The QuoteServices App Case Study: Zippy Microservice Structure & Functionality

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

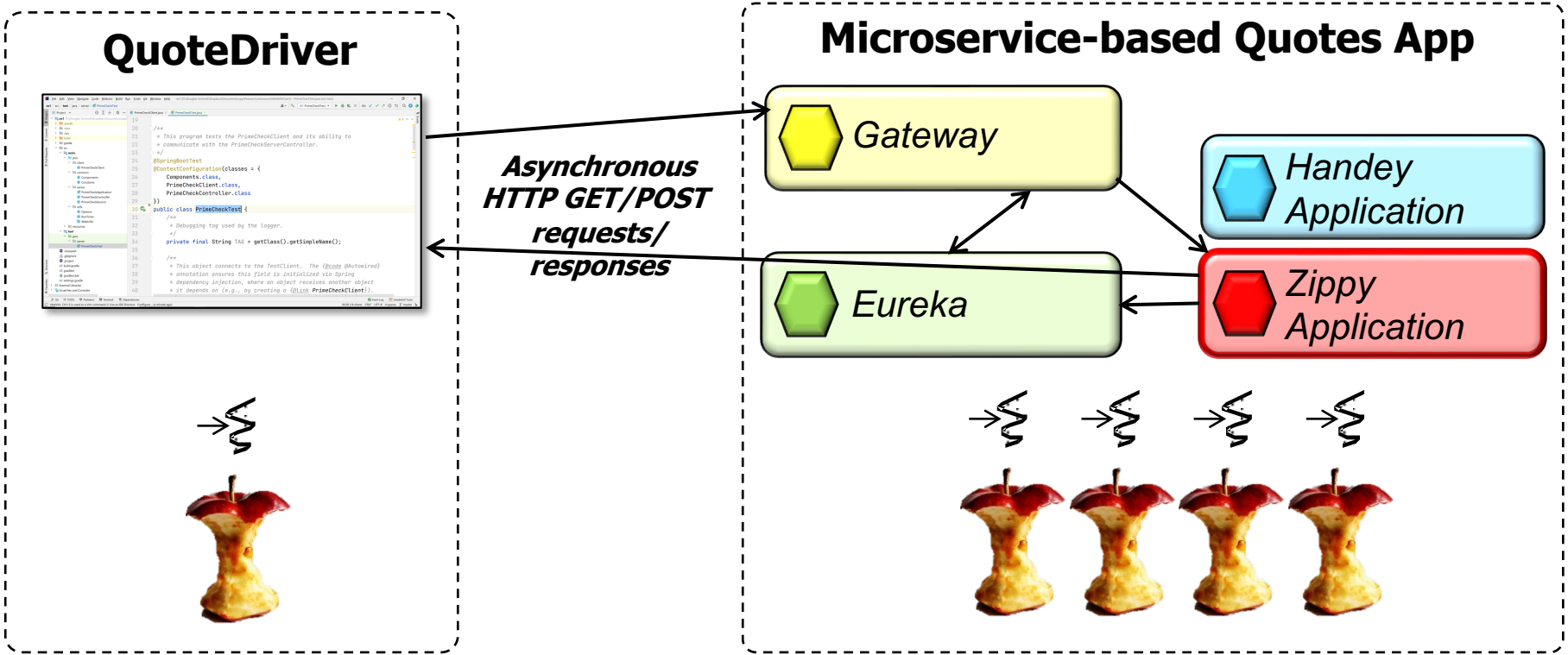
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the ZippyController/ZippyService microservice & how it encapsulates the Jakarta Persistence API (JPA)



This microservice also uses Project Reactor reactive types (Flux)

Structure & Functionality of the ZippyApplication

Structure & Functionality of the ZippyApplication

- Provides the entry point into the Spring WebMVC-based version of the Zippy Quote microservice

```
@SpringBootApplication
@EntityScan(basePackageClasses =
                {Quote.class})
@EnableJpaRepositories(basePackageClasses =
                  {JPAQuoteRepository.class})
public class ZippyApplication
       extends BaseApplication {



  public static void main(String[] args) {
    run(ZippyApplication.class, args);
  }
}
```

See zippymicroservice/src/main/java/edu/vandy/quoteservices/microservice/ZippyApplication.java

Structure & Functionality of the ZippyApplication

- Provides the entry point into the Spring WebMVC-based version of the Zippy Quote microservice

```
@SpringBootApplication
@EntityScan(basePackageClasses =
                {Quote.class})
@EnableJpaRepositories(basePackageClasses =
                  {JPAQuoteRepository.class})
public class ZippyApplication
       extends BaseApplication {

  public static void main(String[] args) {
    run(ZippyApplication.class, args);
  }
}
```

BaseApplication defines the run() method used by both the HandeyApplication & ZippyApplication

See WebFlux/ex3/common/src/main/java/edu/vandy/quoteservices/common

Structure & Functionality of the ZippyApplication

- Provides the entry point into the Spring WebMVC-based version of the Zippy Quote microservice

```
@SpringBootApplication
@EntityScan(basePackageClasses =
                {Quote.class})
@EnableJpaRepositories(basePackageClasses =
                  {JPAQuoteRepository.class})
public class ZippyApplication
       extends BaseApplication {


  public static void main(String[] args) {
    run(ZippyApplication.class, args);
  }
}
```

These annotations enable this microservice to use R2DBC

See springframework/data/jpa/repository/config/EnableJpaRepositories.html

Structure & Functionality of the ZippyApplication

- Provides the entry point into the Spring WebMVC-based version of the Zippy Quote microservice

```
@SpringBootApplication
@EntityScan(basePackageClasses =
                {Quote.class})
@EnableJpaRepositories(basePackageClasses =
                  {JPAQuoteRepository.class})
public class ZippyApplication
       extends BaseApplication {
```

The main entry-point method calls the BaseApplication helper method to build & run the Zippy microservice & register with the Eureka discovery service

```
   public static void main(String[] args) {
      run(ZippyApplication.class, args);
   }
}
```

Structure & Functionality of the ZippyController

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   @Autowired
   ApplicationContext applicationContext;

   @Autowired
   ZippyService mService;

   ...
```

See zippymicroservice/src/main/java/edu/vandy/quoteservices/microservice/ZippyController.java

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   @Autowired
   ApplicationContext applicationContext;

   @Autowired
   ZippyService mService;

   ...
```

This annotation ensures that request handling methods in the controller class all automatically serialize return objects into HttpResponse objects

See www.baeldung.com/spring-controller-vs-restcontroller

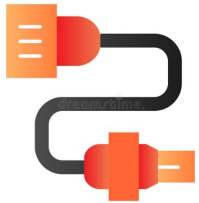
Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   @Autowired
   ApplicationContext applicationContext;

   @Autowired
   ZippyService mService;

   ...
```



These fields are auto-wired by Spring's dependency injection framework

See www.baeldung.com/spring-autowire

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   ...
   @GetMapping("/")
   public ResponseEntity<String> info() {

      return ResponseEntity
         .ok(applicationContext.getId()
             + " is alive and running at "
             + Thread.currentThread()
             + "\n");
   }
   ...
```

A GET request used to test Eureka connectivity

See spring.io/projects/spring-cloud-netflix

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   ...
   @GetMapping(GET_ALL_QUOTES)
   public Flux<Quote> getAllQuotes()
   { return mService.getAllQuotes(); }

   @PostMapping(POST_QUOTES)
   Flux<Quote> postQuotes(@RequestBody List<Integer> quoteIds)
   { return mService.postQuotes(quoteIds); }

   @PostMapping(POST_SEARCHES)
   public Flux<Quote> search(@RequestBody List<String> queries)
   { return mService.search(queries); }
   ...
```

These methods handle GET & POST HTTP requests by forwarding to the service

See quoteservices/microservice/ZippyController.java

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   ...
   @GetMapping(GET_ALL_QUOTES)
   public Flux<Quote> getAllQuotes()
   { return mService.getAllQuotes(); }

   @PostMapping(POST_QUOTES)
   Flux<Quote> postQuotes(@RequestBody List<Integer> quoteIds)
   { return mService.postQuotes(quoteIds); }

   @PostMapping(POST_SEARCHES)
   public Flux<Quote> search(@RequestBody List<String> queries)
   { return mService.search(queries); }
   ...
```

These methods return Flux objects based on service responses

Structure & Functionality of the ZippyController

- ZippyController maps HTTP GET & POST requests to endpoint handlers

```
@RestController
public class ZippyController {
   ...
   @GetMapping(GET_ALL_QUOTES)
   public Flux<Quote> getAllQuotes()
   { return mService.getAllQuotes(); }

   @PostMapping(POST_QUOTES)
   Flux<Quote> postQuotes(@RequestBody List<Integer> quoteIds)
   { return mService.postQuotes(quoteIds); }

   @PostMapping(POST_SEARCHES)
   public Flux<Quote> search(@RequestBody List<String> queries)
   { return mService.search(queries); }
   ...
```

These annotations ensure the HTTP GET & POST requests are properly transformed into Java method calls

Structure & Functionality of the ZippyService

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
   @Autowired
   private JPAQuoteRepository mRepository;

   ...
```

See quoteservices/microservices/zippy/ZippyService.java

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
   @Autowired
   private JPAQuoteRepository mRepository;

   ...
```

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

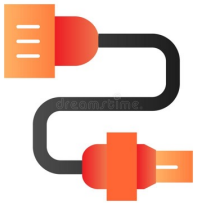
See www.baeldung.com/spring-component-repository-service

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
   @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 HandeyController

```
@Service
public class ZippyService {
  public Flux<Quote> postQuotes(List<Integer> quoteIds) {
    return Flux.fromIterable(mRepository.findAllById(quoteIds));
  }

  public Flux<Quote> search(List<String> queries) {
    return Flux.fromIterable(queries).parallel()...;
  }

  public Flux<Quote> searchEx(List<String> queries) {
    return Flux.fromIterable(mRepository
                .findAllByQuoteContainingAllIn(queries));
  } ...
```

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
  public Flux<Quote> postQuotes(List<Integer> quoteIds) {
    return Flux.fromIterable(mRepository.findAllById(quoteIds));
  }

  public Flux<Quote> search(List<String> queries) {
    return Flux.fromIterable(queries).parallel()...;
  }

  public Flux<Quote> searchEx(List<String> queries) {
    return Flux.fromIterable(mRepository
                .findAllByQuoteContainingAllIn(queries));
  } ...
```

All methods return the Flux reactive type

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
  public Flux<Quote> postQuotes(List<Integer> quoteIds) {
    return Flux.fromIterable(mRepository.findAllById(quoteIds));
  }

  public Flux<Quote> search(List<String> queries) {
    return Flux.fromIterable(queries).parallel()...;
  }

  public Flux<Quote> searchEx(List<String> queries) {
    return Flux.fromIterable(mRepository
                .findAllByQuoteContainingAllIn(queries));
  } ...
```

All methods simply forward to the JPAQuoteRepository

Structure & Functionality of the ZippyService

- ZippyService defines implementation methods called by HandeyController

```
@Service
public class ZippyService {
  public Flux<Quote> postQuotes(List<Integer> quoteIds) {
    return Flux.fromIterable(mRepository.findAllById(quoteIds));
  }

  public Flux<Quote> search(List<String> queries) {
    return Flux.fromIterable(queries).parallel()...;
  }

  public Flux<Quote> searchEx(List<String> queries) {
    return Flux.fromIterable(mRepository
                .findAllByQuoteContainingAllIn(queries));
  } ...
```

Each List object is encapsulated by a Flux

Structure & Functionality of the MultiQueryRepository*

Structure & Functionality of the MultiQueryRepository*

- The MultiQueryRepository defines an interface for creating a custom query

```
public interface MultiQueryRepository {
   ...
```

This interface defines the means to perform multiple queries at once on the Quote database

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

Structure & Functionality of the MultiQueryRepository*

- The MultiQueryRepository defines an interface for creating a custom query

```
public interface MultiQueryRepository {
   ...

  Flux<Quote> findAllByQuoteContainingIgnoreCaseAllIn
                  (List<String> queries);
}
```

Find a List of Quote objects in the database containing all of the queries (ignoring case)

This method can't be generated automatically by the Spring Data API

Structure & Functionality of the MultiQueryRepository*

- The MultiQueryRepository defines an interface for creating a custom query

```
public class MultiQueryRepositoryImpl
        implements MultiQueryRepository {
    ...

}
```

Applies the "Repository Implementation" pattern

See docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.custom-implementations

Structure & Functionality of the MultiQueryRepository*

- The MultiQueryRepository defines an interface for creating a custom query

```
public class MultiQueryRepositoryImpl
        implements MultiQueryRepository {
    @PersistenceContext
    private EntityManager entityManager;
    ...

}
```

This field defines a database session that provides the main API for performing CRUD operations & querying the database

See docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html

Structure & Functionality of the MultiQueryRepository*

- The MultiQueryRepository defines an interface for creating a custom query

```
public class MultiQueryRepositoryImpl
       implements MultiQueryRepository {
   ...
 Flux<Quote> findAllByQuoteContainingIgnoreCaseAllIn
                  (List<String> queries) {
     var criteriaBuilder = entityManager.getCriteriaBuilder();
     var criteriaQuery = criteriaBuilder
       .createQuery(Quote.class);
     var quote = criteriaQuery.from(Quote.class);
     var idExpression = criteriaBuilder
        .lower(quote.get("quote"));
     var andPredicate = ...
     return getQueryResults(criteriaQuery, andPredicate);
  ...
```

Find Quote objects containing all queries (ignoring case)

See upcoming lesson on "*Implementing the Zippy Microservice*"

End of the QuoteServices App Case Study: Zippy MicroService Structure & Functionality