The Structure & Functionality of the RSocket Quotes Server

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

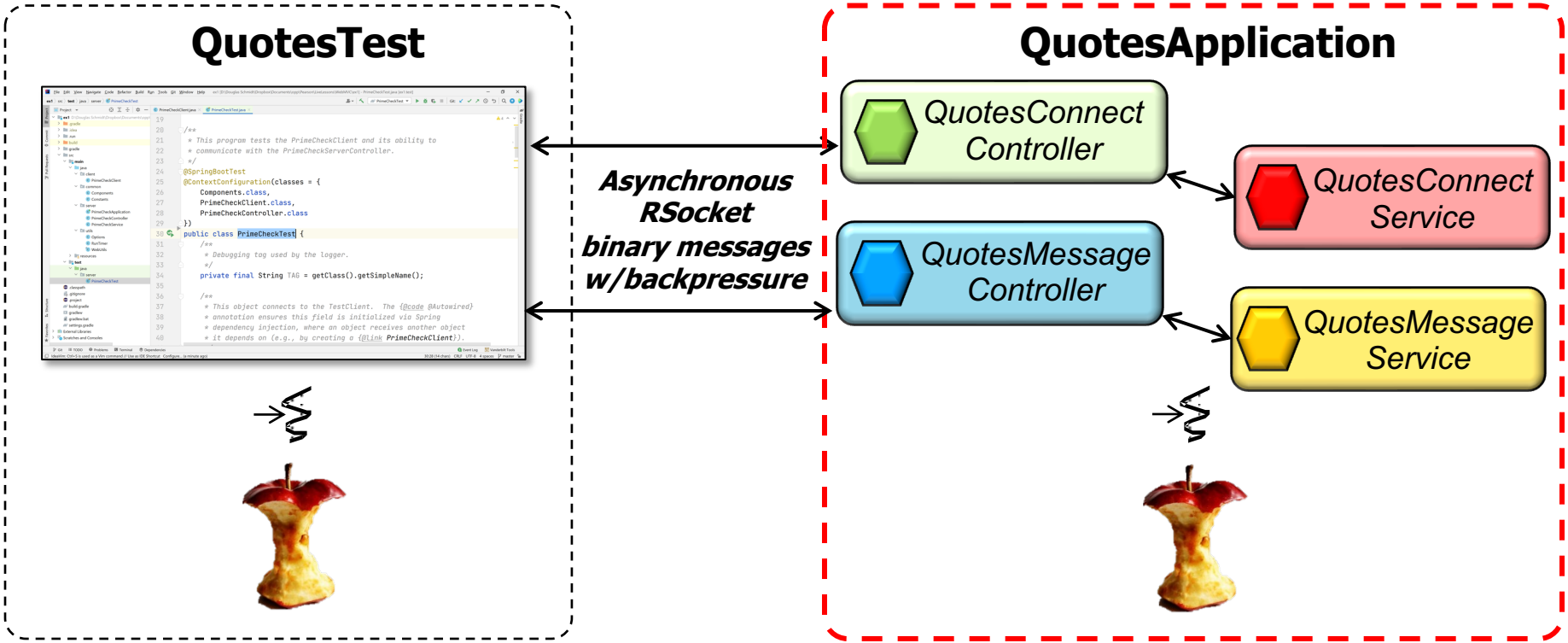
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Understand the structure & functionality of the RSocket Quotes server that connects with & passes binary messages asynchronously via reactive types

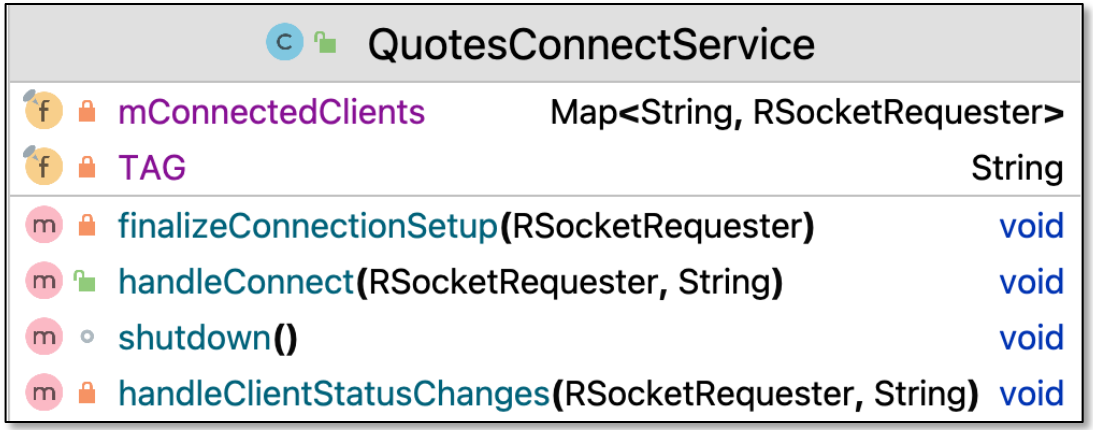


See github.com/douglascraigschmidt/LiveLessons/tree/master/RSocket/ex2

Structure & Functionality of the QuotesConnect* Classes

Structure & Functionality of the QuotesConnect* Classes

- The QuotesConnectController & QuotesConnectService handle client connections



Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
    (RSocketRequester clientRequester,
     @Payload String clientIdentity) {
     mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
     (RSocketRequester clientRequester,
      @Payload String clientIdentity) {
      mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

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

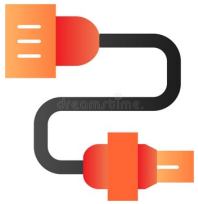
See www.baeldung.com/spring-controllers

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
    (RSocketRequester clientRequester,
     @Payload String clientIdentity) {
     mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

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 client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
    (RSocketRequester clientRequester,
     @Payload String clientIdentity) {
     mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

This hook method is called when a client connects to the server

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectController handles connection requests from a client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
     (RSocketRequester clientRequester,
      @Payload String clientIdentity) {
      mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

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 client

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

  @ConnectMapping(SERVER_CONNECT)
  public void handleConnect
    (RSocketRequester clientRequester,
     @Payload String clientIdentity) {
     mService.handleConnect(clientRequester, clientIdentity);
  }
}
```

Forwards to the service

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables clients to connect with the server securely

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

  public void handleConnect
    (RSocketRequester clientRequester,
     String clientIdentity) {
     handleClientStatusChanges(clientRequester,
                               clientIdentity);

     finalizeConnectionSetup(clientRequester);
  }
}
```

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables clients to connect with the server securely

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

  public void handleConnect
    (RSocketRequester clientRequester,
     String clientIdentity) {
     handleClientStatusChanges(clientRequester,
                               clientIdentity);
}
```

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 clients to connect with the server securely

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

  public void handleConnect
    (RSocketRequester clientRequester,
     String clientIdentity) {
     handleClientStatusChanges(clientRequester,
                               clientIdentity);

     finalizeConnectionSetup(clientRequester);
  }
}
```

Maintains a map of connected clients

Connection-related events can occur concurrently

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables clients to connect with the server securely

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

  public void handleConnect
    (RSocketRequester clientRequester,
     String clientIdentity) {
     handleClientStatusChanges(clientRequester,
                               clientIdentity);

     finalizeConnectionSetup(clientRequester);
  }
}
```

This hook method is called when a client connects to the server

Structure & Functionality of the QuotesConnect* Classes

- QuotesConnectService enables clients to connect with the server securely

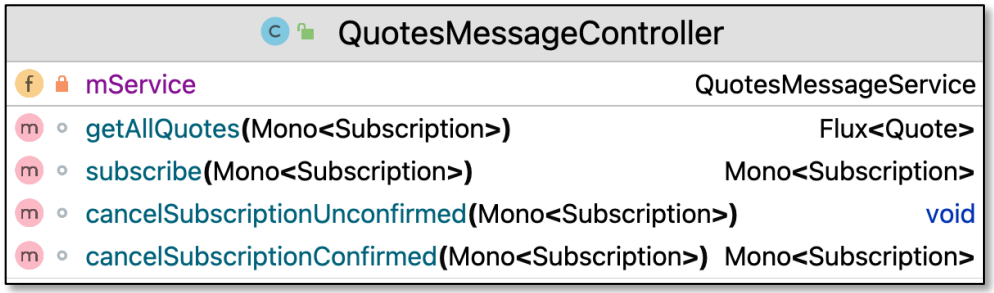
```
@Service
public class QuotesConnectService {
  private final Map<String, RSocketRequester>
    mConnectedClients = new ConcurrentHashMap<>();

  public void handleConnect
     (RSocketRequester clientRequester,
      String clientIdentity) {
      handleClientStatusChanges(clientRequester,
                                clientIdentity);

      finalizeConnectionSetup(clientRequester);
  }
}
```

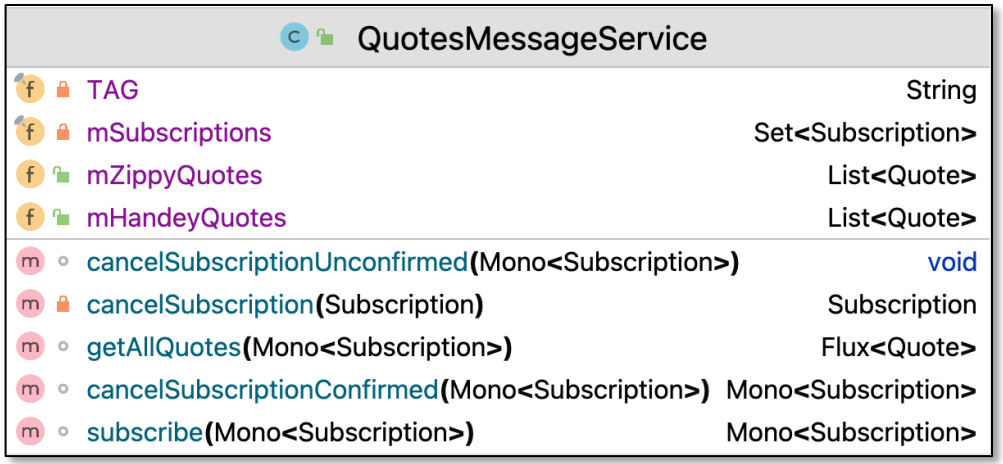
Finalize client connection setup the & handle client status changes

Structure & Functionality of the QuotesMessage* Classes

Structure & Functionality of the QuotesMessage* Classes

- The QuotesMessageController & QuotesMessageService handle client messages





Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables clients 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_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables clients 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_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

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

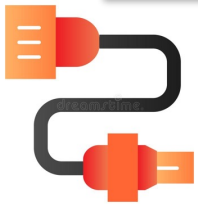
See www.baeldung.com/spring-controllers

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables clients 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_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

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 clients to subscribe & receive quotes

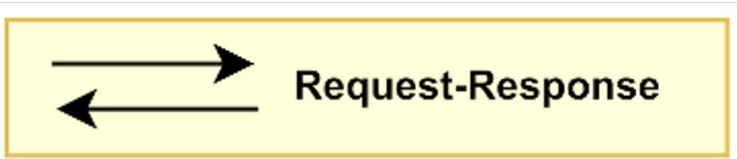
```
@Controller
public class QuotesMessageController {
  @Autowired
  private QuotesMessageService mService;

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

  @MessageMapping(GET_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

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 clients 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_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

Confirms a client's subscription request

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageController enables clients 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_ALL_QUOTES)
  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)
  { return mService.getAllQuotes(quoteIds); }
  ...
```

Return a Flux that emits the specified Quotes quotes

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired @Qualifier(Quotes_QUOTES)
  private List<Quote> mZippyQuotes;

  @Autowired @Qualifier(HANDEY_QUOTES)
  private List<Quote> mHandeyQuotes;




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

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired @Qualifier(Quotes_QUOTES)
  private List<Quote> mZippyQuotes;

  @Autowired @Qualifier(HANDEY_QUOTES)
  private List<Quote> mHandeyQuotes;

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

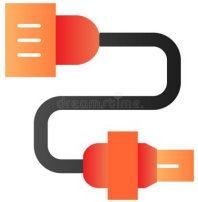
This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

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

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired @Qualifier(Quotes_QUOTES)
  private List<Quote> mZippyQuotes;

  @Autowired @Qualifier(HANDEY_QUOTES)
  private List<Quote> mHandeyQuotes;

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



These fields are 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 @Qualifier(Quotes_QUOTES)
  private List<Quote> mZippyQuotes;

  @Autowired @Qualifier(HANDEY_QUOTES)
  private List<Quote> mHandeyQuotes;


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

This annotation defines a preference when multiple beans of the same type are present

See www.baeldung.com/spring-qualifier-annotation

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  @Autowired @Qualifier(Quotes_QUOTES)
  private List<Quote> mZippyQuotes;

  @Autowired @Qualifier(HANDEY_QUOTES)
  private List<Quote> mHandeyQuotes;

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

Keep track of the client subscriptions

This RSocket server uses a single-threaded event loop

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Mono<Subscription> subscribe(Mono<Subscription> subReq) {
    return subReq.doOnNext(sr -> {
                  sr.setStatus(SubscriptionStatus.CONFIRMED);
                  mSubscriptions.add(sr);
    });
}
```

Confirm a client's subscription request

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Mono<Subscription> subscribe(Mono<Subscription> subReq) {
    return subReq.doOnNext(sr -> {
                    sr.setStatus(SubscriptionStatus.CONFIRMED);
                    mSubscriptions.add(sr);
    });
}
```

Change the subscription's status to CONFIRMED & 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 subReq.doOnNext(sr -> {
                   sr.setStatus(SubscriptionStatus.CONFIRMED);
                   mSubscriptions.add(sr);
    });
}
```

Return the confirmed Subscription

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
  ...

  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest) {
    return subRequest
      .flatMapMany(sr -> mSubs.contains(sr)
        ? Flux.fromIterable(sr.getType() == Quotes
          ? mZippyQuotes : mHandeyQuotes)
        : Flux.empty());
  } ...
```

Return a Flux containing all the designated quotes

Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {
   ...

  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest)  {
    return subRequest
      .flatMapMany(sr -> mSubs.contains(sr)
        ? Flux.fromIterable(sr.getType() == Quotes
          ? mZippyQuotes : mHandeyQuotes)
        : Flux.empty());
  } ...
```

Return an Empty Flux if the subscription isn't valid

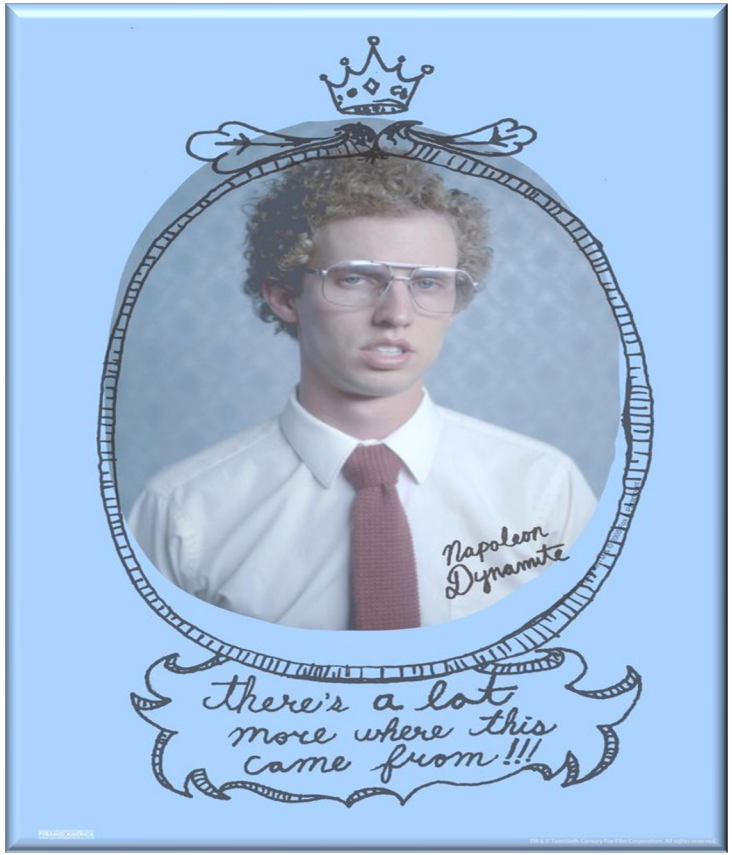
Structure & Functionality of the QuotesMessage* Classes

- QuotesMessageService implements QuotesConnectController endpoints

```
@Service
class QuotesMessageService {}
  ...

  Flux<Quote> getAllQuotes(Mono<Subscription> subRequest) {
    return subRequest
      .flatMapMany(sr -> mSubs.contains(sr)
        ? Flux.fromIterable(sr.getType() == Quotes
          ? mZippyQuotes : mHandeyQuotes
        : Flux.empty());
  } ...
```

Otherwise, return a Flux containing all the designated quotes

Structure & Functionality of the QuotesMessage* Classes



QuotesMessageController	
mService	QuotesMessageService
getAllQuotes(Mono<Subscription>)	Flux<Quote>
subscribe(Mono<Subscription>)	Mono<Subscription>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>

QuotesMessageService	
TAG	String
mSubscriptions	Set<Subscription>
mZippyQuotes	List<Quote>
mHandeyQuotes	List<Quote>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void
cancelSubscription(Subscription)	Subscription
getAllQuotes(Mono<Subscription>)	Flux<Quote>
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>
subscribe(Mono<Subscription>)	Mono<Subscription>

End of the Structure & Functionality of the RSocket Quotes Server