The Structure & Functionality of the RSocket ZippyQuotes Server

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

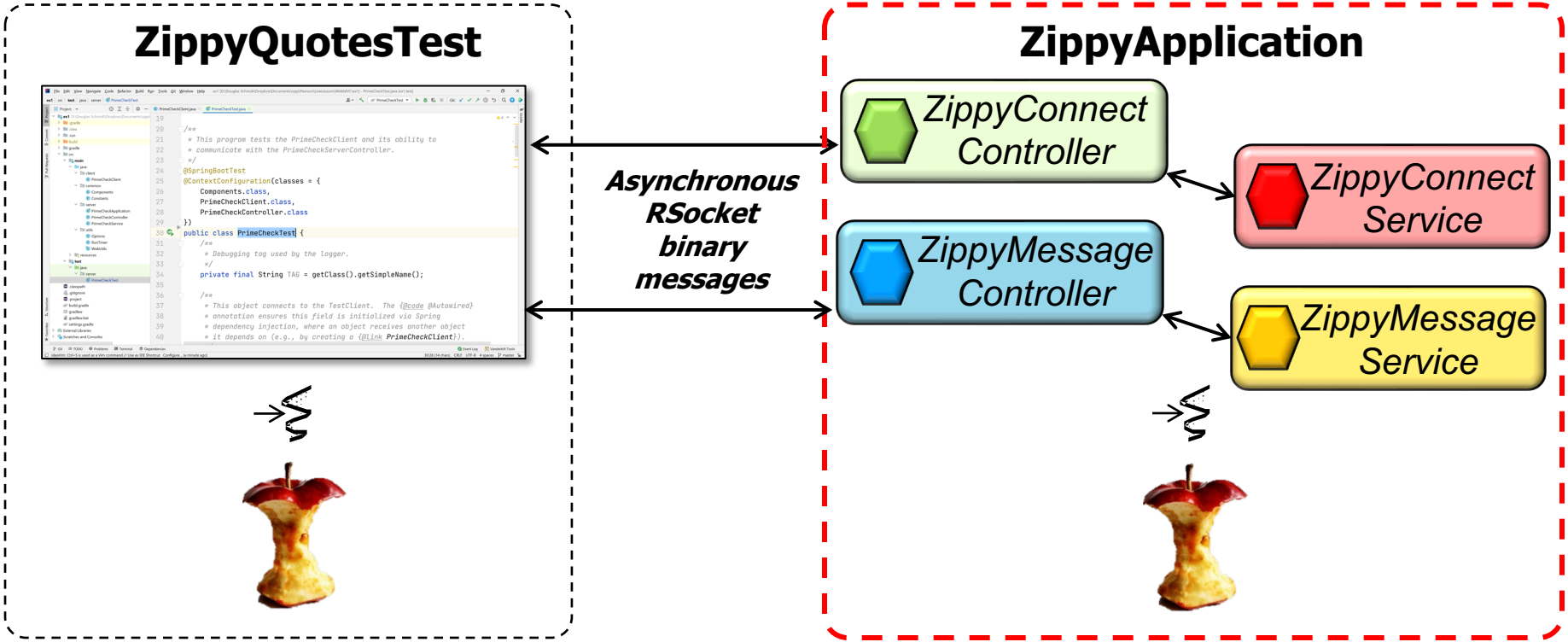
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

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

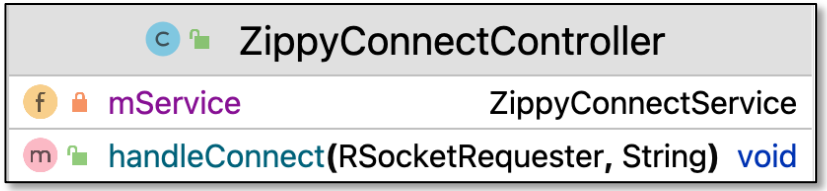


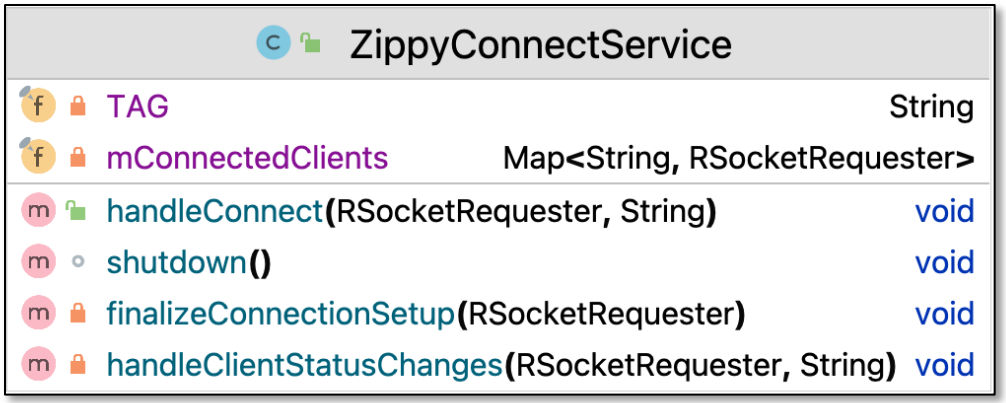
See github.com/douglascraigschmidt/LiveLessons/tree/master/RSocket/ex1

Structure & Functionality of the ZippyConnect* Classes

Structure & Functionality of the ZippyConnect* Classes

- The ZippyConnectController & ZippyConnectService handle client connections





Structure & Functionality of the ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
  @Autowired
  private ZippyConnectService mService;

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

Structure & Functionality of the ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
  @Autowired
  private ZippyConnectService 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

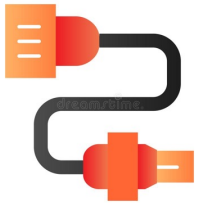
See www.baeldung.com/spring-controllers

Structure & Functionality of the ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
  @Autowired
  private ZippyConnectService 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 ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
   @Autowired
   private ZippyConnectService 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 ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
  @Autowired
  private ZippyConnectService 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 ZippyConnect* Classes

- ZippyConnectController handles connection requests from a client

```
@Controller
public class ZippyConnectController {
  @Autowired
  private ZippyConnectService mService;

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

Forwards to the service

Structure & Functionality of the ZippyConnect* Classes

- ZippyConnectService enables clients to connect with the server securely

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

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

     finalizeConnectionSetup(clientRequester);
  }
}
```

Structure & Functionality of the ZippyConnect* Classes

- ZippyConnectService enables clients to connect with the server securely

```
@Service
public class ZippyConnectService {
  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 ZippyConnect* Classes

- ZippyConnectService enables clients to connect with the server securely

```
@Service
public class ZippyConnectService {
  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 ZippyConnect* Classes

- ZippyConnectService enables clients to connect with the server securely

```
@Service
public class ZippyConnectService {
  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 ZippyConnect* Classes

- ZippyConnectService enables clients to connect with the server securely

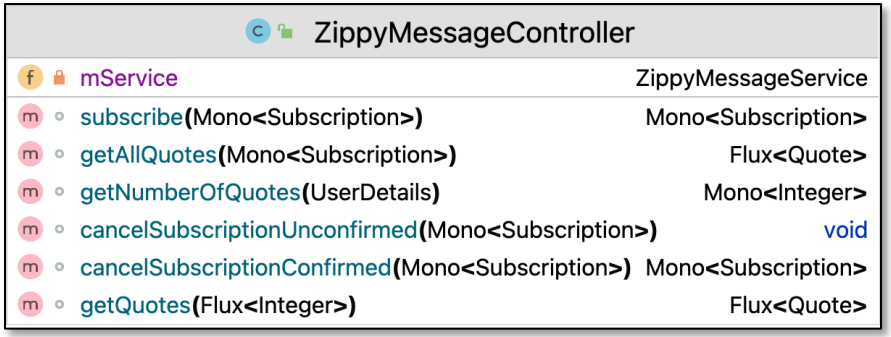
```
@Service
public class ZippyConnectService {
  private final Map<String, RSocketRequester>
    mConnectedClients = new ConcurrentHashMap<>();

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

     finalizeConnectionSetup(clientRequester);
  }
}
```

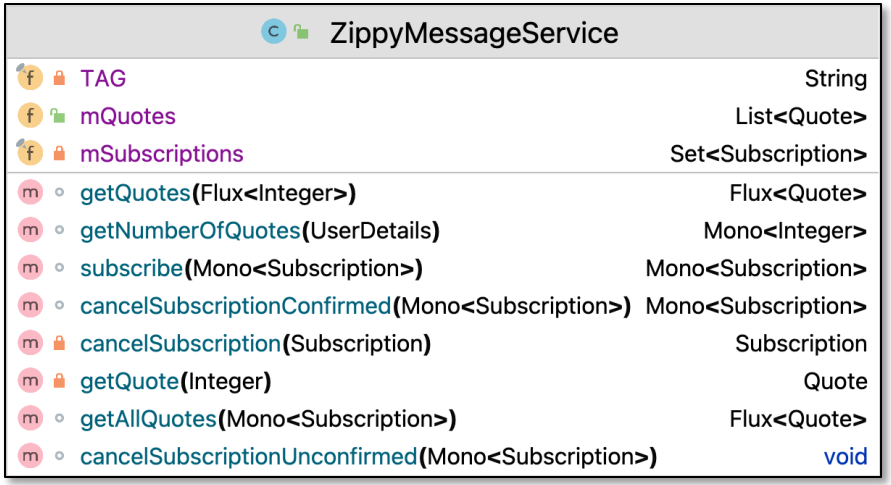
Finalize client connection setup the & handle client status changes

Structure & Functionality of the ZippyMessage* Classes

Structure & Functionality of the ZippyMessage* Classes

- The ZippyMessageController & ZippyMessageService handle client messages





Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes

```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(quoteIds); }
  ...
```

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes

```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(quoteIds); }
  ...
```

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

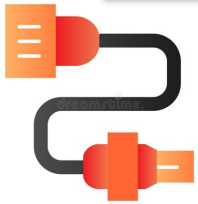
See www.baeldung.com/spring-controllers

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes

```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(quoteIds); }
  ...
```

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



See www.baeldung.com/spring-autowire

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes

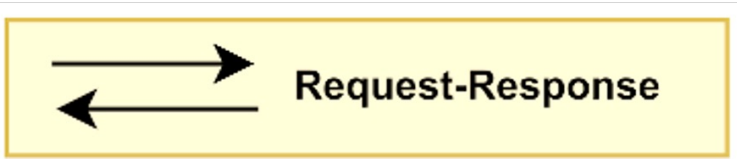
```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(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 ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes



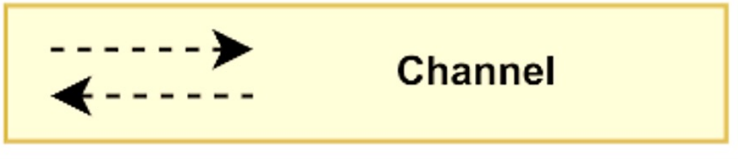

```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(quoteIds); }
  ...
```

Confirms a client's subscription request

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageController enables clients to subscribe & receive Zippy quotes

```
@Controller
public class ZippyMessageController {
  @Autowired
  private ZippyMessageService mService;

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

  @MessageMapping(GET_QUOTES)
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return mService.getQuotes(quoteIds); }
  ...
```



Gets a Flux that emits the specified Zippy quotes

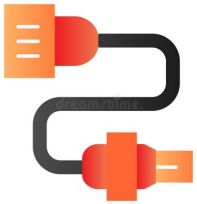
Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageService implements ZippyConnectController endpoint methods

```
@Service
class ZippyMessageService {
  @Autowired private List<Quote> mQuotes;
  private final Set<Subscription> mSubs = new HashSet<>();

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


  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return quoteIds.map(id -> mQuotes.get(id)); } ...
```

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageService implements ZippyConnectController endpoint methods

```
@Service
class ZippyMessageService {
  @Autowired private List<Quote> mQuotes;
  
          final Set<Subscription> mSubs = new HashSet<>();

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


  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return quoteIds.map(id -> mQuotes.get(id)); } ...
```

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

See www.baeldung.com/spring-autowire

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageService implements ZippyConnectController endpoint methods

```
@Service
class ZippyMessageService {
  @Autowired private List<Quote> mQuotes;
  private final Set<Subscription> mSubs = new HashSet<>();

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


  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return quoteIds.map(id -> mQuotes.get(id)); } ...
```

Keep track of client subscriptions

This RSocket server uses a single-threaded event loop

Structure & Functionality of the ZippyMessage* Classes

- ZippyMessageService implements ZippyConnectController endpoint methods

```
@Service
class ZippyMessageService {
  @Autowired private List<Quote> mQuotes;
  private final Set<Subscription> mSubs = new HashSet<>();

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

Confirm the subscription & add it to the Set

```
  Flux<Quote> getQuotes(Flux<Integer> quoteIds)
  { return quoteIds.map(id -> mQuotes.get(id)); } ...
```

Structure & Functionality of the ZippyMessage* Classes

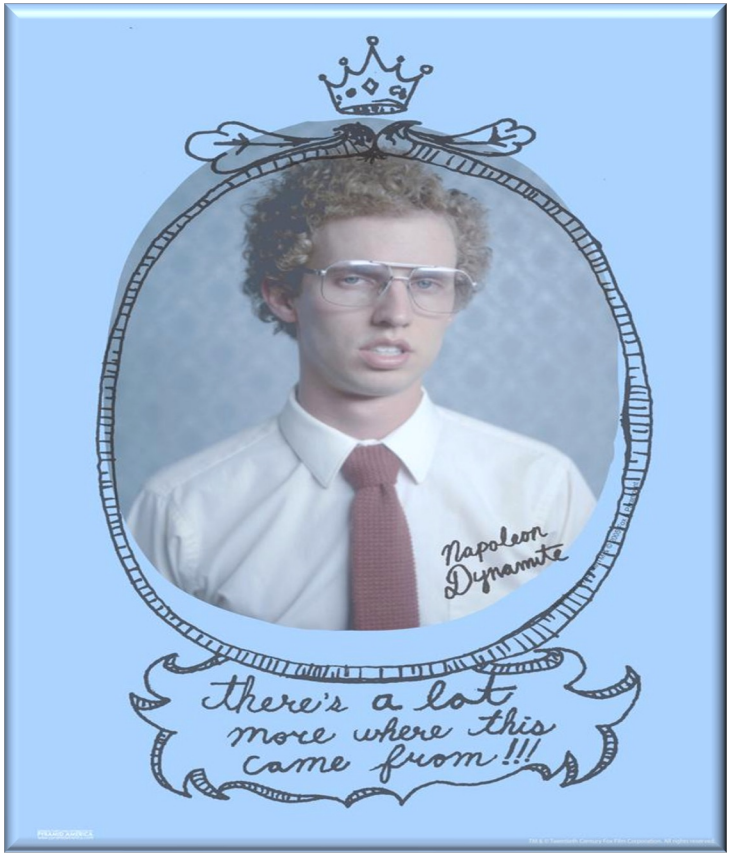
- ZippyMessageService implements ZippyConnectController endpoint methods

```
@Service
class ZippyMessageService {
  @Autowired private List<Quote> mQuotes;
  private final Set<Subscription> mSubs = new HashSet<>();

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

Return a Flux containing the requested IDs

```
Flux<Quote> getQuotes(Flux<Integer> quoteIds)
{ return quoteIds.map(id -> mQuotes.get(id)); } ...
```

Structure & Functionality of the ZippyMessage* Classes



ZippyMessageController	
mService	ZippyMessageService
subscribe(Mono<Subscription>)	Mono<Subscription>
getAllQuotes(Mono<Subscription>)	Flux<Quote>
getNumberOfQuotes(UserDetails)	Mono<Integer>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>
getQuotes(Flux<Integer>)	Flux<Quote>

ZippyMessageService	
TAG	String
mQuotes	List<Quote>
mSubscriptions	Set<Subscription>
getQuotes(Flux<Integer>)	Flux<Quote>
getNumberOfQuotes(UserDetails)	Mono<Integer>
subscribe(Mono<Subscription>)	Mono<Subscription>
cancelSubscriptionConfirmed(Mono<Subscription>)	Mono<Subscription>
cancelSubscription(Subscription)	Subscription
getQuote(Integer)	Quote
getAllQuotes(Mono<Subscription>)	Flux<Quote>
cancelSubscriptionUnconfirmed(Mono<Subscription>)	void

End of the Structure & Functionality of the RSocket ZippyQuotes Server