The QuoteServices App Case Study: Zippy Microservice Structure & Functionality (Part 2)

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

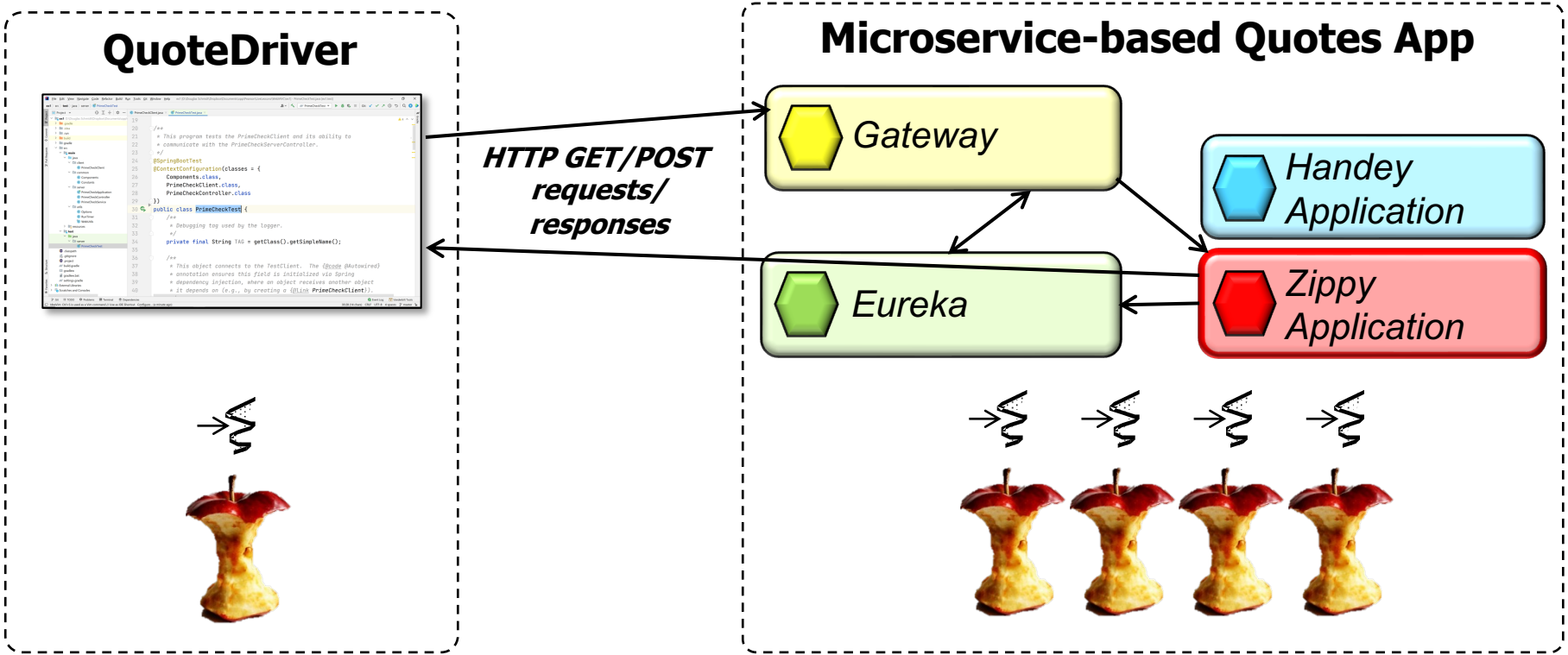
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of the JPAQuoteRepository & how it creates a custom SQL query with the Jakarta Persistence API (JPA)



We also give an overview of the Spring Data API

Overview of the Spring Data API

Overview of the Spring Data API

- The Spring JPA Data API is a framework that simplifies development of data access layers in Spring-based apps

See www.baeldung.com/the-persistence-layer-with-spring-data-jpa

Overview of the Spring Data API

- The Spring JPA Data API is a framework that simplifies development of data access layers in Spring-based apps
  - Developers write repository interfaces that define data access methods they need

Keyword	Sample	JPQL
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection age)	… where x.age not in ?1
TRUE	findByActiveTrue()	… where x.active = true
FALSE	findByActiveFalse()	… where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	… where UPPER(x.firstame) = UPPER(?1)

See www.baeldung.com/spring-data-derived-queries

Overview of the Spring Data API

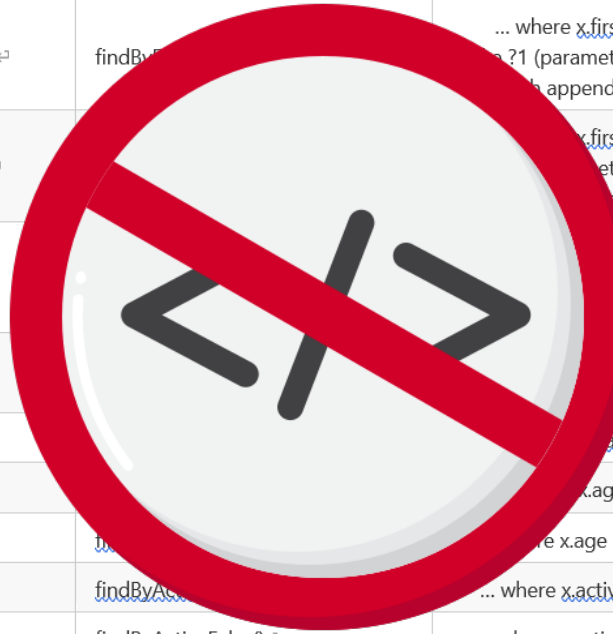
- The Spring JPA Data API is a framework that simplifies development of data access layers in Spring-based apps
  - Developers write repository interfaces that define data access methods they need
  - Spring automatically generates the SQL queries to implement those data access methods

Keyword	Sample	JPQL
NotLike	findByFirstnameNotLike	... where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	... where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	... where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	... where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	... where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	... where x.lastname <> ?1
In	findByAgeIn(Collection ages)	... where x.age in ?1
NotIn	findByAgeNotIn(Collection age)	... where x.age not in ?1
TRUE	findByActiveTrue()	... where x.active = true
FALSE	findByActiveFalse()	... where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	... where UPPER(x.firstame) = UPPER(?1)

Overview of the Spring Data API

- The Spring JPA Data API is a framework that simplifies development of data access layers in Spring-based apps
  - Developers write repository interfaces that define data access methods they need
  - Spring automatically generates the SQL queries to implement those data access methods
    - It reduces boilerplate code needed to implement persistence & data access layers in a Java application

Keyword	Sample	JPQL
NotLike	findByFirstnameNotLike	... where x.firstname not like ?1
StartingWith	findBy	... where x.firstname ?1 (parameter bound appended %)
EndingWith		x.firstname eter bound ded %)
Containing		name bound %)
OrderBy		order by esc
Not		ame <> ?1
In		.age in ?1
NotIn		e x.age not in ?1
TRUE	findByAc	... where x.active = true
FALSE	findByActiveFalse()	... where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	... where UPPER(x.firstame) = UPPER(?1)



See en.wikipedia.org/wiki/Low-code_development_platform

Motivating the Need for a Custom SQL Query

Motivating the Need for a Custom SQL Query

- There are limitations to combos of Spring Data API keywords that are supported via the JPA

Keyword	Sample	JPQL
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection age)	… where x.age not in ?1
TRUE	findByActiveTrue()	… where x.active = true
FALSE	findByActiveFalse()	… where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	… where UPPER(x.firstame) = UPPER(?1)

See www.baeldung.com/spring-data-derived-queries

Motivating the Need for a Custom SQL Query

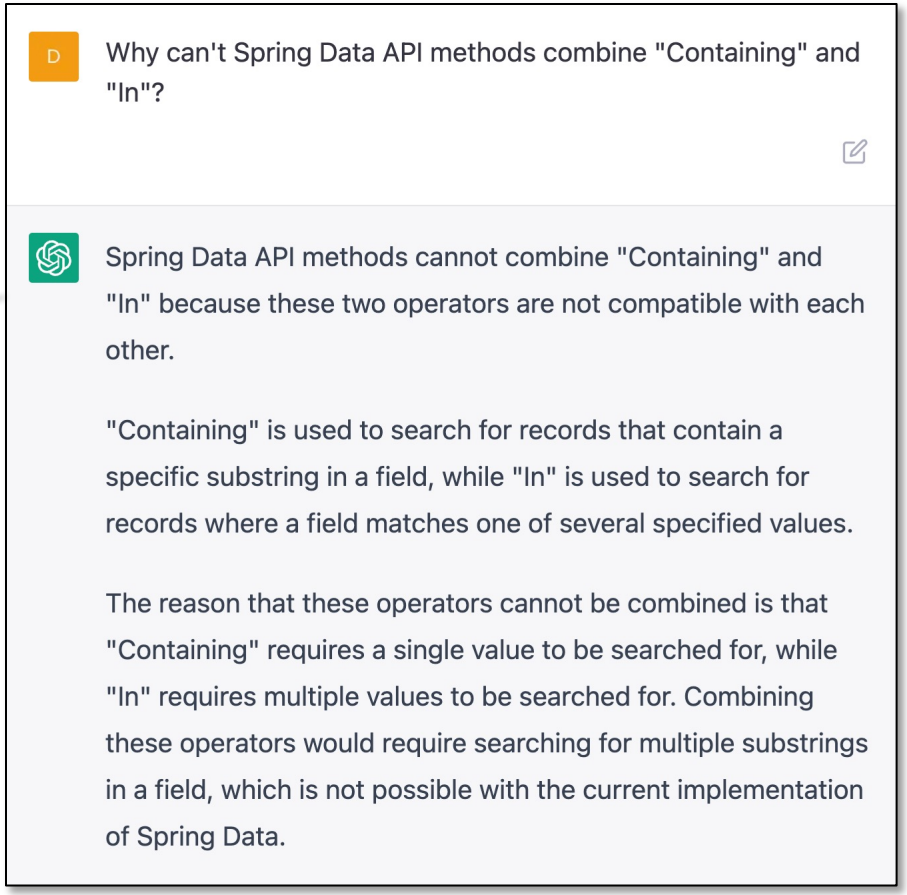
- There are limitations to combos of Spring Data API keywords that are supported via the JPA
  - e.g., combining "Containing" & "In" is not possible

```
ex4 – JPAQuoteRepository.java [ex4.microservices.main]
ex4 › microservices › src › main › java › edu › vandy › quoteservices › m
QuoteDriver.java | JPAQuoteRepository.java

 */
@Repository
public interface JPAQuoteRepository
        extends JpaRepository<Quote, Integer> {
    List<Quote> findByQuoteContainingInIgnoreCase(List<String> queries);

    Method 'findByQuoteContainingInIgnoreCase(java.util.List<java.lang.String>)' is never used
    Cannot resolve property 'containing'
    Safe delete 'findByQuoteContainingInIgnoreCase(List<String>)'  More actions...

    /**
     * Find all {@link Quote}
     * {@code query} {@link String} (ignoring case).
     *
     * @param query The {@link String} to search for
     * @return A {@link List} of {@link Quote} objects that match the
     *         {@code query}
     */
    List<Quote> findByQuoteContainingIgnoreCase(String query);

}

Git | TODO | Database Changes | Problems | Terminal | Endpoints | Services | Profiler | Build | Logcat | Dependencies | App Inspection
* daemon started successfully (9 minutes ago)
19:47  CRLF  UTF-8  4 spaces  master
```

Motivating the Need for a Custom SQL Query

- There are limitations to combos of Spring Data API keywords that are supported via the JPA
  - e.g., combining "Containing" & "In" is not possible

Here's ChatGPT's explanation of why these methods can't be combined



Good luck finding even this explanation with a Google search..

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 {
   ...

  List<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 {
   ...
 List<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 (Part 2)