

The QuoteServices App Case Study: Implementing the Gateway Microservice

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

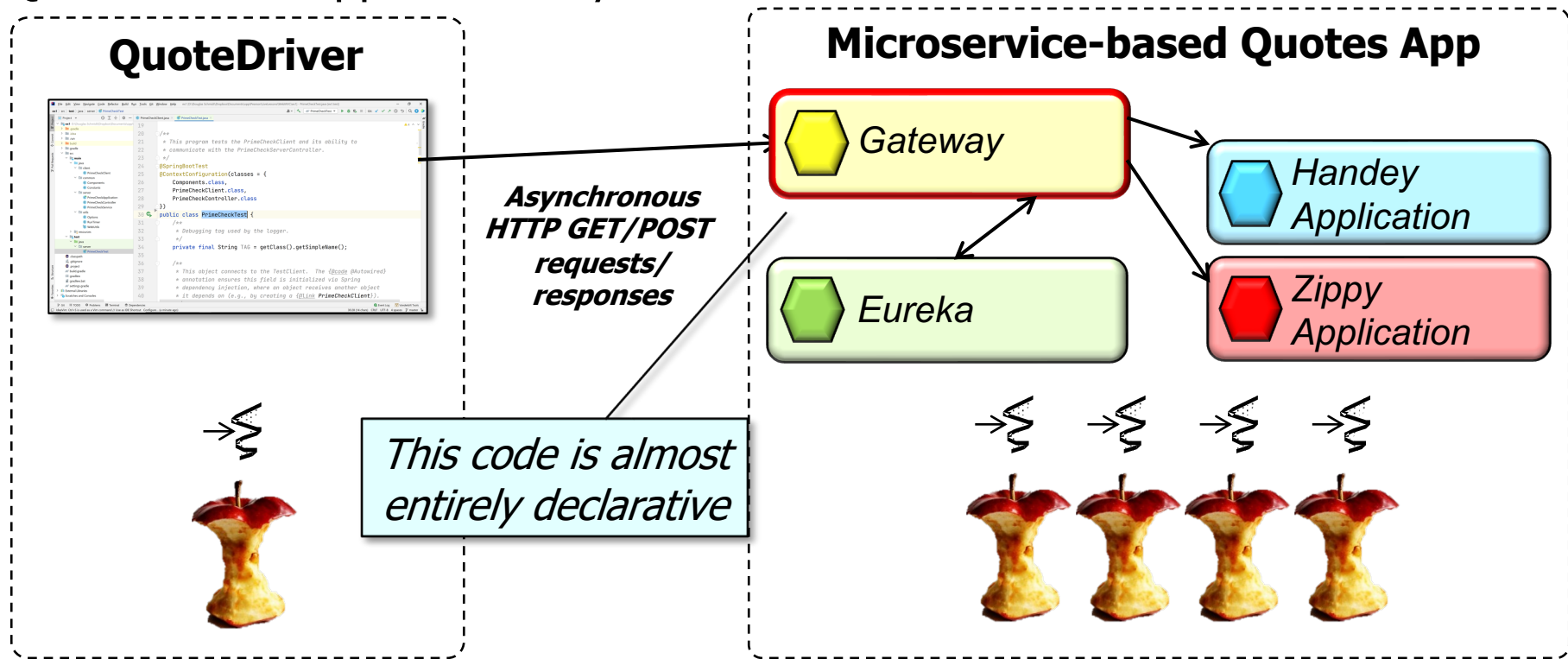
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

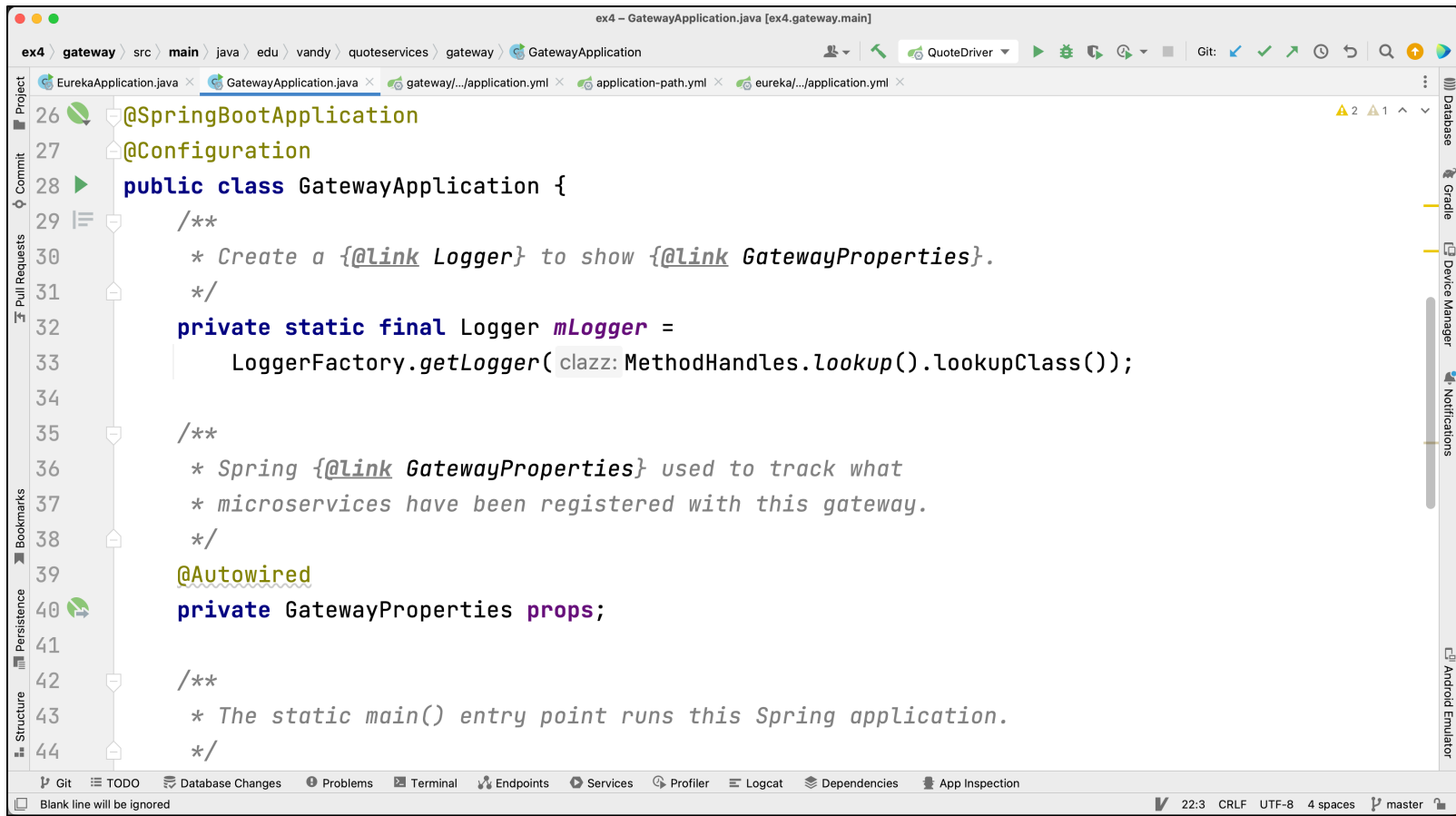
- Understand the implementation of the Gateway microservice in the reactive QuoteServices app case study



The implementation of this Gateway is quite minimal due to the use of Eureka!

Implementing the Gateway Microservice

Implementing the Gateway Microservice



The screenshot shows an IDE window titled "ex4 - GatewayApplication.java [ex4.gateway.main]". The code is for a Spring Boot application. The class is annotated with `@SpringBootApplication` and `@Configuration`. It contains a `public class GatewayApplication` with a `private static final Logger mLogger` initialized using `LoggerFactory.getLogger()`. The class is annotated with `@Autowired` and has a `private GatewayProperties props` field. The `main()` method is the entry point for the application.

```
26 @SpringBootApplication
27 @Configuration
28 public class GatewayApplication {
29     /**
30      * Create a {@link Logger} to show {@link GatewayProperties}.
31      */
32     private static final Logger mLogger =
33         LoggerFactory.getLogger( clazz: MethodHandles.lookup().lookupClass());
34
35     /**
36      * Spring {@link GatewayProperties} used to track what
37      * microservices have been registered with this gateway.
38      */
39     @Autowired
40     private GatewayProperties props;
41
42     /**
43      * The static main() entry point runs this Spring application.
44      */
45 }
```

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

End of the QuoteServices App Case Study: Implementing the Gateway MicroService