

Overview of the RSocket Quotes Backpressure App

Douglas C. Schmidt

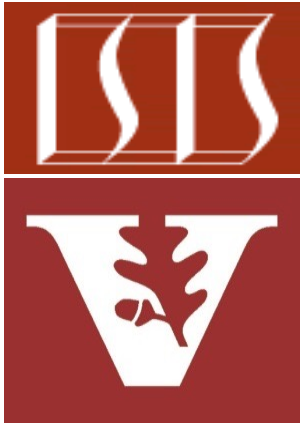
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

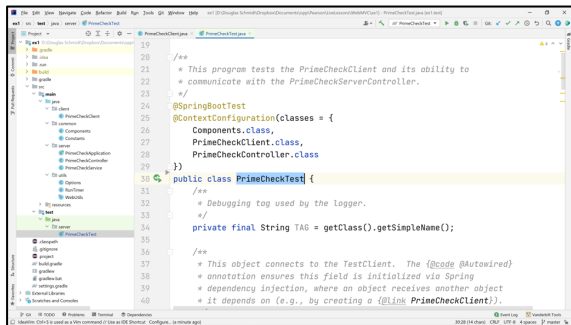
**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Lesson

- Understand how Spring RSocket can exchange binary messages to get Zippy & Handey quotes asynchronously using Project Reactor backpressure

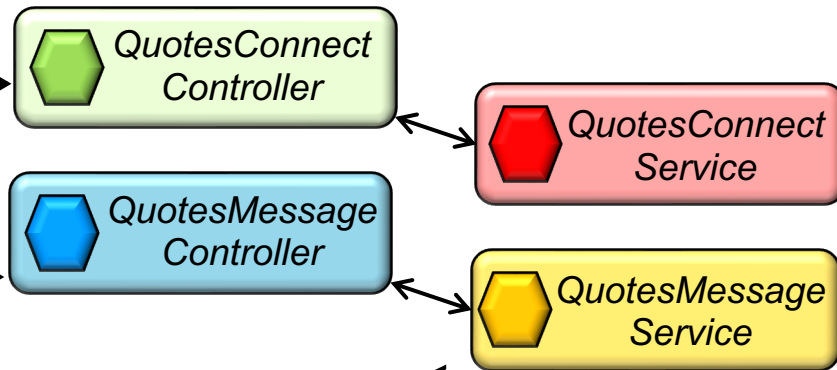
QuotesTest



```
19 //**
20 //**
21 //** This program tests the PrimeCheckClient and its ability to
22 //** communicate with the PrimeCheckServerController.
23 //**
24 @SpringBootTest
25 @ContextConfiguration(classes = {
26     PrimeCheckClient.class,
27     PrimeCheckController.class
28 })
29 //**
30 public class PrimeCheckTest {
31     //**
32     //** Debugging tag used by the logger.
33     //**
34     private final String TAG = getClass().getSimpleName();
35
36     //**
37     //** This object connects to the TestClient. The @Code @Autowired
38     //** annotation ensures this field is initialized via Spring
39     //** dependency injection, where an object receives another object
40     //** it depends on (e.g., by creating a @Link PrimeCheckClient);
41 }
```



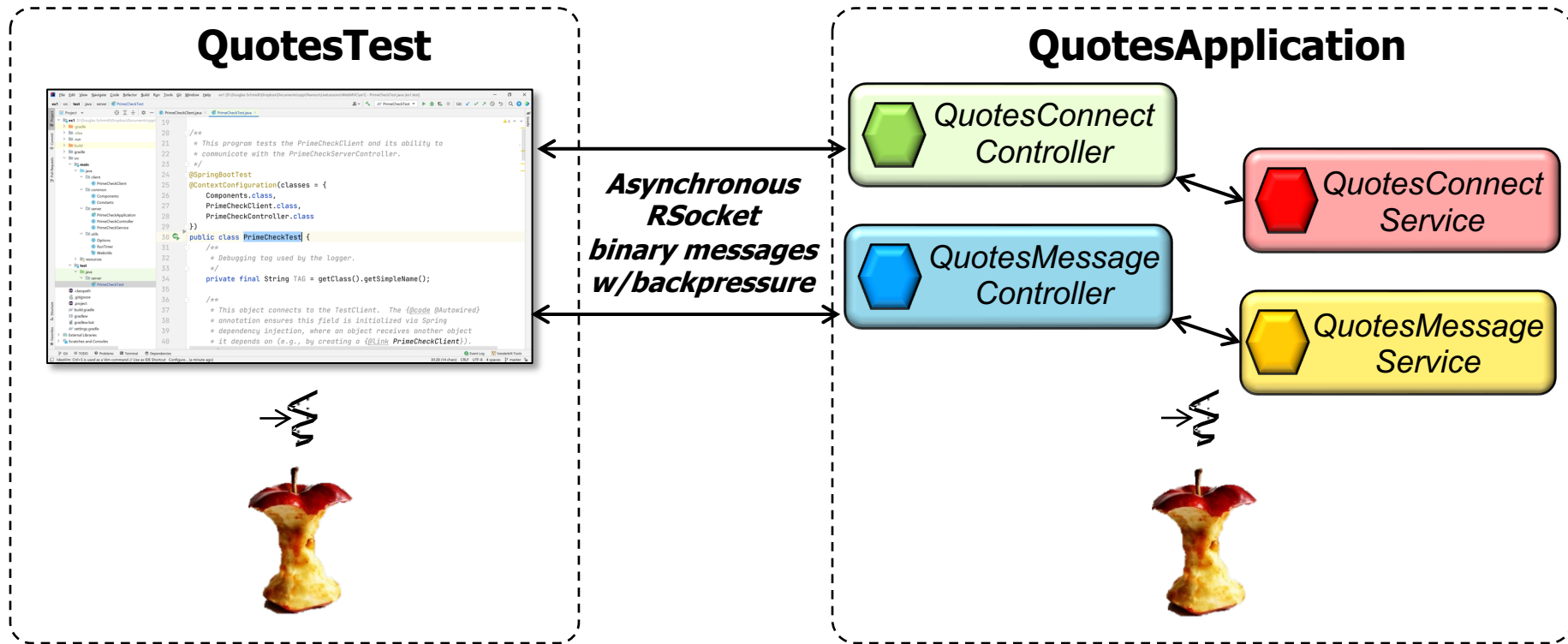
QuotesApplication



*Asynchronous
RSocket
binary messages
w/backpressure*

Learning Objectives in this Lesson

- Understand how Spring RSocket can exchange binary messages to get Zippy & Handey quotes asynchronously using Project Reactor backpressure

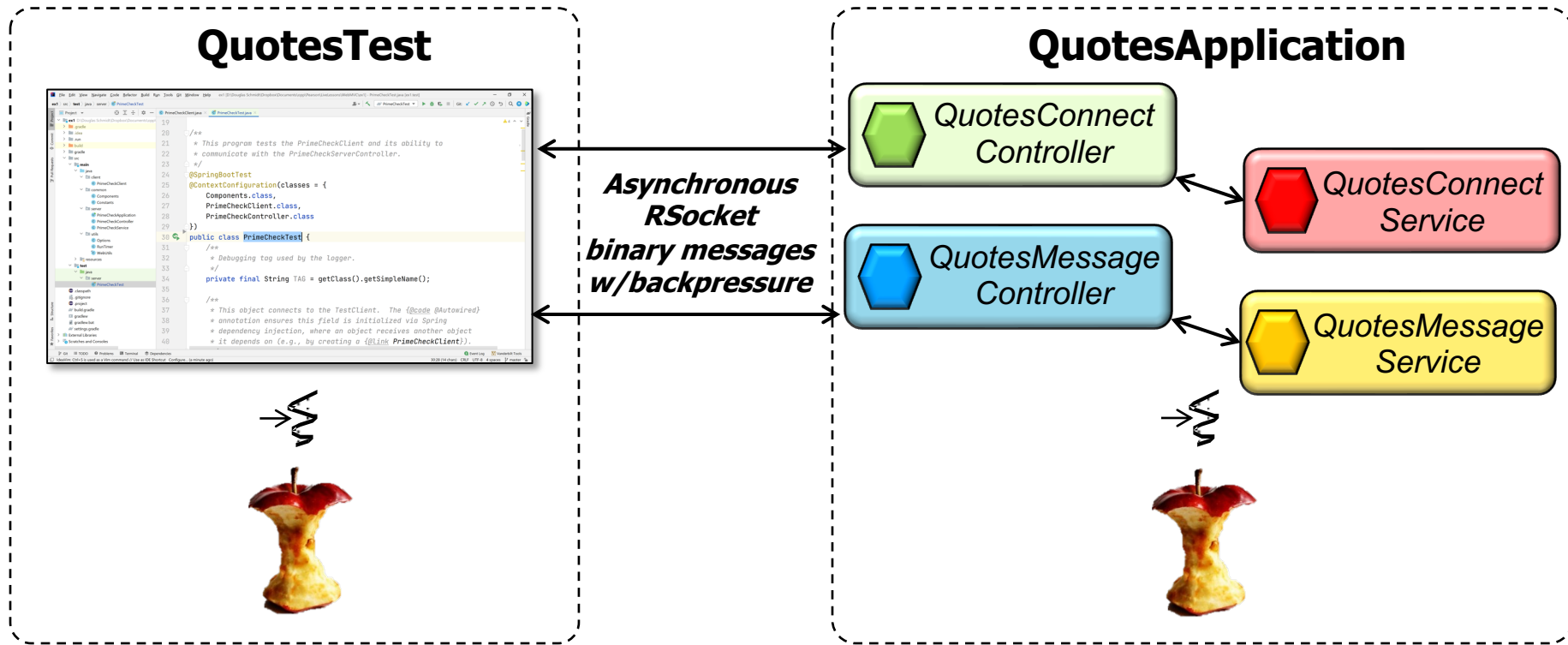


This example also shows how to use RSocket security & authentication features

Overview of the RSocket Quotes Backpressure App

Overview of the RSocket Quotes Backpressure App

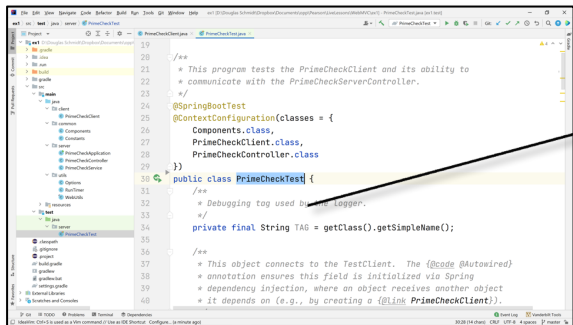
- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice using backpressure



Overview of the RSocket Quotes Backpressure App

- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice

QuotesTest



```
19 /**  
20  * This program tests the PrimeCheckClient and its ability to  
21  * communicate with the PrimeCheckServerController.  
22  */  
23 //}  
24 @SpringBootTest  
25 @ContextConfiguration(classes = {  
26     Components.class,  
27     PrimeCheckClient.class,  
28     PrimeCheckController.class  
29 })  
30 public class PrimeCheckTest {  
31     /**  
32     * Debugging tag used by log4j.  
33     */  
34     private final String TAG = getClass().getSimpleName();  
35  
36     /**  
37     * This object connects to the TestClient. The @Code @Autowired  
38     * annotation ensures this field is initialized via Spring  
39     * dependency injection, where an object receives another object  
40     * it depends on (e.g., by creating a @Link PrimeCheckClient).  
41     */  
42     @Autowired  
43     PrimeCheckClient client;  
44 }  
45
```

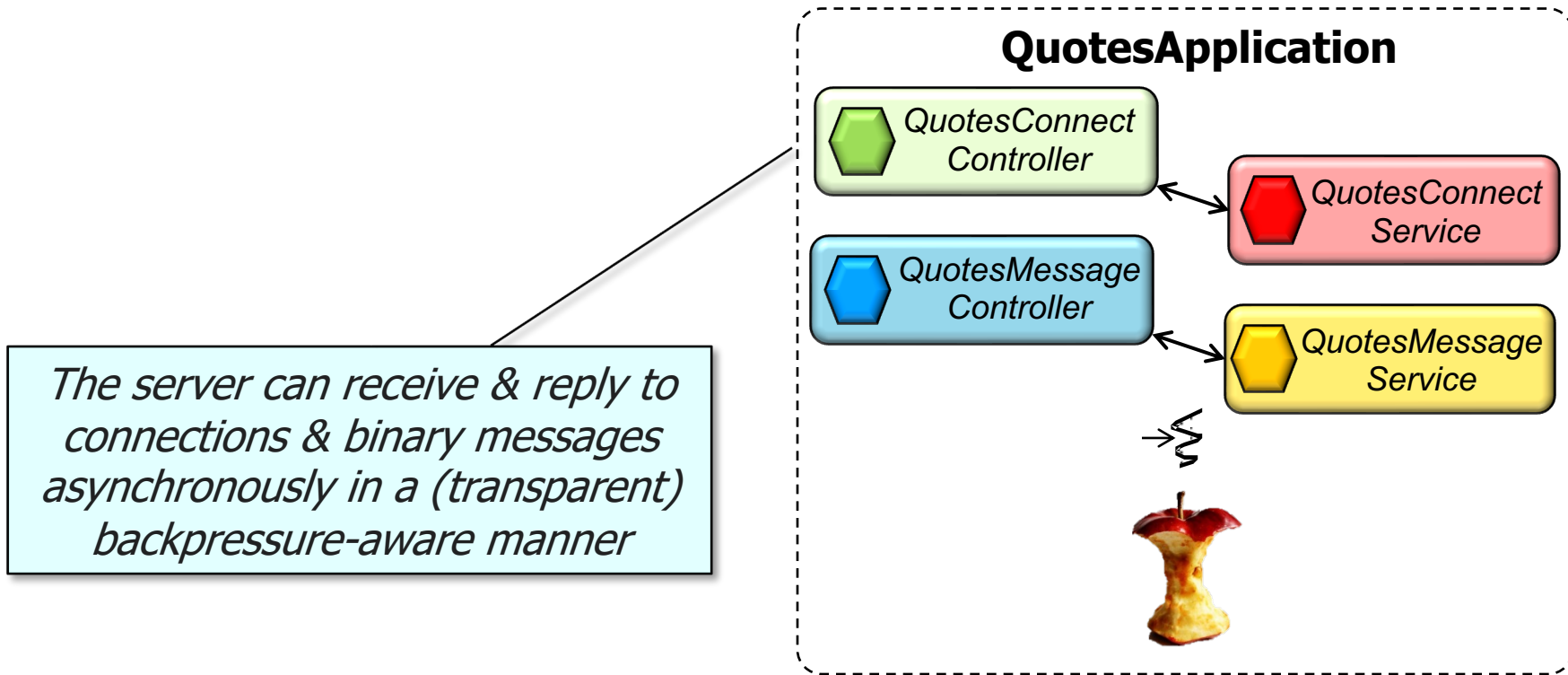
The client can asynchronously connect to the server, subscribe to either Zippy or Handey quotes, & receive these quotes via the Spring RSocket APIs using backpressure



The client also authenticates itself to the server

Overview of the RSocket Quotes Backpressure App

- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice

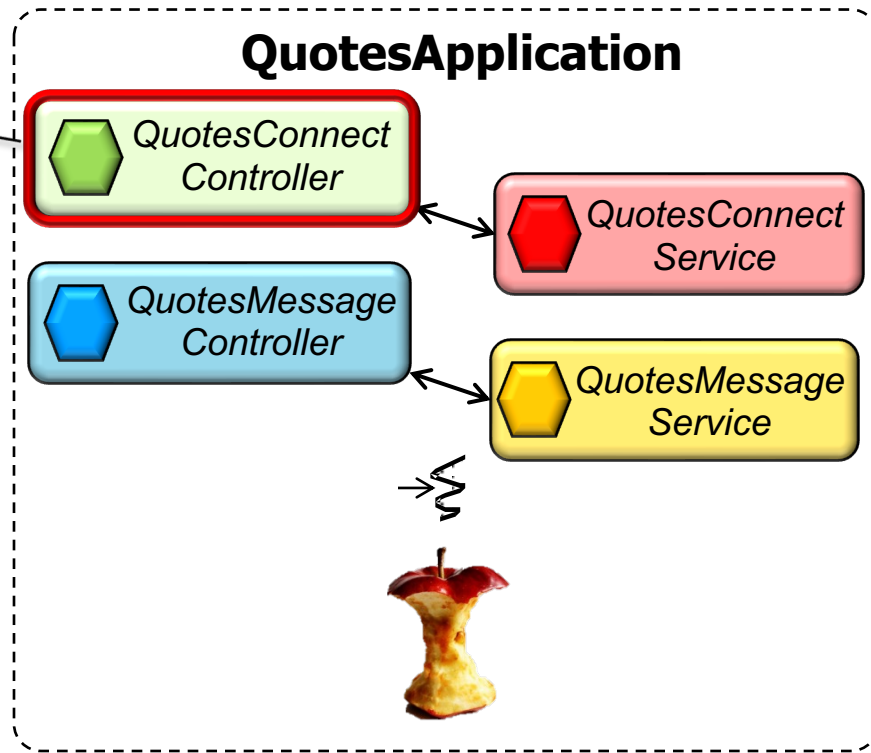


The server also requires the client to authentic itself before finalizing the connection

Overview of the RSocket Quotes Backpressure App

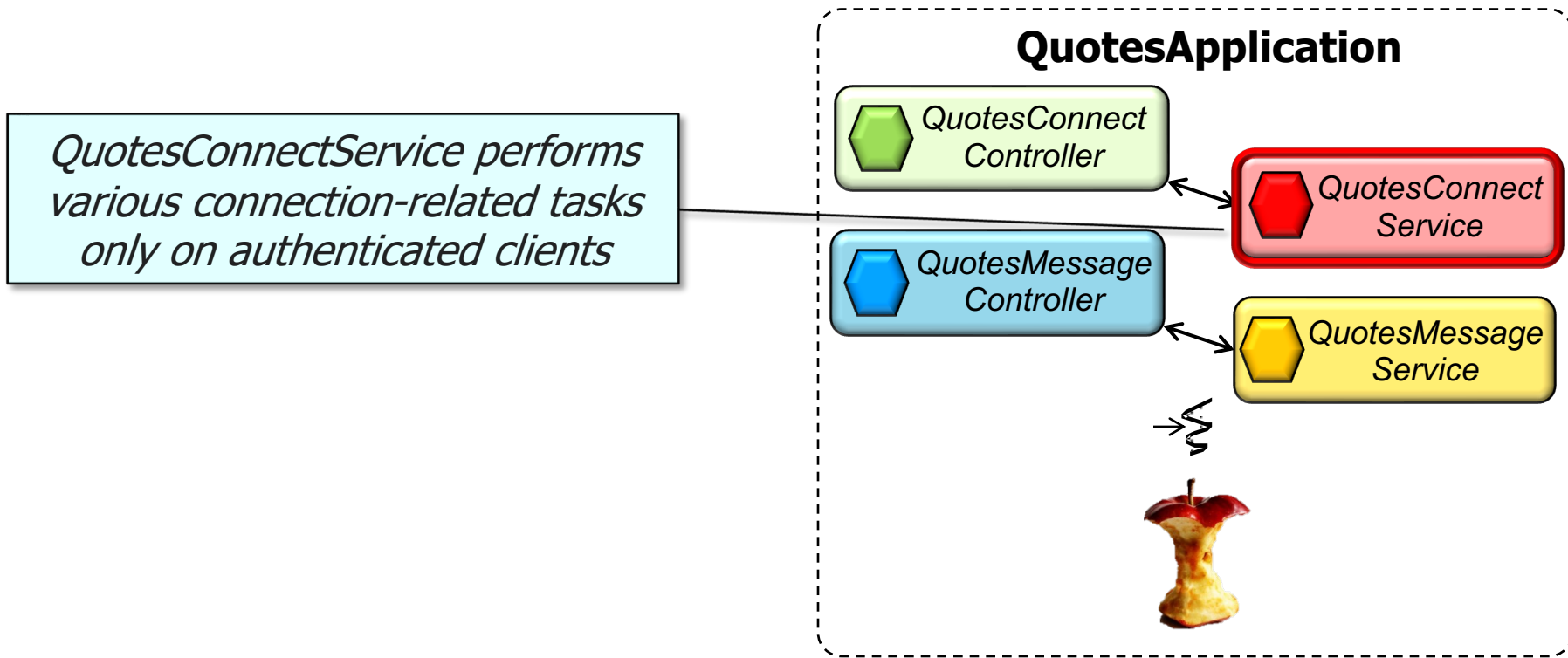
- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice

QuotesConnectController receives client connection requests & forwards them to the QuotesConnectService



Overview of the RSocket Quotes Backpressure App

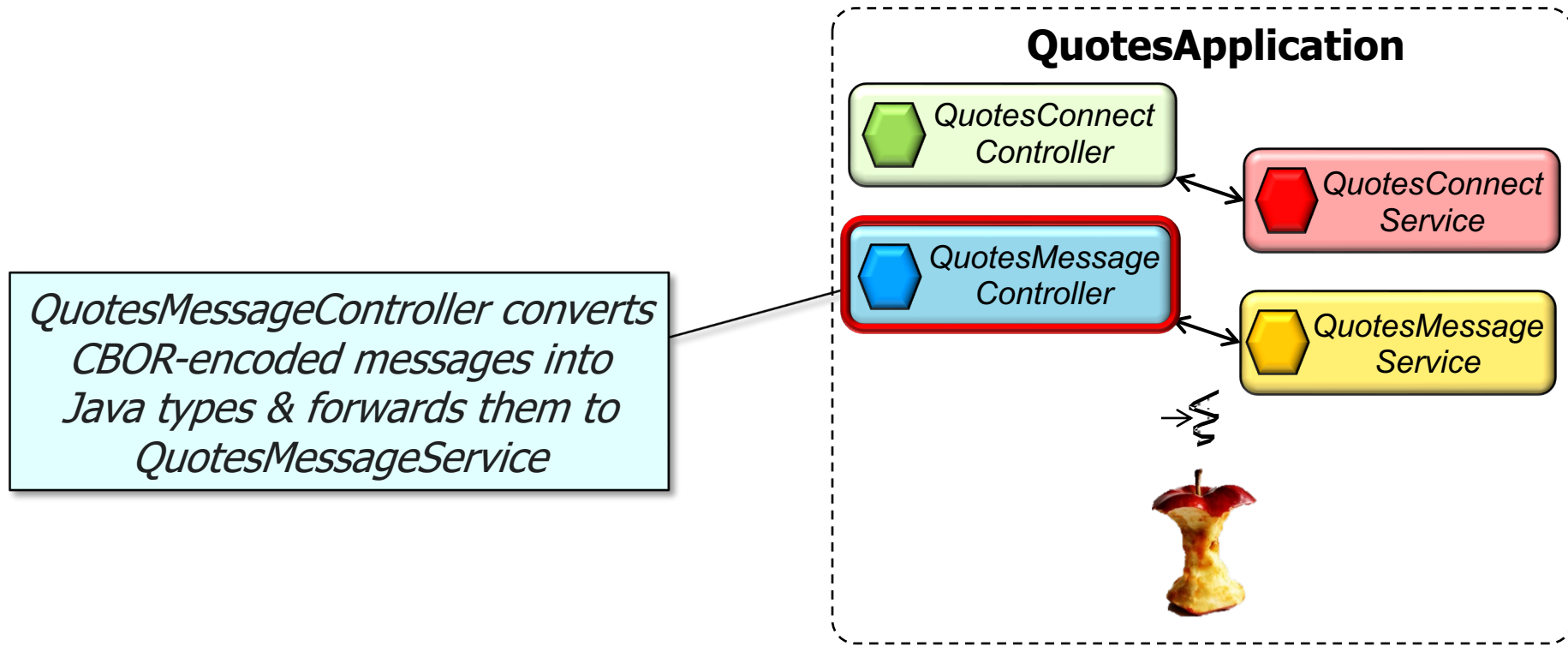
- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice



RSocket & Spring perform authentication transparently to the server itself

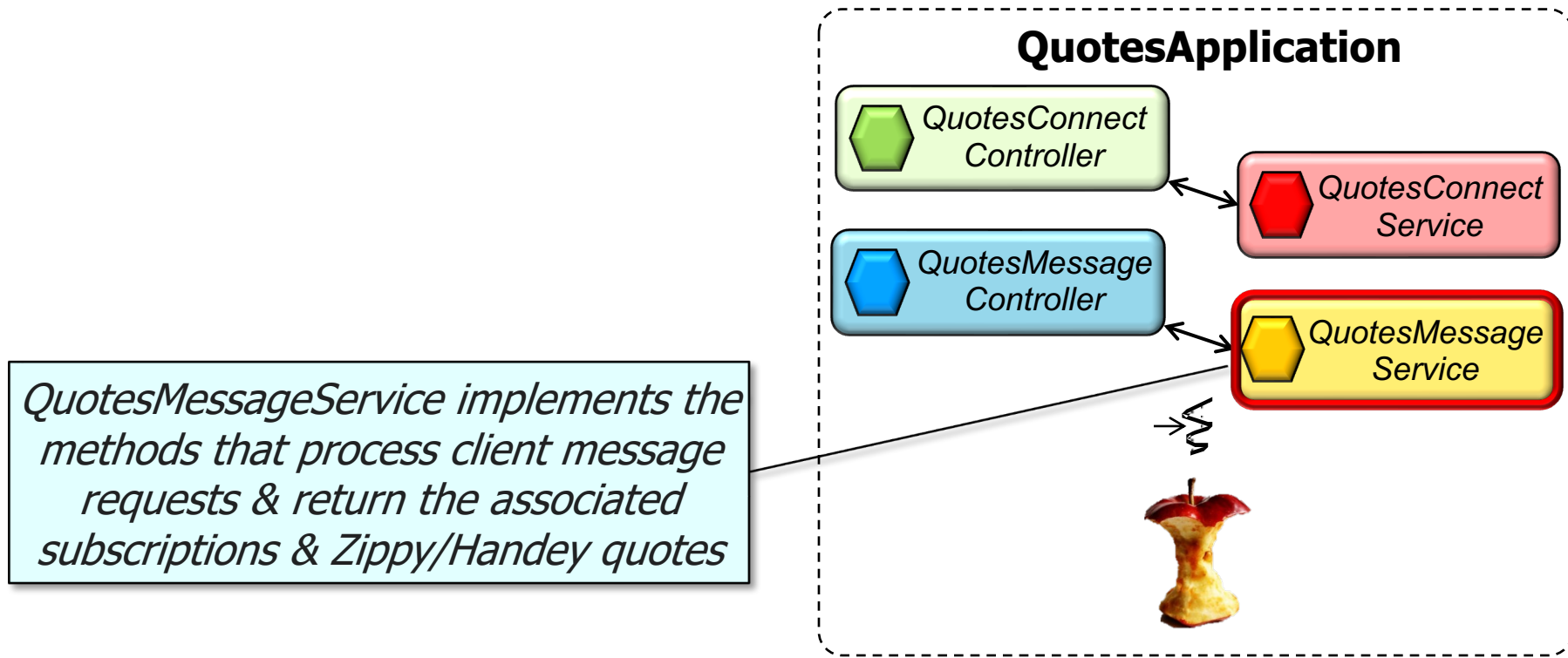
Overview of the RSocket Quotes Backpressure App

- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice



Overview of the RSocket Quotes Backpressure App

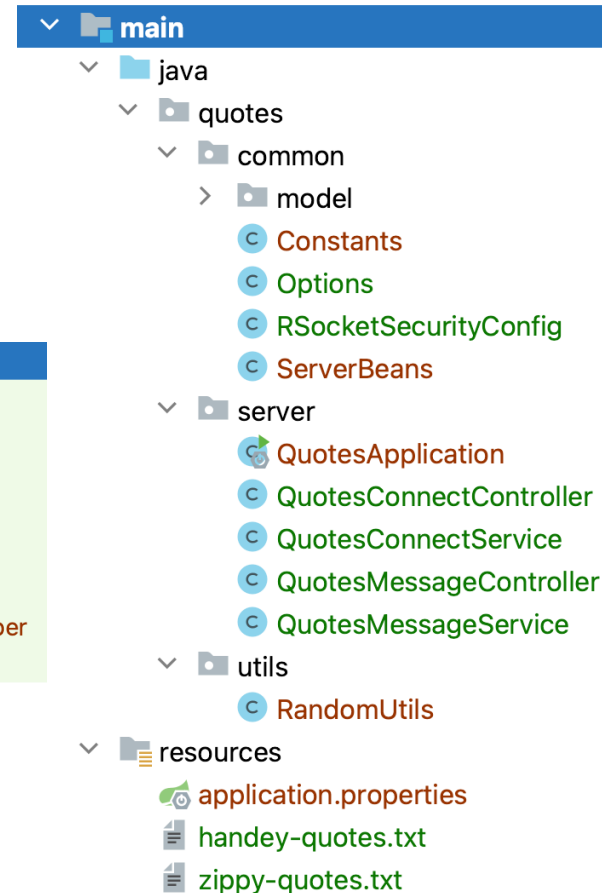
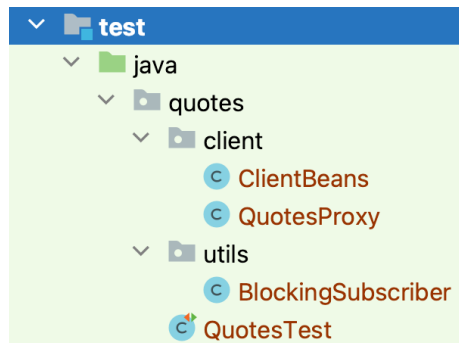
- This case study shows how an RSocket client can exchange binary messages asynchronously with the QuotesApplication microservice



Structure & Functionality of the RSocket Quotes App

Structure & Functionality of the RSocket Quotes App

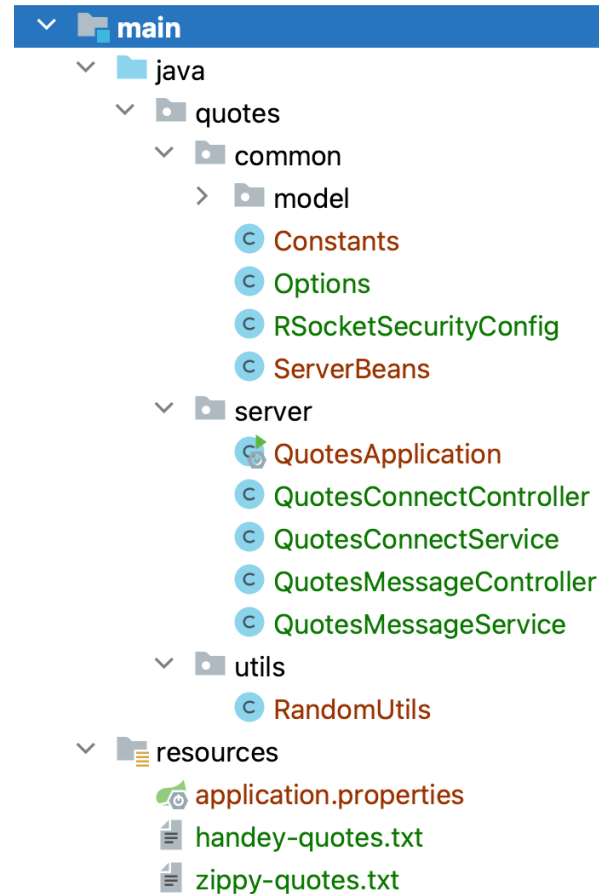
- The RSocket Quotes backpressure App project source code is organized into several packages



See github.com/douglasraigschmidt/LiveLessons/tree/master/RSocket/ex2

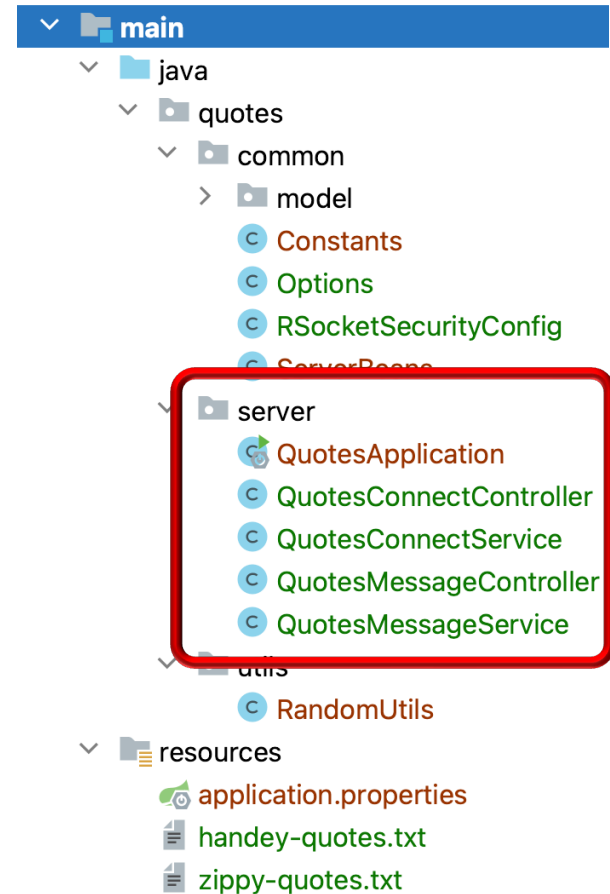
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes



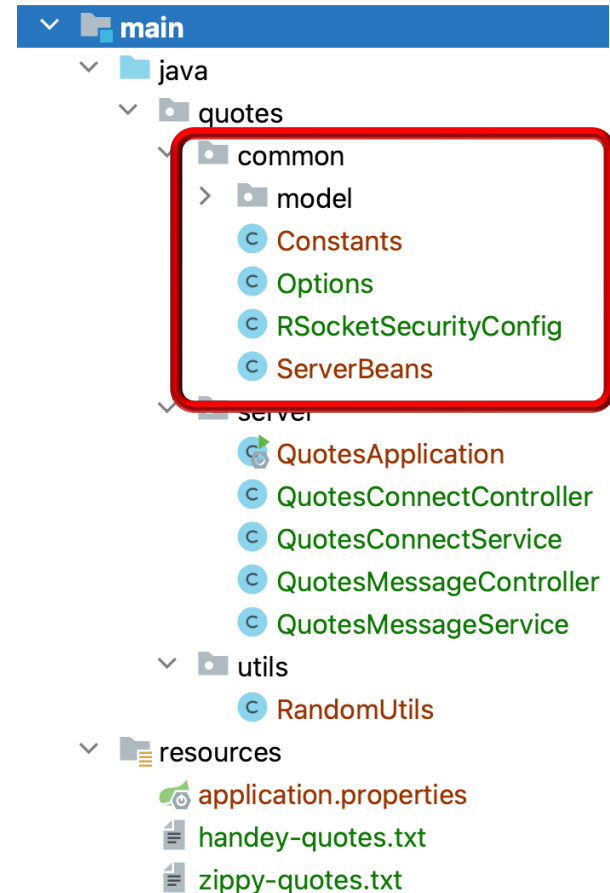
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - server
 - The Application, Controller, & Service classes that enable message reception & responses



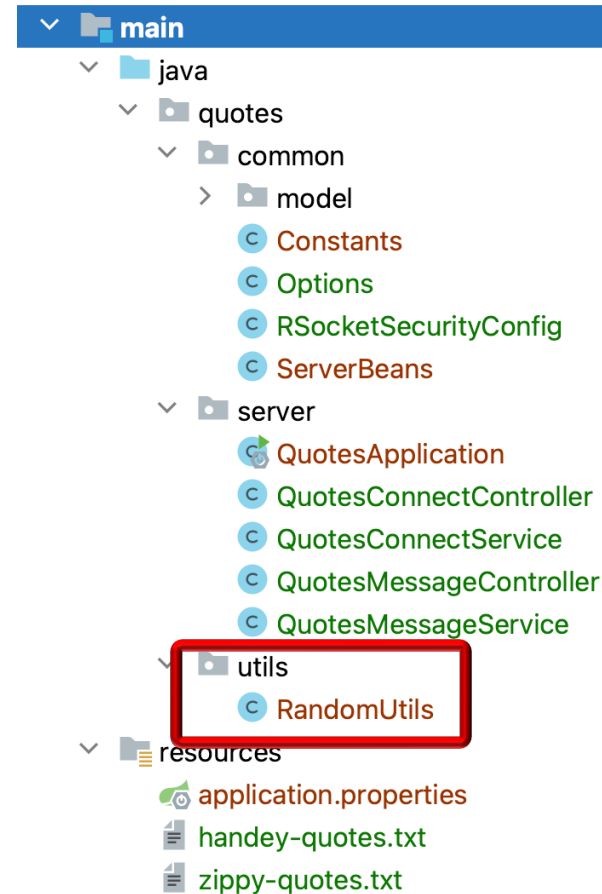
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - server
 - common
 - The project-specific reusable classes, including a security module



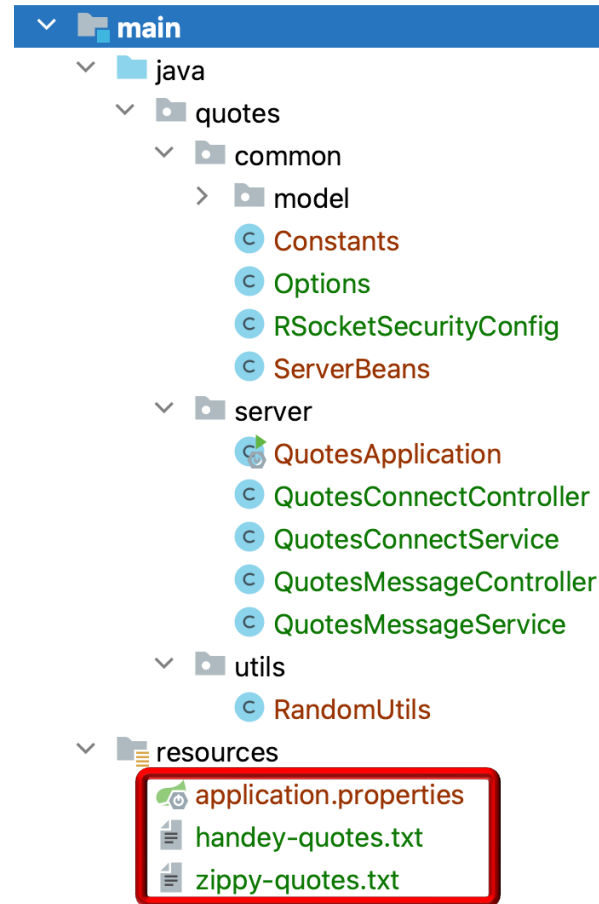
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - server
 - common
 - utils
 - The project-independent reusable classes



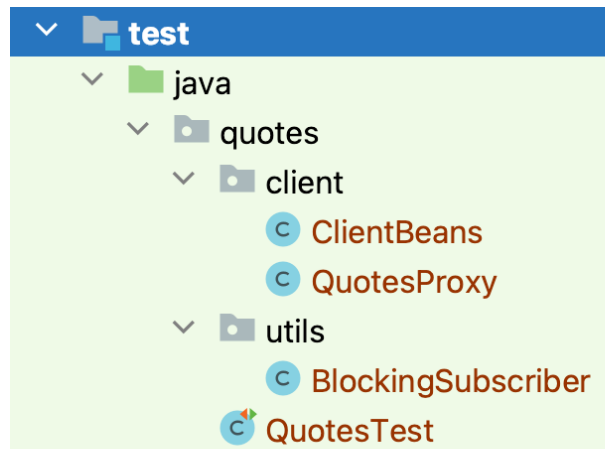
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - server
 - common
 - utils
 - resources
 - The server name, port number, & Zippy/Handey quotes



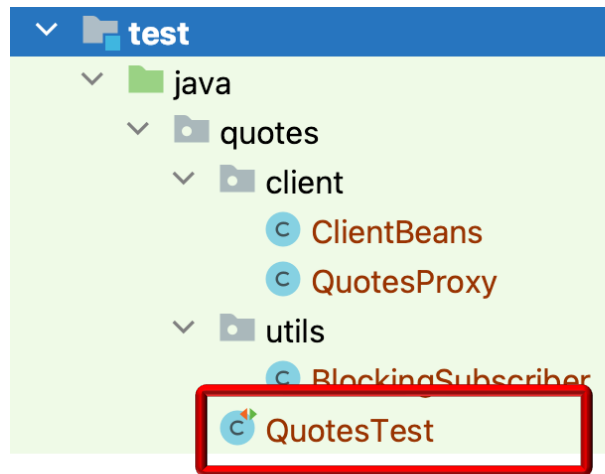
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - The test folder contains the client-side classes



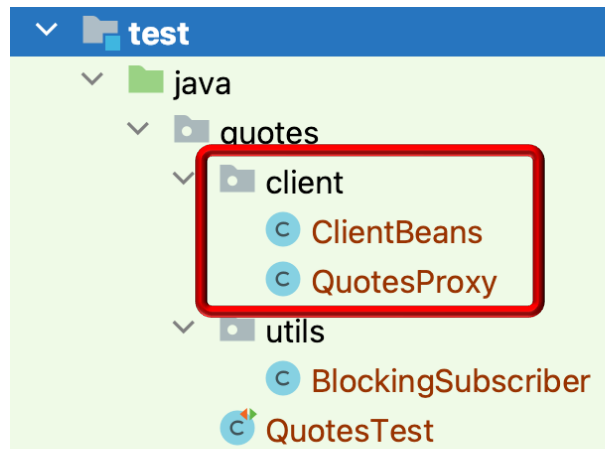
Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - The test folder contains the client-side classes
 - The test driver that connects with the server & exchanges messages



Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - The test folder contains the client-side classes
 - The test driver that connects with the server & exchanges messages
 - client
 - The proxies & client bean that establishes a secure connection with the server



Structure & Functionality of the RSocket Quotes App

- The RSocket Quotes backpressure App project source code is organized into several packages
 - The main folder contains the server-side classes
 - The test folder contains the client-side classes
 - The test driver that connects with the server & exchanges messages
 - client
 - utils
 - A project-independent reusable class that enables backpressure



End of Overview of the RSocket Quotes Backpressure App