The QuoteServices App Case Study: Structure & Functionality of Client Classes

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

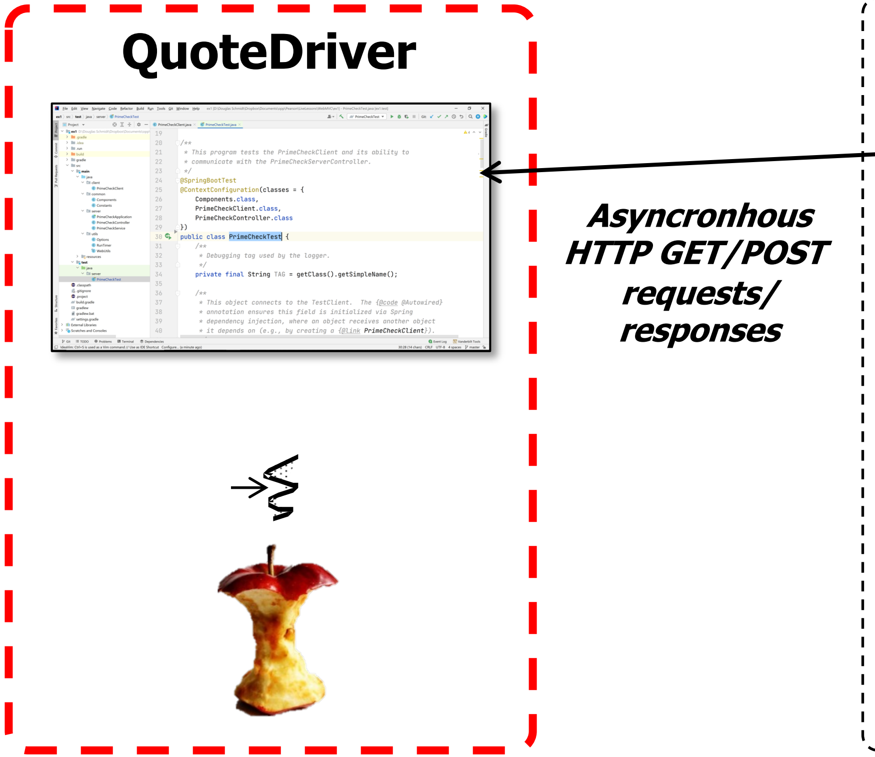
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

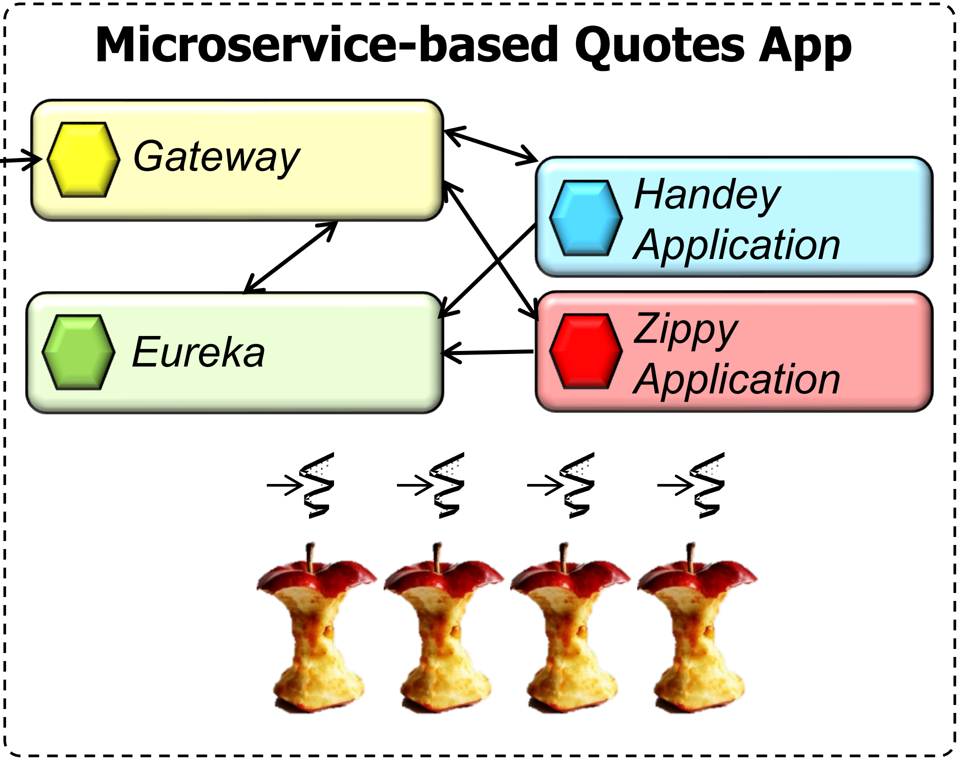
Learning Objectives in this Part of the Lesson

- Understand the client structure & functionality using Spring 6+ HTTP interface clients to send /receive HTTP GET/POST requests/responses to/from the microservices



```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

The Structure & Functionality of QuoteClient Class

The Structure & Functionality of QuoteClient Class

- The QuoteClient class performs asynchronous remote method invocations on the API Gateway microservice to obtain Handey & Zippy quotes

```
@Component
public class QuoteClient {
  @Autowired QuoteAPI mQuoteAPI;
  ...
  public Flux<Quote> postQuotes
    (String routename, List<Integer> quoteIds) {
    return mQuoteAPI.postQuotes(routename, quoteIds);
  }

  public Flux<Quote> searchQuotes(String routename,
                                  List<String> queries) {
    return mQuoteAPI.search(routename, queries);
  } ...
```

See WebFlux/ex3/client/src/main/java/edu/vandy/quoteservices/client/QuoteClient.java

The Structure & Functionality of QuoteClient Class

- The QuoteClient class performs asynchronous remote method invocations on the API Gateway microservice to obtain Handey & Zippy quotes

```
@Component
public class QuoteClient {
  @Autowired QuoteAPI mQuoteAPI;
  ...
  public Flux<Quote> postQuotes
    (String routename, List<Integer> quoteIds) {
    return mQuoteAPI.postQuotes(routename, quoteIds);
  }

  public Flux<Quote> searchQuotes(String routename,
                                  List<String> queries) {
    return mQuoteAPI.search(routename, queries);
  } ...
```

Enables auto-detection & wiring of dependent implementation classes via classpath scanning

See www.baeldung.com/spring-component-repository-service

The Structure & Functionality of QuoteClient Class

- The QuoteClient class performs asynchronous remote method invocations on the API Gateway microservice to obtain Handey & Zippy quotes

```
@Component
public class QuoteClient {
  @Autowired QuoteAPI mQuoteAPI;
  ...
  public Flux<Quote> postQuotes
    (String routename, List<Integer> quoteIds) {
    return mQuoteAPI.postQuotes(routename, quoteIds);
  }

  public Flux<Quote> searchQuotes(String routename,
                                  List<String> queries) {
    return mQuoteAPI.search(routename, queries);
  } ...
```

Spring's dependency injection framework auto-wires this field

See www.baeldung.com/spring-autowire

The Structure & Functionality of QuoteClient Class

- The QuoteClient class performs asynchronous remote method invocations on the API Gateway microservice to obtain Handey & Zippy quotes

```
@Component
public class QuoteClient {
  @Autowired QuoteAPI mQuoteAPI;
  ...
  public Flux<Quote> postQuotes
    (String routename, List<Integer> quoteIds) {
    return mQuoteAPI.postQuotes(routename, quoteIds);
  }

  public Flux<Quote> searchQuotes(String routename,
                                  List<String> queries) {
    return mQuoteAPI.search(routename, queries);
  } ...
```

Get a Flux of Quotes based on their quote Ids from the given microservice

The Structure & Functionality of QuoteClient Class

- The QuoteClient class performs asynchronous remote method invocations on the API Gateway microservice to obtain Handey & Zippy quotes

```
@Component
public class QuoteClient {
  @Autowired QuoteAPI mQuoteAPI;
  ...
  public Flux<Quote> postQuotes
    (String routename, List<Integer> quoteIds) {
    return mQuoteAPI.postQuotes(routename, quoteIds);
  }

  public Flux<Quote> searchQuotes(String routename,
                                  List<String> queries) {
    return mQuoteAPI.search(routename, queries);
  } ...
```

'routename' enables dynamic path updating via HTTP interface

The Structure & Functionality of the QuoteAPI Interface

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST_QUOTES)
  Flux<Quote> postQuotes(@PathVariable String routename,
                         @RequestBody List<Integer> quoteIds);

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                     @RequestBody List<String> queries);
  ...
```

See WebFlux/ex3/client/src/main/java/edu/vandy/quoteservices/client/QuoteAPI.java

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST
  Flux<Quote> postQuotes(@PathVariable
                              @RequestBody

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                          @RequestBody List<String> queries);
  ...
```

These annotations mark a proxy method as accessing an HTTP GET/POST endpoint

See http-declarative-http-client-httpexchange/#3-creating-an-http-service-interface

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST_QUOTES)
  Flux<Quote> postQuotes(@PathVariable String routename,
                         @RequestBody List<Integer> quoteIds);

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                     @RequestBody List<String> queries);
  ...
```

These proxy methods mimic the signature of controller methods

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST_QUOTES)
  Flux<Quote> postQuotes(@PathVariable String routename,
                         @RequestBody List<Integer> quoteIds);

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                     @RequestBody List<String> queries);
  ...
```

These methods return the Project Reactor Flux type!

In contrast, Retrofit does not support the Project Reactor Flux type

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST_QUOTES)
  Flux<Quote> postQuotes(@PathVariable String routename,
                         @RequestBody List<Integer> quoteIds);

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                     @RequestBody List<String> queries);

  ...
```

Even the annotations are the same as the Spring controller annotations!

The Structure & Functionality of the *QuoteAPI Classes

- The *QuoteAPI classes uses the Spring 6+ HTTP interface to abstract details of making async remote method invocations using HTTP

```
public interface QuoteAPI {
  @GetExchange(ROUTENAME + "/" + GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(@PathVariable String routename);

  @PostExchange(ROUTENAME + "/" + POST_QUOTES)
  Flux<Quote> postQuotes(@PathVariable String routename,
                         @RequestBody List<Integer> quoteIds);

  @PostExchange(ROUTENAME + "/" + POST_SEARCHES)
  Flux<Quote> search(@PathVariable String routename,
                     @RequestBody List<String> queries);

  ...
```

This annotation supports dynamic path routing!

The Structure & Functionality of the ClientBeans Class

The Structure & Functionality of the ClientBeans Class

- The ClientBeans class contains factory method beans that create the Zippy QuoteAPI & HandeyQuoteAPI proxies

```
@Component
public class ClientBeans {
  @Bean
  public QuoteAPI getQuoteAPI() {
    var webClient = WebClient.builder()
        .baseUrl(GATEWAY_BASE_URL).build();

    return HttpServiceProxyFactory
        .builder(WebClientAdapter
                 .forClient(webClient))
        .build()
        .createClient(QuoteAPI.class);
  ...
```

This @Bean annotation can be injected into classes using Spring's @Autowired annotation

See WebFlux/ex3/client/src/main/java/edu/vandy/quoteservices/common/ClientBeans.java

The Structure & Functionality of the QuoteProxy Class

- The ClientBeans class contains factory method beans that create the Zippy QuoteAPI & HandeyQuoteAPI proxies

```
@Component
public class ClientBeans {
  @Bean
  public QuoteAPI getQuoteAPI() {
    var webClient = WebClient.builder()
        .baseUrl(GATEWAY_BASE_URL).build();

    return HttpServiceProxyFactory
        .builder(WebClientAdapter
                 .forClient(webClient))
        .build()
        .createClient(QuoteAPI.class);
  ...
```

Create a QuoteAPI HTTP client that's used to make HTTP requests to the GatewayApplication microservice

The Structure & Functionality of the QuoteProxy Class

- The ClientBeans class contains factory method beans that create the Zippy QuoteAPI & HandeyQuoteAPI proxies

```
@Component
public class ClientBeans {
  @Bean
  public QuoteAPI getQuoteAPI() {
    var webClient = WebClient.builder()
        .baseUrl(GATEWAY_BASE_URL).build();

    return HttpServiceProxyFactory
        .builder(WebClientAdapter
                 .forClient(webClient))
        .build()
        .createClient(QuoteAPI.class);
  ...
```

Create a new WebClient that connects to the Gateway

The Structure & Functionality of the QuoteProxy Class

- The ClientBeans class contains factory method beans that create the Zippy QuoteAPI & HandeyQuoteAPI proxies

```
@Component
public class ClientBeans {
  @Bean
  public QuoteAPI getQuoteAPI() {
    var webClient = WebClient.builder()
        .baseUrl(GATEWAY_BASE_URL).build();

    return HttpServiceProxyFactory
        .builder(WebClientAdapter
                 .forClient(webClient))
        .build()
        .createClient(QuoteAPI.class);
  ...
```

Return a new instance of QuoteAPI that makes HTTP requests via WebClient

End of the QuoteServices App Case Study: Structure & Functionality of Client Classes