The QuoteServices App Case Study: Zippy Microservice Structure & Functionality (Part 4)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

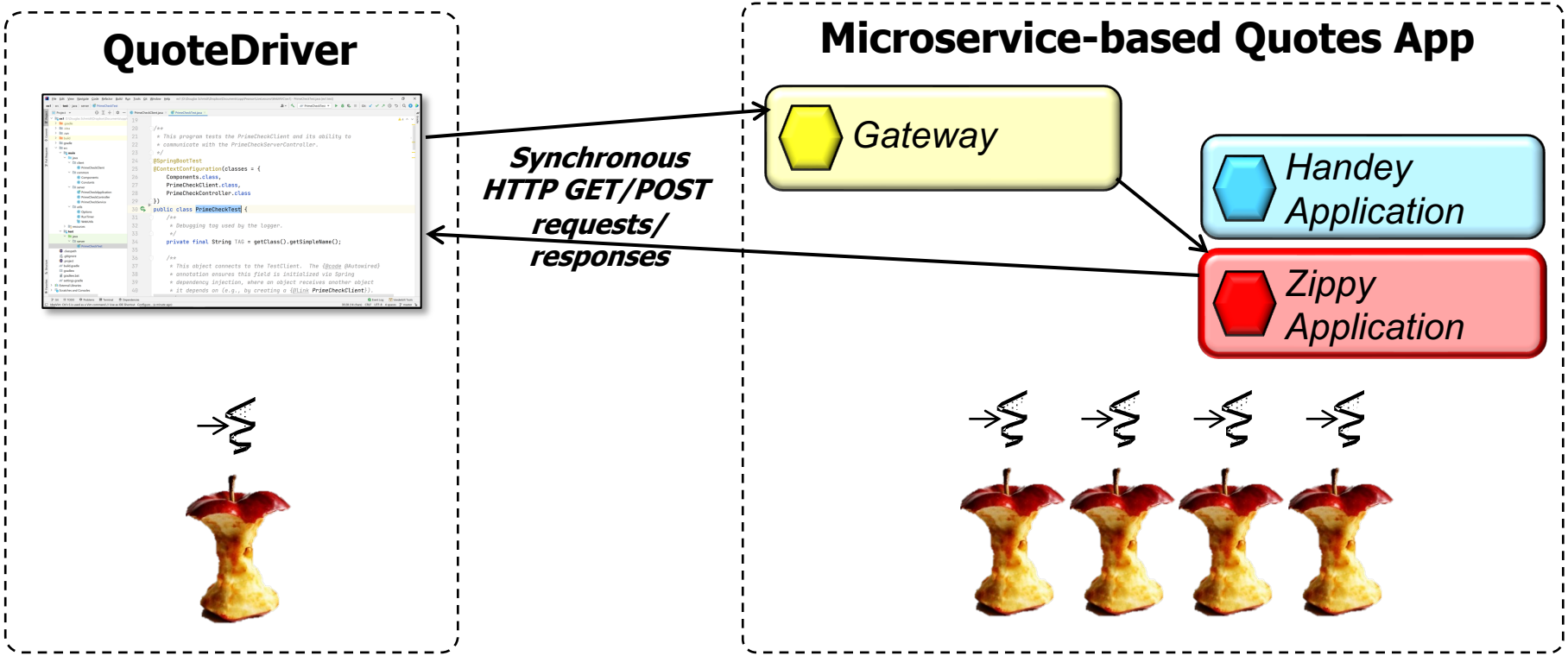
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the the ZippyService class & how its applies the Jakarta Persistence API (JPA)



This microservice uses Java parallel & sequential streams a bit..

Structure & Functionality of the ZippyService

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

The BaseService defines the methods overridden by both HandeyService & ZippyService

See microservices/src/main/java/edu/vandy/quoteservices/common/BaseService.java

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

Defines the classic Java type returned by endpoint handler methods in the ZippyService

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

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

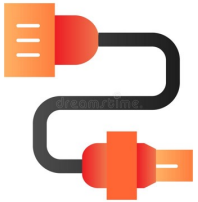
See www.baeldung.com/spring-component-repository-service

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

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



See www.baeldung.com/spring-autowire

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  @Autowired private JPAQuoteRepository mRepository;
  ...
```

This repository stores Zippy quotes persistently

```
(default, 'All of life is a blur of Republicans and meat!'),
(default, '..Are we having FUN yet...?'),
(default, 'Life is a POPULARITY CONTEST! I''m REFRESHINGLY CANDID!!'),
(default, 'You were s''posed to laugh!'),
(default, 'Fold, fold, FOLD!!  FOLDING many items!!'),
```

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  @Cacheable(ZIPPY_CACHE)
  public List<Quote> getAllQuotes() {
    return mRepository
      .findAll();
  } ...
```

This method uses the JPA to return all the Zippy quotes

The generated SQL query would be "SELECT * FROM quote"

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  @Cacheable(ZIPPY_CACHE)
  public List<Quote> getAllQuotes() {
    return mRepository
      .findAll();
  } ...
```

It also uses an annotation to cache the List of Quote objects in memory on the server-side

See www.baeldung.com/spring-cache-tutorial

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  @Cacheable(value = ZIPPY_CACHE, key = "#quoteId")
  public Quote getQuote(Integer quoteId) {
    return mRepository
      .findById(quoteId)
        .orElseThrow(IllegalArgumentException::new);
  } ...
```

Get the Quote associated with the given quoteId

The generated SQL query would be "SELECT * FROM quotes WHERE id = ?"

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  @Cacheable(value = ZIPPY_CACHE, key = "#quoteId")
  public Quote getQuote(Integer quoteId) {
    return mRepository
      .findById(quoteId)
        .orElseThrow(IllegalArgumentException::new);
  } ...
```

This annotation caches the Quote associated with the quoteId in memory on the server-side

See www.baeldung.com/spring-cache-tutorial

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  public List<Quote> postQuotes(List<Integer> quoteIds,
                                Boolean notUsed) {
    return mRepository
      .findAllById(quoteIds);
  } ...
```

This method uses the JPA to find all quotes matching the quoteIds

The generated SQL query would be "SELECT * FROM quote WHERE id IN (?, ?, ...)"

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  public List<Quote> search(List<String> queries,
                            Boolean parallel) {
    return StreamSupport
      .stream(queries.spliterator(), parallel)
      .flatMap(query -> mRepository
               .findByQuoteContainingIgnoreCase(query)
               .stream())
      .distinct()
      .toList();
  } ...
```

Uses Java parallel & sequential streams & the JPA to find Quote objects matching any query

The SQL query is "SELECT * FROM quote WHERE LOWER(quote) LIKE LOWER('%{query}%')"

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by ZippyController

```
@Service
public class ZippyService implements BaseService<List<Quote>> {
  ...
  public List<Quote> searchEx(List<String> queries,
                              Boolean notUsed) {
    return mRepository
      .findAllByQuoteContainingAllIn(queries);
  } ...
```

This method uses a custom SQL query to find all Quote objects containing all the queries

See part 2 of this lessons on "*Zippy Microservice Structure & Functionality*"

End of the QuoteServices App Case Study: Zippy MicroService Structure & Functionality (Part 4)