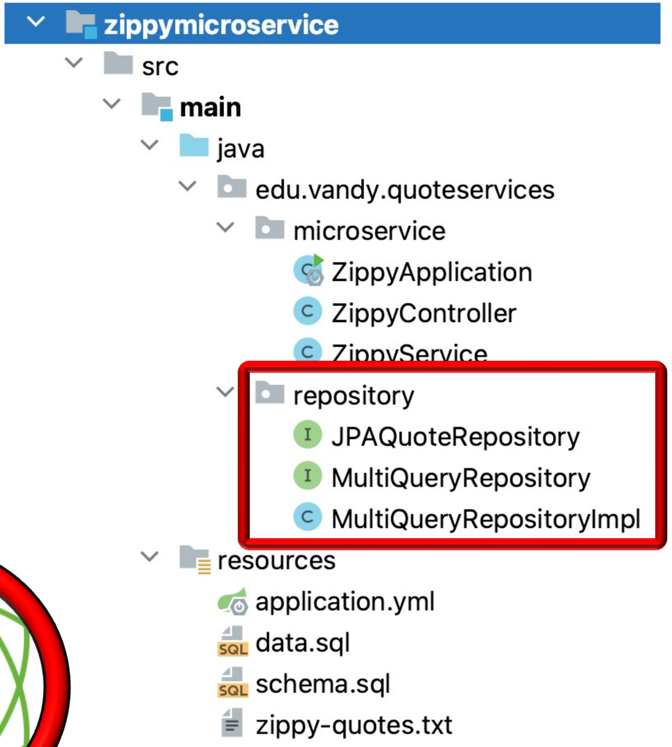
- The QuoteServices App project source code is organized into several modules & packages
  - zippymicroservice
    - microservice
      - Contains the "app" entry points & the controller for a JPA database
        - Returns reactive types, however



```
zippymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
            ZippyApplication
            ZippyController
            ZippyService
          repository
            JPAQuoteRepository
            MultiQueryRepository
            MultiQueryRepositoryImpl
      resources
        application.yml
        data.sql
        schema.sql
        zippy-quotes.txt
```

Structure of the Reactive QuoteServices App Project

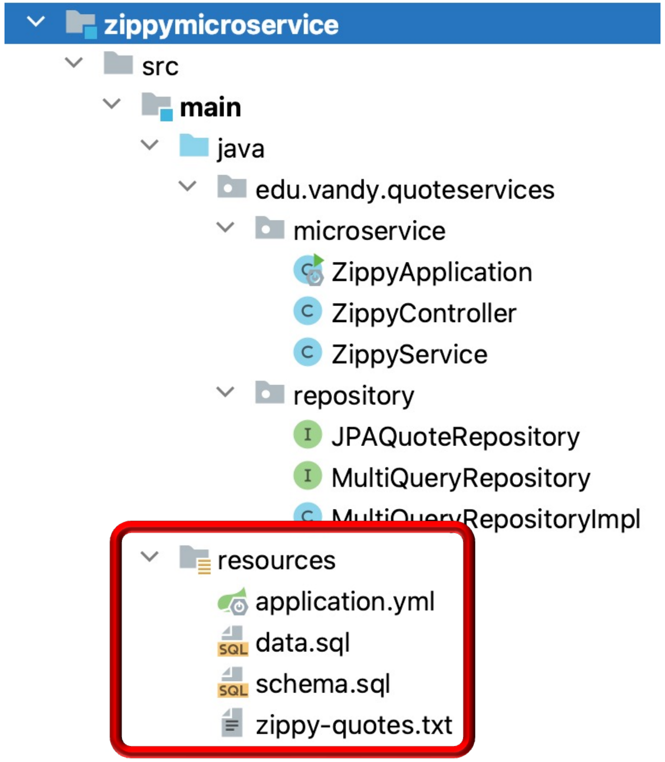
- The QuoteServices App project source code is organized into several modules & packages
  - zippymicroservice
    - microservice
    - repository
      - Implements the JPA database repository
        - Does not return reactive types



```
zippymicroservice
  src
    main
      java
        edu.vandy.quoteservices
          microservice
            ZippyApplication
            ZippyController
            ZippyService
          repository
            JPAQuoteRepository
            MultiQueryRepository
            MultiQueryRepositoryImpl
      resources
        application.yml
        data.sql
        schema.sql
        zippy-quotes.txt
```

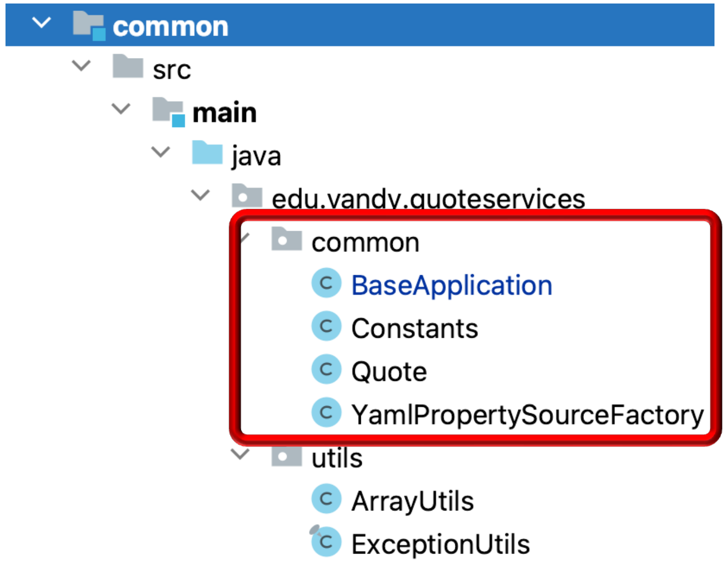
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - zippymicroservice
    - microservice
    - repository
    - resources
      - Defines various application properties via YAML & SQL files
        - e.g., microservice name, Eureka client configuration, schema definitions & data for Zippy quotes


zippymicroservice
src
main
java
edu.vandy.quoteservices
microservice
ZippyApplication
ZippyController
ZippyService
repository
JPAQuoteRepository
MultiQueryRepository
MultiQueryRepositoryImpl
resources
application.yml
data.sql
schema.sql
zippy-quotes.txt

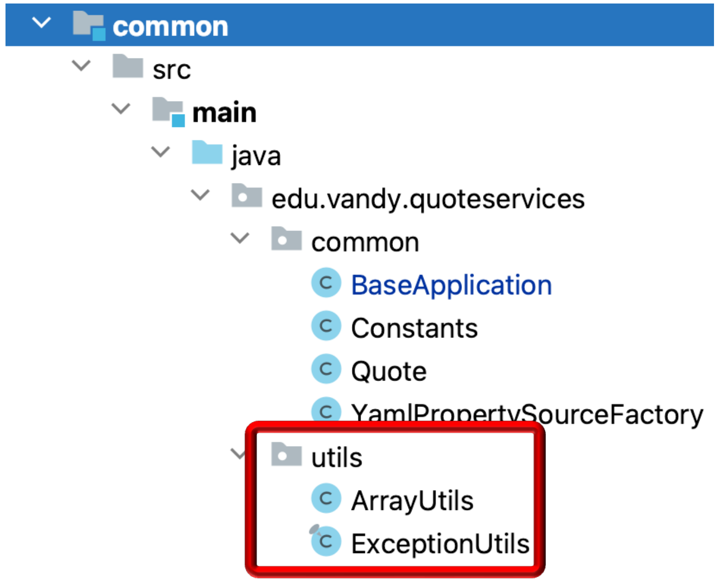
Structure of the Reactive QuoteServices App Project

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



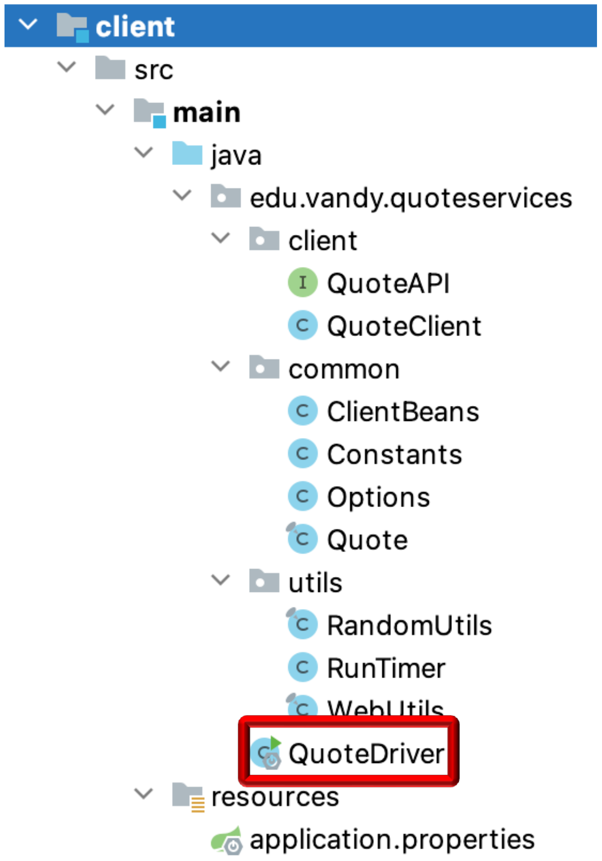
Structure of the Reactive QuoteServices App Project

- The QuoteServices App project source code is organized into several modules & packages
  - common
    - common
    - utils
      - Helper classes that are reused by other projects



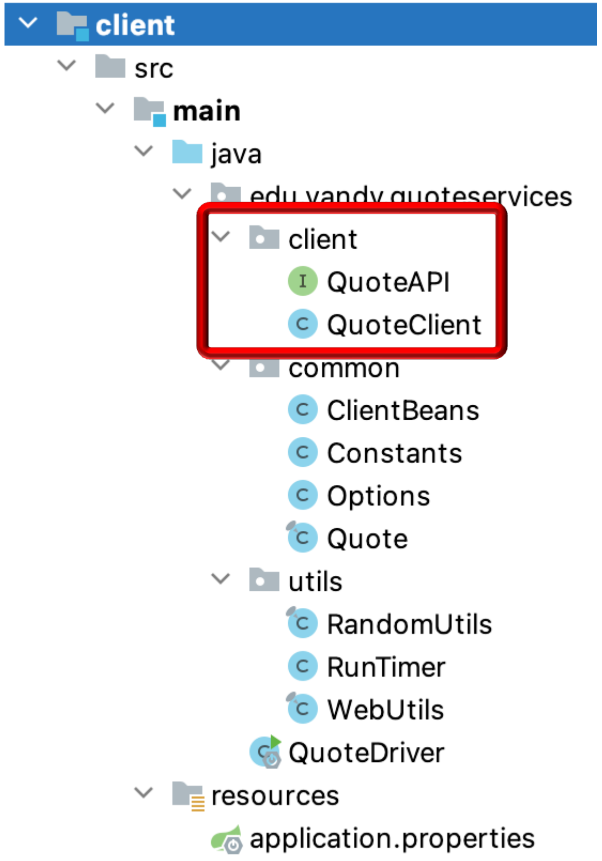
Structure of the Reactive QuoteServices App Project

- 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 API gateway running on the server & displays results



Structure of the Reactive QuoteServices App Project

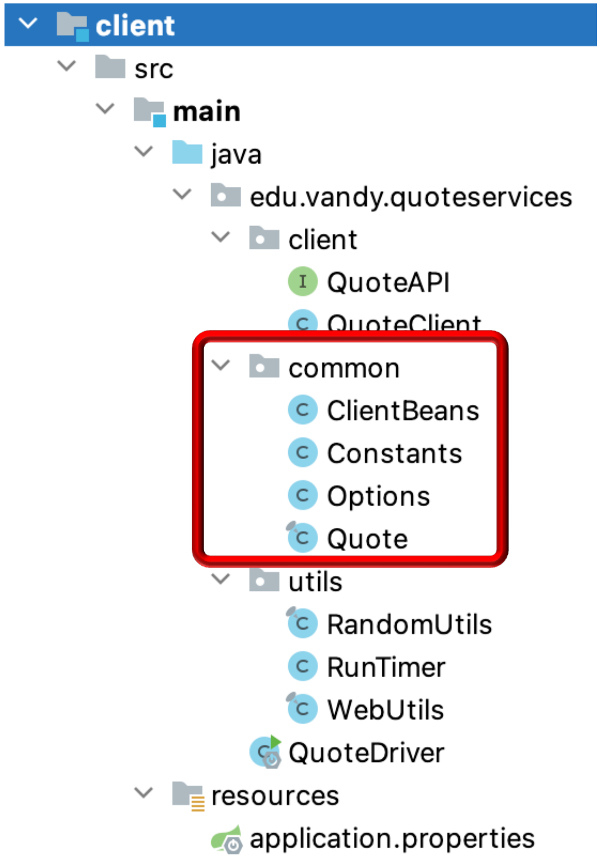
- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
    - client
      - Sends HTTP GET/POST requests to the microservices using reactive types
        - Implemented using Spring 6+ HTTP interface clients



```
client
  src
    main
      java
        edu.vandy.quoteservices
          client
            QuoteAPI
            QuoteClient
          common
            ClientBeans
            Constants
            Options
            Quote
          utils
            RandomUtils
            RunTimer
            WebUtils
          QuoteDriver
      resources
        application.properties
```

Structure of the Reactive QuoteServices App Project

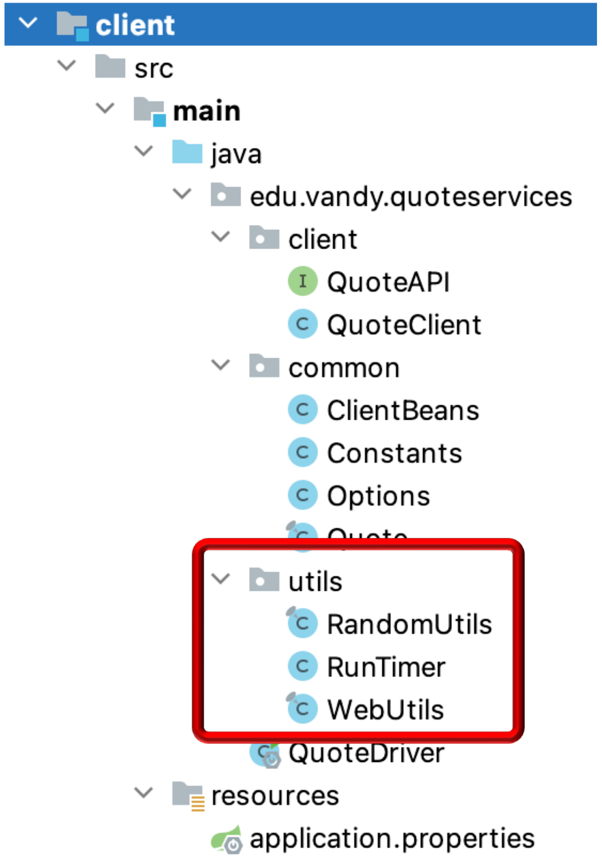
- The QuoteServices App project source code is organized into several modules & packages
  - client
    - QuoteDriver
    - client
    - common
      - Helper classes that are specific to this client driver



```
client
  src
    main
      java
        edu.vandy.quoteservices
          client
            QuoteAPI
            QuoteClient
          common
            ClientBeans
            Constants
            Options
            Quote
          utils
            RandomUtils
            RunTimer
            WebUtils
          QuoteDriver
      resources
        application.properties
```

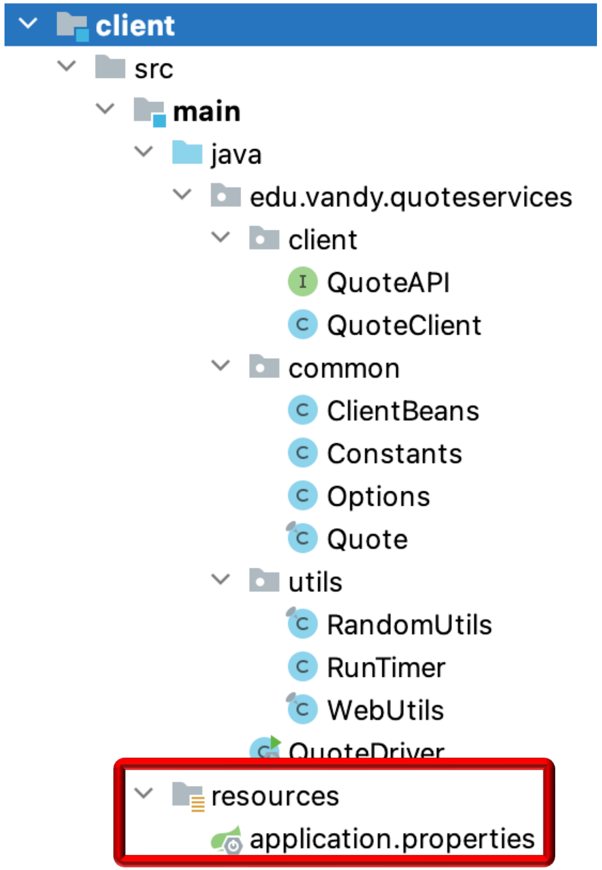
Structure of the Reactive QuoteServices App Project

- 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


```
client
  src
    main
      java
        edu.vandy.quoteservices
          client
            QuoteAPI
            QuoteClient
          common
            ClientBeans
            Constants
            Options
            Quote
          utils
            RandomUtils
            RunTimer
            WebUtils
          QuoteDriver
      resources
        application.properties
```

End of the Reactive QuoteServices App Case Study: Overview