The Structure & Functionality of the RSocket Shakespeare Quotes Requester

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

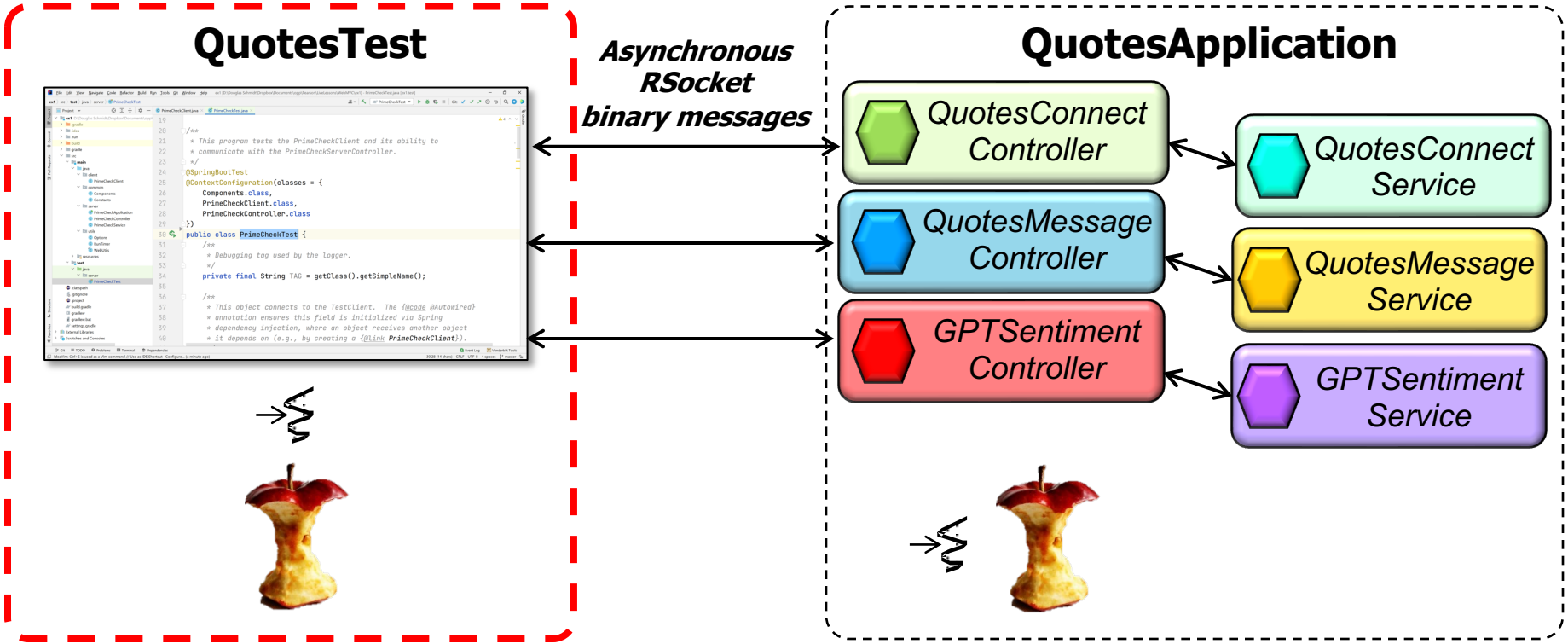
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Understand the structure & functionality of the RSocket Quotes requester that asynchronously connects to & exchanges messages with the responder

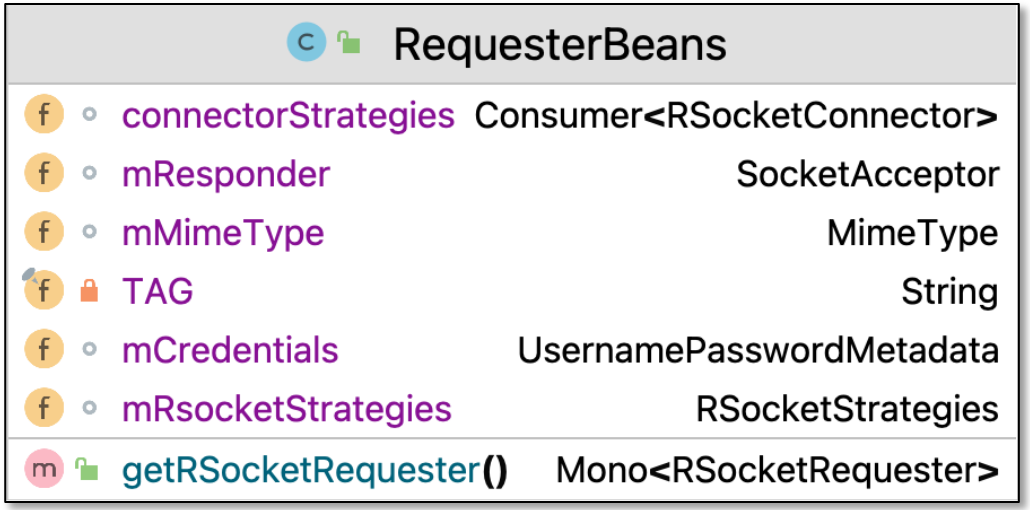


See github.com/douglascraigschmidt/LiveLessons/tree/master/RSocket/ex3

Overview of the Quotes Requester Connection Initiator

Overview of the Quotes Requester Connection Initiator

- The RequesterBeans class defines a @Bean that returns a Mono emitting an RSocketRequester connected to the responder



See RSocket/ex3/src/test/java/quotes/requester/RequesterBeans.java

Overview of the Quotes Requester Connection Initiator

- To obtain an RSocketRequester on the requester side involves a fluent multi-step chain of calls

```
Mono<RSocketRequester>
            getRequester(...) {
  return Mono
    .just(RSocketRequester
        .builder()
        .dataMimeType(...)
        .rsocketStrategies(...)
        .rsocketConnector(...)
        .setupRoute(...)
        .setupData(...)
        .setupMetadata(...)
        .tcp(...));
```

See en.wikipedia.org/wiki/Fluent_interface

Overview of the Quotes Requester Connection Initiator

- To obtain an RSocketRequester on the requester side involves a fluent multi-step chain of calls

```
Mono<RSocketRequester>
                getRequester(...) {
  return Mono
    .just(RSocketRequester
          .builder()
          ...
```

Obtain a builder to create a requester Rsocket Requester by connecting to an RSocket responder

See springframework/messaging/rsocket/RSocketRequester.html#builder

Overview of the Quotes Requester Connection Initiator

- To obtain an RSocketRequester on the requester side involves a fluent multi-step chain of calls
  - The mime type for data on the connection can be set

```
Mono<RSocketRequester>
                getRequester(...) {
  return Mono
    .just(RSocketRequester
          .builder()

          .dataMimeType
             (MediaType
              .APPLICATION_CBOR)
    ...
```

Use the concise binary object representation (CBOR) protocol

See cbor.io

Overview of the Quotes Requester Connection Initiator

- RSocketRequester.Builder accepts RSocketStrategies to configure the requester

```
Mono<RSocketRequester>
            getRequester(...) {
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .rsocketStrategies
            (rSsocketStrategies)
          ...
```

Select the Jackson CBOR encoder/decode as the protocol

```
var rSocketStrategies = RSocketStrategies
   .builder()
   .encoders(encoders -> encoders.add(new Jackson2CborEncoder()))
   .decoders(decoders -> decoders.add(new Jackson2CborDecoder()))
   .encoder(new SimpleAuthenticationEncoder())
   .build();
```

See springframework/http/codec/cbor/Jackson2CborEncoder.html

Overview of the Quotes Requester Connection Initiator

- RSocketRequester.Builder accepts RSocketStrategies to configure the requester

```
Mono<RSocketRequester>
            getRequester(...) {
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .rsocketStrategies
            (rSsocketStrategies)
          ...
```

Sends requester's authentication credentials to the responder

```
var rSocketStrategies = RSocketStrategies
   .builder()
   .encoders(encoders -> encoders.add(new Jackson2CborEncoder()))
   .decoders(decoders -> decoders.add(new Jackson2CborDecoder()))
   .encoder(new SimpleAuthenticationEncoder())
   .build();
```

See springframework/security/rsocket/metadata/SimpleAuthenticationEncoder.html

Overview of the Quotes Requester Connection Initiator

- RSocketRequester.Builder provides a callback that exposes the under lying RSocketConnector for further configuration options

```
Mono<RSocketRequester>
            getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .rsocketConnector(
               connectorStrategies
               ::accept)
          ...
```

```
Consumer<RSocketConnector> connectorStrategies =
    connector -> connector
    .reconnect(Retry.fixedDelay(2, ofSeconds(2)))
    .acceptor(mResponder);
```

Overview of the Quotes Requester Connection Initiator

- RSocketRequester.Builder provides a callback that exposes the under lying RSocketConnector for further configuration options, e.g.
  - Keepalive intervals, interceptors, re-connection policy, acceptors, etc.

```
Mono<RSocketRequester>
            getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .rsocketConnector(
               connectorStrategies
              ::accept)
          ...
```

Try to reconnect twice, with a delay of 2 seconds per retry

```
Consumer<RSocketConnector> connectorStrategies =
    connector -> connector
    .reconnect(Retry.fixedDelay(2, ofSeconds(2)))
    .acceptor(mResponder);
```

Overview of the Quotes Requester Connection Initiator

- RSocketRequester.Builder provides a callback that exposes the under lying RSocketConnector for further configuration options, e.g.
  - Keepalive intervals, interceptors, re-connection policy, acceptors, etc.

```
SocketAcceptor mResponder =
  RSocketMessageHandler
    .responder(mRsocketStrategies,
               new
      ConnectResponseHandler());
```

Receives responder's response to the connection request

```
Consumer<RSocketConnector> connectorStrategies =
    connector -> connector
    .reconnect(Retry.fixedDelay(2, ofSeconds(2)))
    .acceptor(mResponder);
```

See javadoc.io/static/io.rsocket/rsocket-core/1.1.1/io/rsocket/SocketAcceptor.html

Overview of the Quotes Requester Connection Initiator

- There are also methods the requester uses to connect with the responder & authenticate itself

```
Mono<RSocketRequester>
            getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .setupRoute
              (responder_CONNECT)
          ...
```

Set up the route to connect with the responder

See earlier discussion about @ConnectMapping

Overview of the Quotes Requester Connection Initiator

- There are also methods the requester uses to connect with the responder & authenticate itself

```
Mono<RSocketRequester>
            getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .setupMetadata
              (mCreds, mMimeType)
          ...
```

```
UsernamePasswordMetadata mCreds
  = new UsernamePasswordMetadata
    ("d.schmidt@vanderbilt.edu",
     "you-shall-not-pass");

MimeType mMimeType = MimeTypeUtils
  .parseMimeType(MESSAGE_RSOCKET_AUTHENTICATION.getString());
```

Set up the metadata to pass the credentials

Overview of the Quotes Requester Connection Initiator

- There are also methods the requester uses to connect with the responder & authenticate itself

```
Mono<RSocketRequester>
            getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .setupData(UUID
                       .randomUUID()
                       .toString())
          ...
```

Set up the data payload to send the responder initially

Overview of the Quotes Requester Connection Initiator

- Finally, the chain must connect with the responder to obtain a Mono to an RSocketRequester

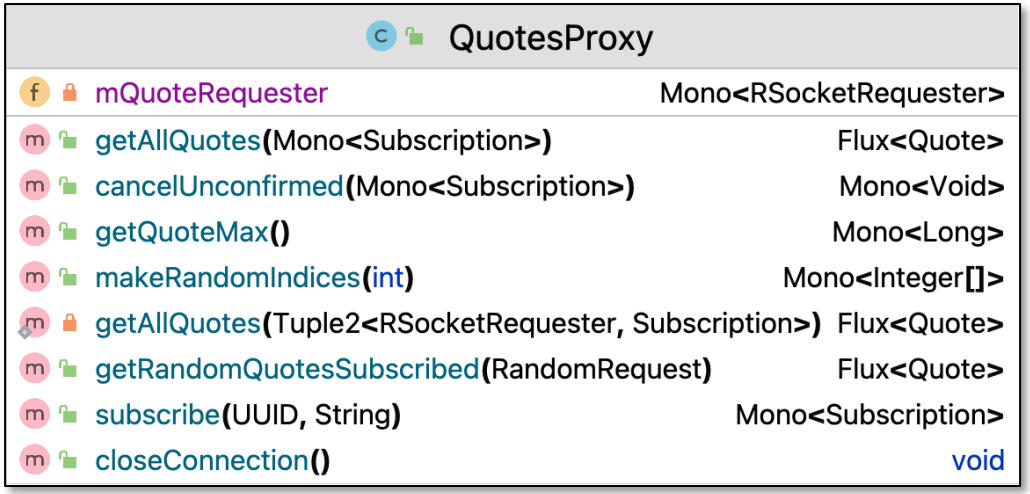
```
Mono<RSocketRequester>
                getRequester(...) {
  ...
  return Mono
    .just(RSocketRequester
          .builder()
          ...
          .tcp(LOCAL_HOST,
               RESPONDER_PORT));
```

Build a requester that's connected to the localhost via TCP RESPONDER_PORT

Overview of the QuotesProxy Class

Overview of the QuotesProxy Class

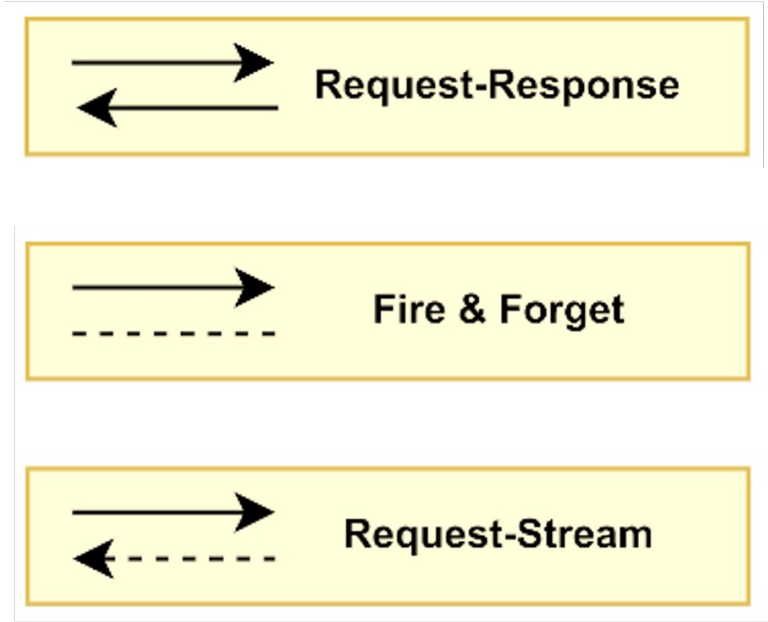
- This proxy defines methods used to send messages to endpoints provided by the QuotesApplication



See RSocket/ex3/src/test/java/quotes/requester/QuotesProxy.java

Overview of the QuotesProxy Class

- This proxy defines methods used to send messages to endpoints provided by the QuotesApplication
  - These methods demo these three interaction models supported by RSocket



Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created

```
class QuotesProxy {
  @Autowired
  private Mono<RSocketRequester>
    mQuoteRequester;
  ...
```

The RSocketRequester is autowired in QuotesProxy

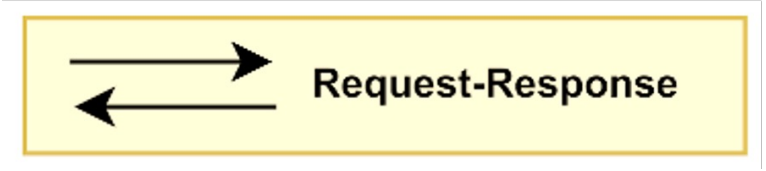
See RSocket/ex3/src/test/java/quotes/requester/QuotesProxy.java

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                      (uuid, PENDING,
                       play)))

      .flatMap(r -> r
               .retrieveMono
                 (Subscription.class))

      .cache();
}
```

Create & return a Subscription that the responder confirms



Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                      (uuid, PENDING,
                       play)))

      .flatMap(r -> r
              .retrieveMono
                (Subscription.class))

      .cache();
}
```

Create metadata for a message containing data sent to the responder to subscribe for quotes

See springframework/messaging/rsocket/RSocketRequester.html#route

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                 (uuid, PENDING,
                  play)))

      .flatMap(r -> r
               .retrieveMono
                 (Subscription.class))

      .cache();
}
```

Create data for a message containing a subscription id & the desired Shakespeare play

See springframework/messaging/rsocket/RSocketRequester.RequestSpec.html#data

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                    (uuid, PENDING,
                     play)))

      .flatMap(r -> r
              .retrieveMono
                (Subscription.class))

      .cache();
}
```

Send a message to the responder to subscribe w/ the subscription & return a confirmed Subscription

See springframework/messaging/rsocket/RSocketRequester.RetrieveSpec.html#retrieveMono

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                      (uuid, PENDING,
                       play)))

      .flatMap(r -> r
              .retrieveMono
                (Subscription.class))

      .cache();
}
```

Project Reactor operators are applied on Mono & Flux types to initiate & handle two-way message passing

See spring.io/blog/2016/04/19/understanding-reactive-types

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()

```
Mono<Subscription> subscribe
   (UUID uuid, String play) {
   return mQuoteRequester
      .map(r -> r
           .route(SUBSCRIBE)
           .data(new Subscription
                      (uuid, PENDING,
                       play)))

      .flatMap(r -> r
              .retrieveMono
                (Subscription.class))

      .cache();
}
```

Turn this Mono into a hot source & cache last emitted signals for future subscribers

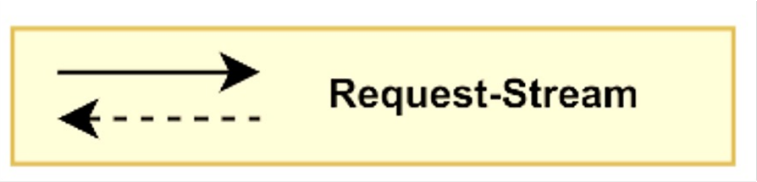
See projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#cache

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()
  - The proxy method get RandomQuotesSubscribed()

```
Flux<Quote>
getRandomQuotesSubscribed
   (RandomRequest randomRequest) {
   return mQuoteRequester
     .map(r -> r
          .route
              (GET_QUOTES_SUBSCRIBED)

          .data(randomRequest))

     .flatMapMany(r -> r
        retrieveFlux(Quote.class));
}
```

Returns a Flux that emits random Bard Quotes that are associated with a Subscription



Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()
  - The proxy method get RandomQuotesSubscribed()

```
Flux<Quote>
getRandomQuotesSubscribed
   (RandomRequest randomRequest) {
   return mQuoteRequester
      .map(r -> r
            .route
               (GET_QUOTES_SUBSCRIBED)

            .data(randomRequest))

      .flatMapMany(r -> r
         retrieveFlux(Quote.class));
}
```

Create metadata for a message containing data sent to the responder get random quotes

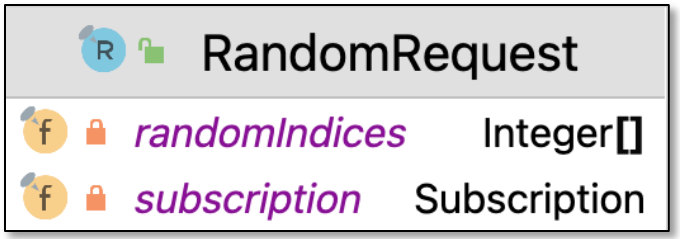
See springframework/messaging/rsocket/RSocketRequester.html#route

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()
  - The proxy method get RandomQuotesSubscribed()

```
Flux<Quote>
getRandomQuotesSubscribed
   (RandomRequest randomRequest) {
   return mQuoteRequester
      .map(r -> r
           .route
              (GET_QUOTES_SUBSCRIBED)

           .data(randomRequest))

      .flatMapMany(r -> r
         retrieveFlux(Quote.class));
}
```

Create data for a message containing a random request



See springframework/messaging/rsocket/RSocketRequester.RequestSpec.html#data

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()
  - The proxy method get RandomQuotesSubscribed()

```
Flux<Quote>
getRandomQuotesSubscribed
   (RandomRequest randomRequest) {
   return mQuoteRequester
      .map(r -> r
           .route
              (GET_QUOTES_SUBSCRIBED)

           .data(randomRequest))

      .flatMapMany(r -> r
        retrieveFlux(Quote.class));
}
```

Send a message to the responder to get a Flux of random Bard Quote objects

See springframework/messaging/rsocket/RSocketRequester.RetrieveSpec.html#retrieveFlux

Overview of the QuotesProxy Class

- Message can be sent after an RSocketRequester is created, e.g.
  - The proxy method subscribe()
  - The proxy method get RandomQuotesSubscribed()

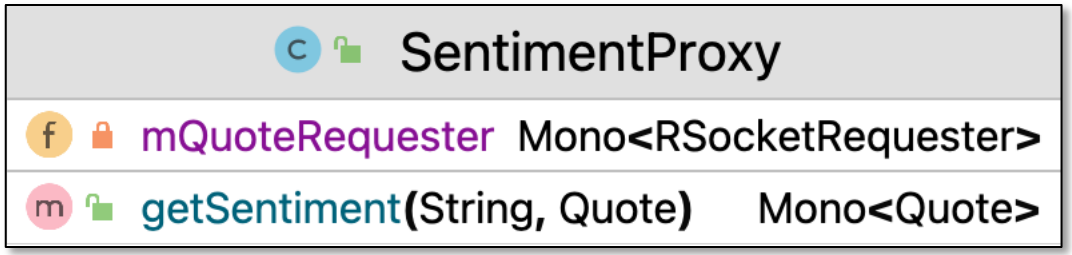
```
Flux<Quote>
getRandomQuotesSubscribed
   (RandomRequest randomRequest) {
   return mQuoteRequester
      .map(r -> r
           .route
              (GET_QUOTES_SUBSCRIBED)

           .data(randomRequest))

      .flatMapMany(r -> r
         retrieveFlux(Quote.class));
}
```

This Flux of Bard Quote objects is then returned to the caller

Overview of the SentimentProxy Class

Overview of the SentimentProxy Class

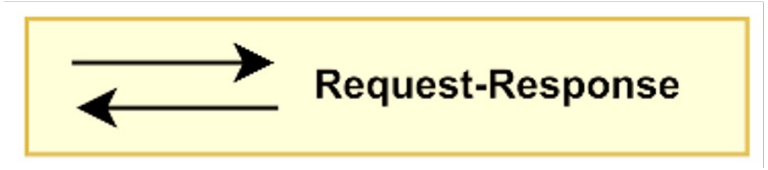
- This proxy defines a method that's used to send messages to the endpoint defined by the GPTSentimentController



See RSocket/ex3/src/test/java/quotes/requester/SentimentProxy.java

Overview of the SentimentProxy Class

- This proxy defines a method that's used to send messages to the endpoint defined by the GPTSentimentController
  - This method demos this RSocket interaction model



Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

```
class SentimentProxy {
  @Autowired
  private Mono<RSocketRequester>
    mQuoteRequester;
  ...
```

The RSocketRequester is autowired in SentimentProxy

See RSocket/ex3/src/test/java/quotes/requester/SentimentProxy.java

Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

Send a Quote to the responder's messagePath endpoint to use AI to analyze its sentiment

```
Mono<Quote> getSentiment
   (String messagePath,
    Quote quote) {
   return mQuoteRequester
      .map(r -> r
            .route(messagePath)
            .data(quote))

      .flatMap(r -> r
                  .retrieveMono
                     (quote.class))
}
```

Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

```
Mono<Quote> getSentiment
   (String messagePath,
    Quote quote) {
   return mQuoteRequester
      .map(r -> r
            .route(messagePath)
            .data(quote))

      .flatMap(r -> r
               .retrieveMono
                  (quote.class))
}
```

Create metadata for a message containing data the responder uses to perform sentiment analysis

See springframework/messaging/rsocket/RSocketRequester.html#route

Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

```
Mono<Quote> getSentiment
   (String messagePath,
    Quote quote) {
   return mQuoteRequester
      .map(r -> r
            .route(messagePath)
            .data(quote))

      .flatMap(r -> r
                .retrieveMono
                   (quote.class))
}
```

Create data for a message containing the Quote from a Shakespeare play to analyze

See springframework/messaging/rsocket/RSocketRequester.RequestSpec.html#data

Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

```
Mono<Quote> getSentiment
   (String messagePath,
    Quote quote) {
   return mQuoteRequester
      .map(r -> r
            .route(messagePath)
            .data(quote))

      .flatMap(r -> r
               .retrieveMono
                  (quote.class))
}
```

Send a message to the responder & retrieve the updated Quote in response

See springframework/messaging/rsocket/RSocketRequester.RetrieveSpec.html#retrieveMono

Overview of the SentimentProxy Class

- Message can be sent after an RSocketRequester is created

```
Mono<Quote> getSentiment
  (String messagePath,
   Quote quote) {
  return mQuoteRequester
    .map(r -> r
          .route(messagePath)
          .data(quote))

    .flatMap(r -> r
              .retrieveMono
                 (quote.class))
}
```

This Mono containing the Bard Quote analyzed for sentiment is then returned to the caller

End of the Structure & Functionality of the RSocket Shakespeare Quotes Requester