Structure & Functionality of the RSocket Shakespeare Quotes Responder (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

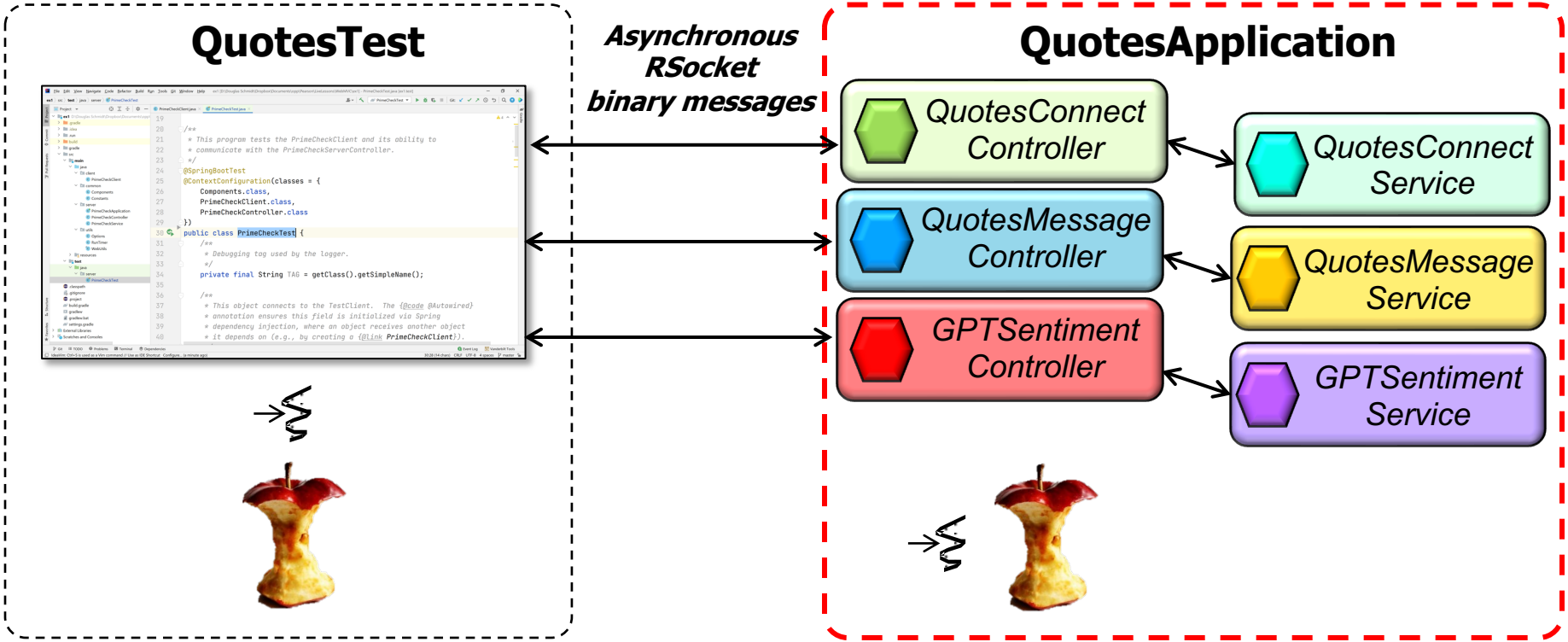
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Lesson

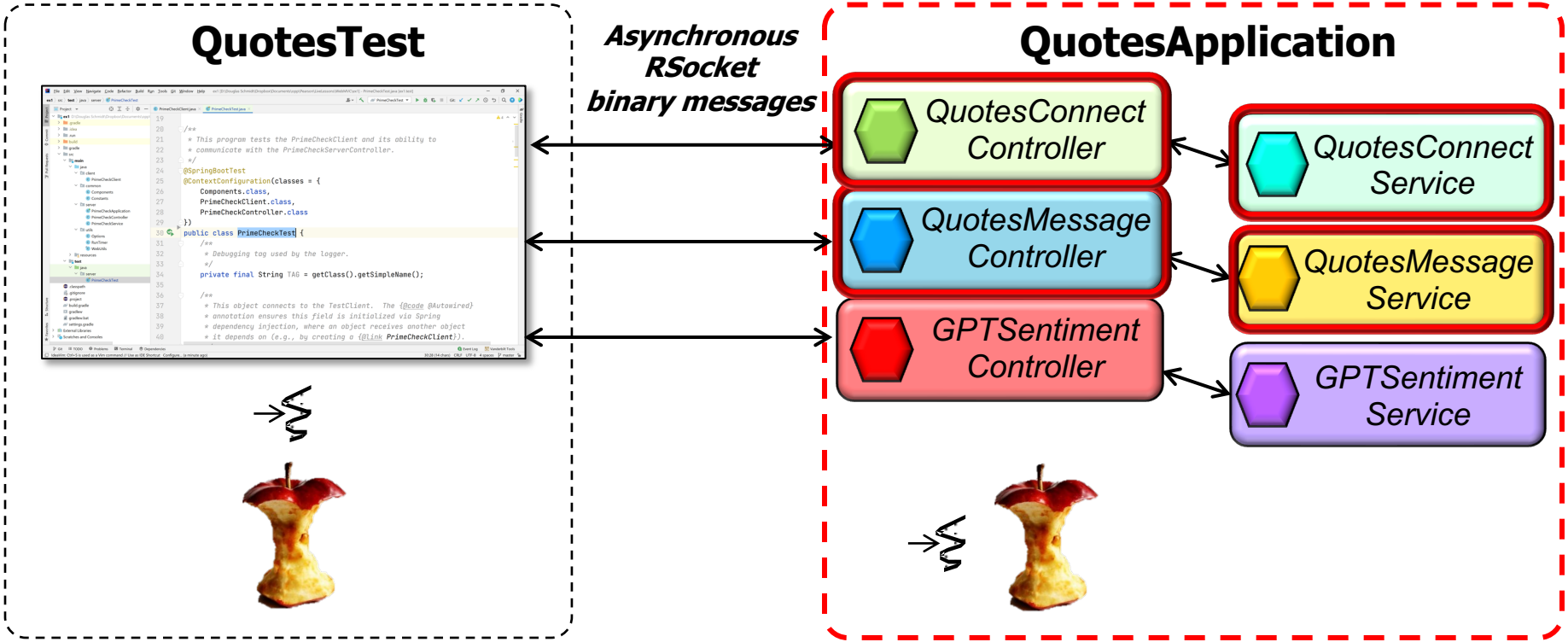
- Understand the structure & functionality of the RSocket Quotes responder that connects with & exchanges binary messages asynchronously w/the requester



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

Learning Objectives in this Lesson

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

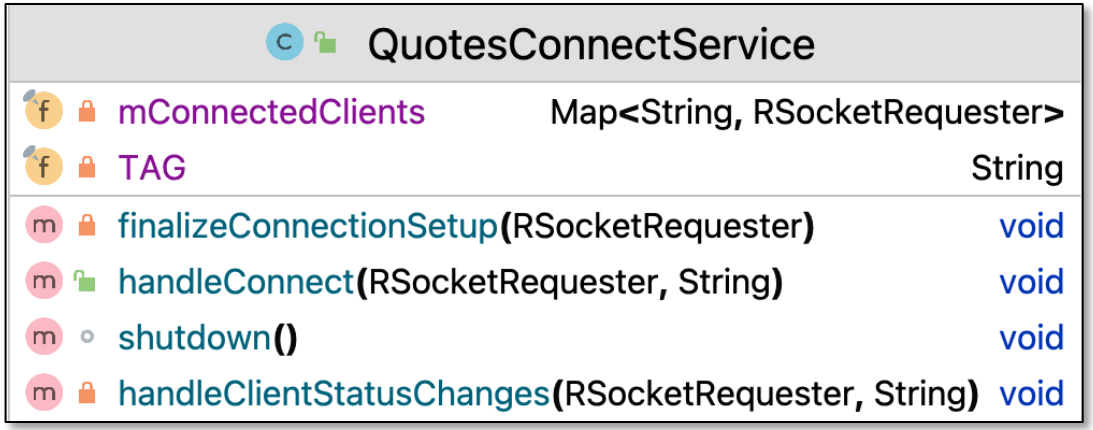


The focus is on the QuotesConnect* & QuotesMessage* classes

Structure & Functionality of the QuotesConnect* Classes

Structure & Functionality of the QuotesConnect* Classes

- The QuotesConnectController & QuotesConnectService handle requester connections



See RSocket/ex3/src/main/java/quotes/responder/connect

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
    (RSocketRequester requester,
     @Payload String requesterIdentity) {
     mService.handleConnect(requester, requesterIdentity);
  }
}
```

See RSocket/ex3/src/main/java/quotes/responder/connect/QuotesConnectController.java

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
     (RSocketRequester requester,
      @Payload String requesterIdentity) {
      mService.handleConnect(requester, requesterIdentity);
  }
}
```

This annotation enables auto-detection of implementation classes via classpath scanning

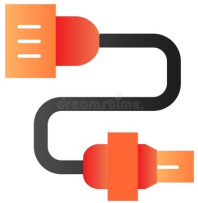
See www.baeldung.com/spring-controllers

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
    (RSocketRequester requester,
     @Payload String requesterIdentity) {
     mService.handleConnect(requester, requesterIdentity);
  }
}
```

This field is auto-wired by Spring's dependency injection framework



See www.baeldung.com/spring-autowire

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
    (RSocketRequester requester,
     @Payload String requesterIdentity) {
     mService.handleConnect(requester, requesterIdentity);
  }
}
```

This hook method is called when a requester connects to the responder

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
    (RSocketRequester requester,
     @Payload String requesterIdentity) {
     mService.handleConnect(requester, requesterIdentity);
  }
}
```

This annotation defines an endpoint that handles connection requests

See springframework/messaging/rsocket/annotation/ConnectMapping.html

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a requester

```
@Controller
public class QuotesConnectController {
  @Autowired
  private QuotesConnectService mService;

  @ConnectMapping(REQUESTER_CONNECT)
  public void handleConnect
    (RSocketRequester requester,
     @Payload String requesterIdentity) {
     mService.handleConnect(requester, requesterIdentity);
  }
}
```

Forwards to the service

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables requesters to connect with the responder securely

```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedrequesters = new ConcurrentHashMap<>();

  public void handleConnect
     (RSocketRequester requester,
      String requesterIdentity) {
      handleRequesterStatusChanges(requester,
                                   requesterIdentity);

      finalizeConnectionSetup(requester);
  }
}
```

See RSocket/ex3/src/main/java/quotes/responder/connect/QuotesConnectService.java

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables requesters to connect with the responder securely

```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedrequesters = new ConcurrentHashMap<>();

  public void handleConnect
    (RSocketRequester requester,
     String requesterIdentity) {
     handleRequesterStatusChanges(requester,
                                  requesterIdentity);
}
```

This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

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

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables requesters to connect with the responder securely

```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedrequesters = new ConcurrentHashMap<>();

  public void handleConnect
    (RSocketRequester requester,
     String requesterIdentity) {
     handleRequesterStatusChanges(requester,
                                  requesterIdentity);

     finalizeConnectionSetup(requester);
  }
}
```

Maintains a map of connected requesters

Connection-related events can occur concurrently

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables requesters to connect with the responder securely

```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedrequesters = new ConcurrentHashMap<>();

  public void handleConnect
    (RSocketRequester requester,
     String requesterIdentity) {
     handleRequesterStatusChanges(requester,
                                  requesterIdentity);

     finalizeConnectionSetup(requester);
  }
}
```

This hook method is called when a requester connects to the responder

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables requesters to connect with the responder securely

```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedrequesters = new ConcurrentHashMap<>();

  public void handleConnect
    (RSocketRequester requester,
     String requesterIdentity) {
     handleRequesterStatusChanges(requester,
                                  requesterIdentity);

     finalizeConnectionSetup(requester);
  }
}
```

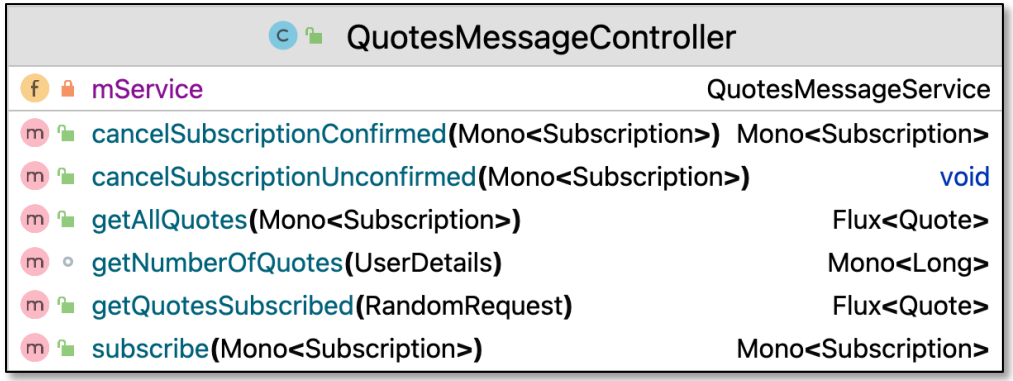
Finalize requester connection setup the & handle requester status changes

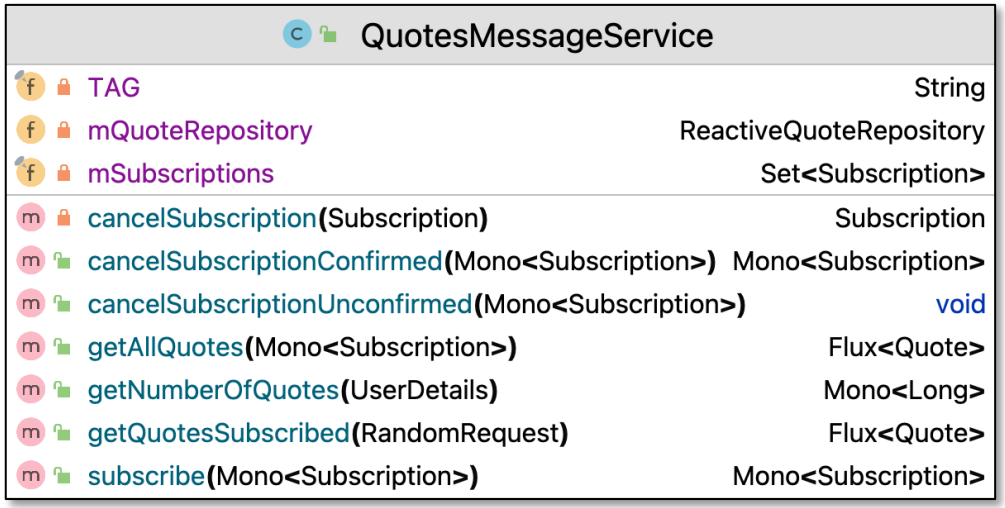
See next part of the lesson for details of these methods

Structure & Functionality of the QuotesMessage* Classes

Structure & Functionality of the QuotesMessage* Classes

- The QuotesMessageController & QuotesMessageService handle requester subscription & quote -related messages





See RSocket/ex3/src/main/java/quotes/responder/quoter

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes

```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

See RSocket/ex3/src/main/java/quotes/responder/quoter/QuotesMessageController.java

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes

```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

This annotation enables auto-detection of implementation classes via classpath scanning

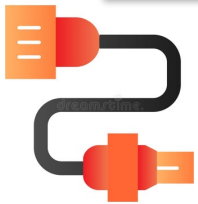
See www.baeldung.com/spring-controllers

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes

```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

This field is auto-wired by Spring's dependency injection framework



See www.baeldung.com/spring-autowire

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes

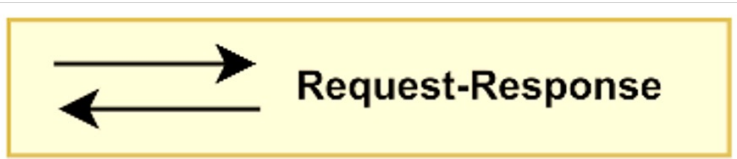
```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

Maps a message to a message-handler by matching the declared patterns to a destination from the message

See springframework/messaging/handler/annotation/MessageMapping.html

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes




```
@Controller
public class QuotesMessageController
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

Confirms a requester's subscription request

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables requesters to subscribe & receive quotes

```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

  @MessageMapping(SUBSCRIBE)
  Mono<Subscription> subscribe
       (Mono<Subscription> subRequest)
  { return mService.subscribe(subRequest); }

  @MessageMapping(GET_QUOTES_SUBSCRIBED)
  Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest)
  { return mService.getQuotesSubscribed(randomRequest); }
  ...
```

Return a Flux emitting random Bard Quotes

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired
  private ReactiveQuoteRepository mQuoteRepository;




  private final Set<Subscription> mSubs = new HashSet<>();
  ...
```

See RSocket/ex3/src/main/java/quotes/responder/quoter/QuotesMessageService.java

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired
  private ReactiveQuoteRepository mQuoteRepository;

  private final Set<Subscription> mSubs = new HashSet<>();
  ...
```

This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

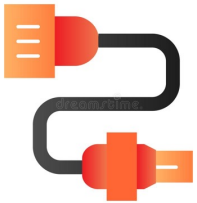
See www.baeldung.com/spring-component-repository-service

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired
  private ReactiveQuoteRepository mQuoteRepository;

  private final Set<Subscription> mSubs = new HashSet<>();
  ...
```

This field is auto-wired by Spring's dependency injection framework



See www.baeldung.com/spring-autowire

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired
  private ReactiveQuoteRepository mQuoteRepository;

  private final Set<Subscription> mSubs = new HashSet<>();
  ...
```

Keep track of the requester subscriptions

This RSocket responder uses a single-threaded event loop

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Mono<Subscription> subscribe(Mono<Subscription> subRequest) {
    return subRequest
        .map(sr -> {
             var subscriptionResponse = new Subscription
               (sr.requestId(), CONFIRMED, sr.play());
             mSubscriptions.add(subscriptionResponse);
             return subscriptionResponse;
    });
}
```

Confirm a requester's subscription request

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Mono<Subscription> subscribe(Mono<Subscription> subReq) {
    return subRequest
        .map(sr -> {
             var subscriptionResponse = new Subscription
               (sr.requestId(), CONFIRMED, sr.play());
             mSubscriptions.add(subscriptionResponse);
             return subscriptionResponse;
    });
}
```

Create a confirmed Subscription & add it to the Set

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Mono<Subscription> subscribe(Mono<Subscription> subReq) {
    return subRequest
        .map(sr -> {
             var subscriptionResponse = new Subscription
               (sr.requestId(), CONFIRMED, sr.play());
             mSubscriptions.add(subscriptionResponse);
             return subscriptionResponse;
    });
}
```

Return the confirmed Subscription

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  ...

 Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest) {
   var subscription = randomRequest
     .subscription();

   return mSubscriptions
      .contains(subscriptions)
         ? mRepository.findAllByIdIn(List.of
                                (randomRequest.randomIndices()))
         : Flux.empty();
  } ...
```

Return a Flux containing random requested quotes

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  ...

 Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest) {
   var subscription = randomRequest
     .subscription();

   return mSubscriptions
      .contains(subscriptions)
         ? mRepository.findAllByIdIn(List.of
                                (randomRequest.randomIndices()))
         : Flux.empty();
  } ...
```

Get the Subscription info

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  ...

 Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest) {
   var subscription = randomRequest
     .subscription();

   return mSubscriptions
      .contains(subscriptions)
         ? mRepository.findAllByIdIn(List.of
                                     (randomRequest.randomIndices()))
         : Flux.empty();
  } ...
```

Return an Empty Flux if the subscription isn't valid

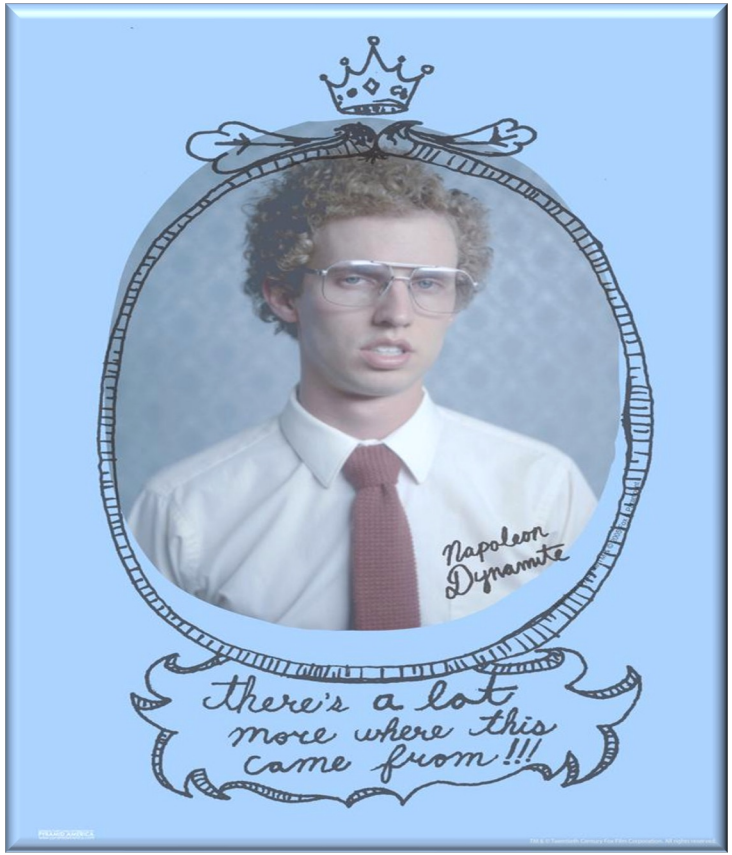
Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  ...

 Flux<Quote> getQuotesSubscribed(RandomRequest randomRequest) {
   var subscription = randomRequest
     .subscription();

   return mSubscriptions
      .contains(subscriptions)
         ? mRepository.findAllByIdIn(List.of
                                     (randomRequest.randomIndices()))
         : Flux.empty();
  } ...
```

Otherwise, return a Flux containing all the random quotes from the R2DBC database

Structure & Functionality of the QuotesMessage* Classes



QuotesMessageController	
mService	QuotesMessageService
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void
getAllQuotes(Mono<Subscription>)	Flux<Quote>
getNumberOfQuotes(UserDetails)	Mono<Long>
getQuotesSubscribed(RandomRequest)	Flux<Quote>
subscribe(Mono<Subscription>)	Mono<Subscription>

QuotesMessageService	
TAG	String
mQuoteRepository	ReactiveQuoteRepository
mSubscriptions	Set<Subscription>
cancelSubscription(Subscription)	Subscription
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void
getAllQuotes(Mono<Subscription>)	Flux<Quote>
getNumberOfQuotes(UserDetails)	Mono<Long>
getQuotesSubscribed(RandomRequest)	Flux<Quote>
subscribe(Mono<Subscription>)	Mono<Subscription>

See next part of the lesson for details of these methods

End of Structure & Functionality of the RSocket Shakespeare Quotes Responder (Part 1)