Structure & Functionality of the RSocket Shakespeare Quotes Responder (Part 2)

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

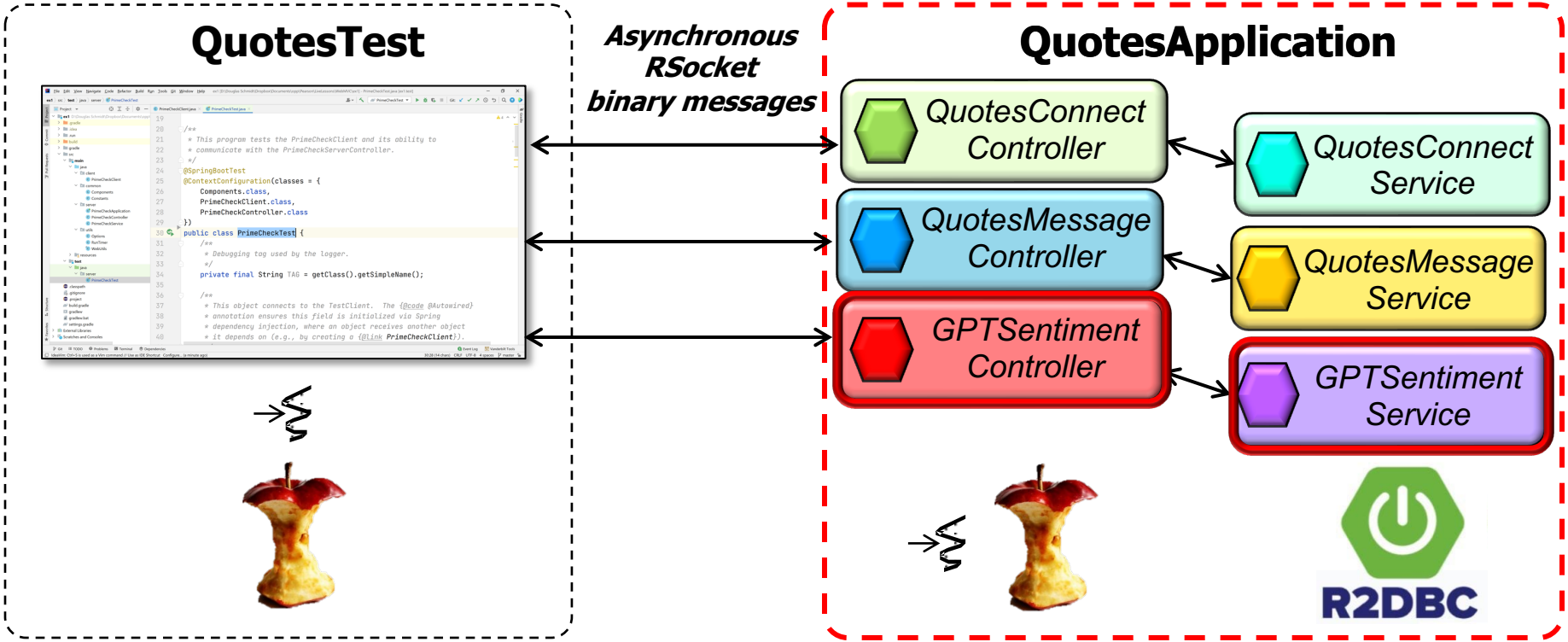
Professor of Computer Science

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 & passes binary messages asynchronously via reactive types



The focus is on the ReactiveQuotesRepository & GPTSentiment* classes

Structure & Functionality of the ReactiveQuoteRepository

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database


```
@Repository public interface
ReactiveQuoteRepository
   extends ReactiveCrudRepository
                  <Quote, Integer> {
    Flux<Quote> findAllByPlay
     (String play);

    Flux<Quote> findAllByIdIn
     (List<Integer> queryIds);
}
```

R2DBC allows programs & programmers to access & interact with relational databases in a reactive & non-blocking manner

See r2dbc.io

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
  extends ReactiveCrudRepository
                <Quote, Integer> {
  Flux<Quote> findAllByPlay
    (String play);

  Flux<Quote> findAllByIdIn
    (List<Integer> queryIds);
}
```

See RSocket/ex2/src/main/java/quotes/repository/ReactiveQuoteRepository.java

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
  extends ReactiveCrudRepository
                <Quote, Integer> {
  Flux<Quote> findAllByPlay
    (String play);

  Flux<Quote> findAllByIdIn
    (List<Integer> queryIds);
}
```

This annotation indicates the ReactiveQuoteRepository class provides mechanisms for storage, retrieval, search, update & delete (CRUD) operation on objects

See org/springframework/stereotype/Repository.html

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
   extends ReactiveCrudRepository
                  <Quote, Integer> {
   Flux<Quote> findAllByPlay
      (String play);

   Flux<Quote> findAllByIdIn
      (List<Integer> queryIds);
}
```

Interface for generic CRUD operations on a repository for a specific type that follows reactive paradigms & uses Project Reactor types built on top of Reactive Streams

See springframework/data/repository/reactive/ReactiveCrudRepository.html

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
   extends ReactiveCrudRepository
                  <Quote, Integer> {
   Flux<Quote> findAllByPlay
     (String play);

   Flux<Quote> findAllByIdIn
     (List<Integer> queryIds);
}
```

Asynchronously find all Quote rows in the database that are associated with the given Shakespeare play

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
  extends ReactiveCrudRepository
                <Quote, Integer> {
  Flux<Quote> findAllByPlay
    (String play);

  Flux<Quote> findAllByIdIn
    (List<Integer> queryIds);
}
```

This Spring Data API method implementation is automatically generated as SELECT * FROM quote WHERE play = (:play)

ChatGPT can generate the SQL statement corresponding to the Spring Data API keywords!

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

```
@Repository public interface
ReactiveQuoteRepository
   extends ReactiveCrudRepository
                  <Quote, Integer> {
   Flux<Quote> findAllByPlay
      (String play);

   Flux<Quote> findAllByIdIn
      (List<Integer> queryIds);
}
```

Asynchronously find all Quote rows in the database that contain the List of requested subscription types (i.e., ZIPPY and/or HANDEY)

See springframework/data/repository/reactive/ReactiveCrudRepository.html

Structure & Functionality of the ReactiveQuoteRepository

- This class implements a persistent repository containing information about Shakespeare Quote objects using the R2DBC database

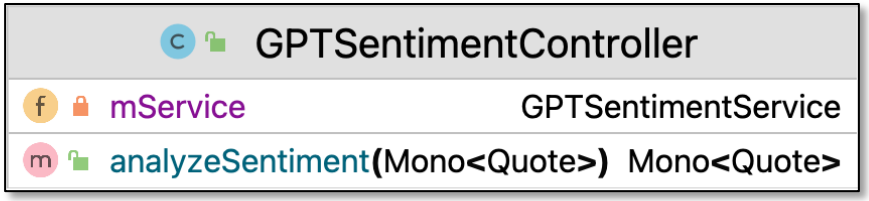
```
@Repository public interface
ReactiveQuoteRepository
   extends ReactiveCrudRepository
                  <Quote, Integer> {
   Flux<Quote> findAllByPlay
      (String play);

   Flux<Quote> findAllByIdIn
      (List<Integer> queryIds);
}
```

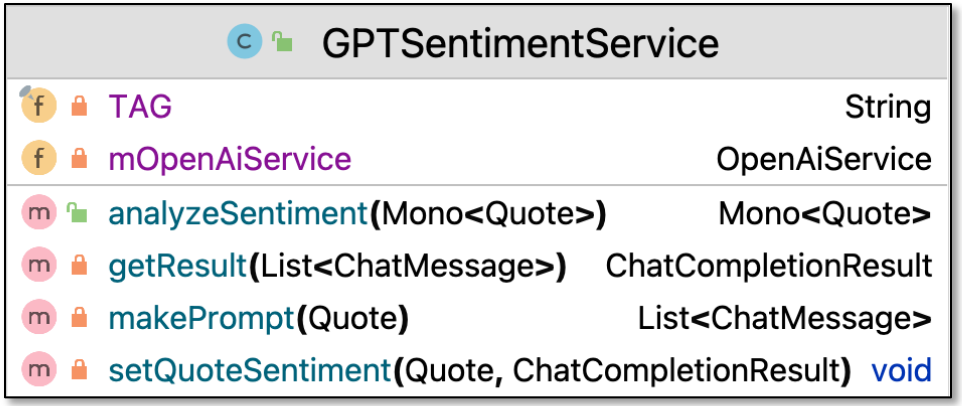
This Spring Data API method implementation is automatically generated as SELECT * FROM quote WHERE id IN (:queryIds)

Structure & Functionality of the GPTSentiment* Classes

Structure & Functionality of the GPTSentiment* Classes

- The GPTSentimentController & GPTSentimentService handle requester messages that trigger ChatGPT sentiment analysis





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

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;



  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM) {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

See RSocket/ex3/src/main/java/quotes/responder/chatgptsentiment/GPTSentimentController.java

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;


  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM)  {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

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

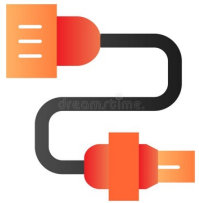
See www.baeldung.com/spring-controllers

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;


  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM)  {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

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



See www.baeldung.com/spring-autowire

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;


  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM)  {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

This endpoint handler is called to perform sentiment analysis of a Shakespeare quote

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;

  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM)  {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

This annotation defines an endpoint that handles an RSocket binary message with the given path

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

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentController defines an endpoint that sents requests to ChatGPT

```
@Controller
public class GPTSentimentController {
  @Autowired
  private GPTSentimentService mService;

  @MessageMapping(CHAT_GPT_SENTIMENT_ANALYSIS)
  public Mono<Quote> analyzeSentiment(Mono<Quote> quoteM) {
    return mService
      .analyzeSentiment(quoteM);
  }
}
```

Forwards to the service

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   @Autowired
   private OpenAiService mOpenAiService;
   ...
```

See RSocket/ex3/src/main/java/quotes/responder/chatgptsentiment/GPTSentimentService.java

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   @Autowired
   private OpenAiService mOpenAiService;
   ...
```

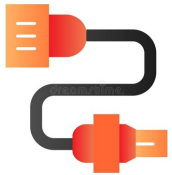
This annotation indicates the class implements "business logic" & enables auto-detection & wiring of dependent classes via classpath scanning

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

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   @Autowired
   private OpenAiService mOpenAiService;
   ...
```



Auto-wire the means to access Chat GPT via Spring dependency injection

See github.com/TheoKanning/openai-java

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   @Autowired
   private OpenAiService mOpenAiService;
   ...
```

This factory method Bean creates an instance of OpenAiService

```
public class ResponderBeans {
   @Bean
   OpenAiService getOpenAiService() {
      return new OpenAiService(BuildConfig.API_KEY);
   }
}
```

See RSocket/ex3/src/main/java/quotes/common/ResponderBeans.java

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
  @Autowired
  private OpenAiService mOpenAiService;
  ...
```

The API key is configured in the private.properties file

```
public class ResponderBeans {
  @Bean
  OpenAiService getOpenAiService() {
    return new OpenAiService(BuildConfig.API_KEY);
  }
}
```

See github.com/douglascraigschmidt/LiveLessons/blob/master/RSocket/ex3/build-config.gradle

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   ...

   public Mono<Quote> analyzeSentiment
      (Mono<Quote> quoteM) {
      return quoteM.map(quote -> {
             List<ChatMessage> messages = makePrompt(quote);
             var ccResult = getResult(messages);
             setQuoteSentiment(quote, ccResult);
             return quote;
         });
   ...
```

Uses ChatGPT to perform sentiment analysis of a quote from Shakespeare

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   ...

   public Mono<Quote> analyzeSentiment
      (Mono<Quote> quoteM) {
      return quoteM.map(quote -> {
            List<ChatMessage> messages = makePrompt(quote);
            var ccResult = getResult(messages);
            setQuoteSentiment(quote, ccResult);
            return quote;
         });
   ...
```

Create a prompt, pass it to ChatGPT, & update the quote

See next part of the lesson for details of these methods

Structure & Functionality of the GPTSentiment* Classes

- GPTSentimentService uses ChatGPT to analyze the sentiment of Bard quotes

```
@Service
public class GPTSentimentService {
   ...

   public Mono<Quote> analyzeSentiment
      (Mono<Quote> quoteM) {
      return quoteM.map(quote -> {
             List<ChatMessage> messages = makePrompt(quote);
             var ccResult = getResult(messages);
             setQuoteSentiment(quote, ccResult);
             return quote;
         });
   ...
```

Return a Mono that emits the updated Bard Quote

End of Structure & Functionality of the RSocket Shakespeare Quotes Responder (Part 2)