The QuoteServices App Case Study: Zippy Microservice Structure & Functionality (Part 3)

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

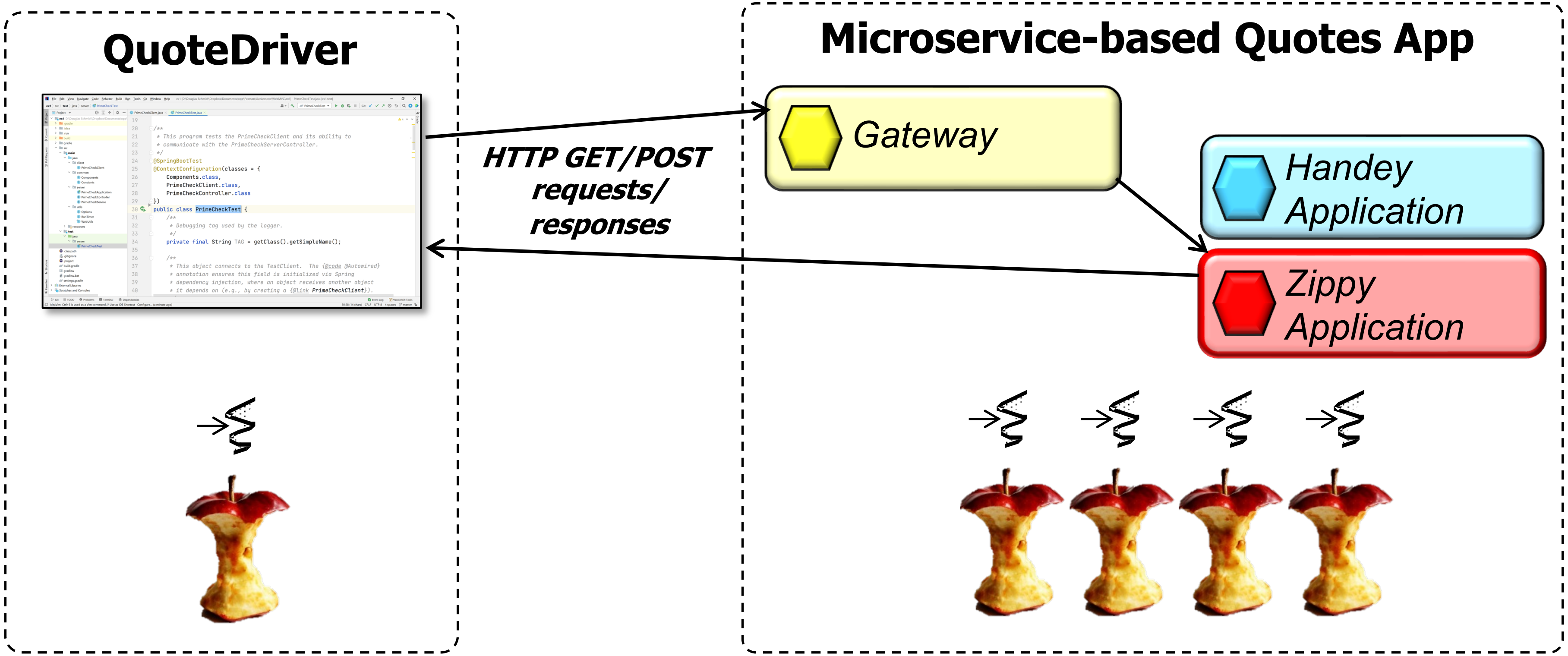
Professor of Computer Science

Institute for Software Integrated Systems

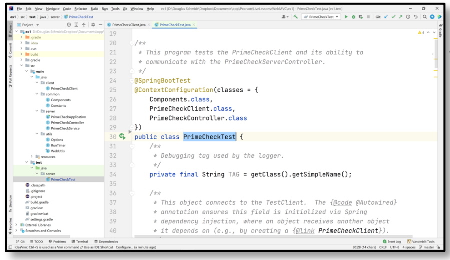
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how (&why) to create a custom SQL query with the Jakarta Persistence API (JPA)



QuoteDriver
HTTP GET/POST requests/ responses
Microservice-based Quotes App
Gateway
Handey Application
Zippy Application



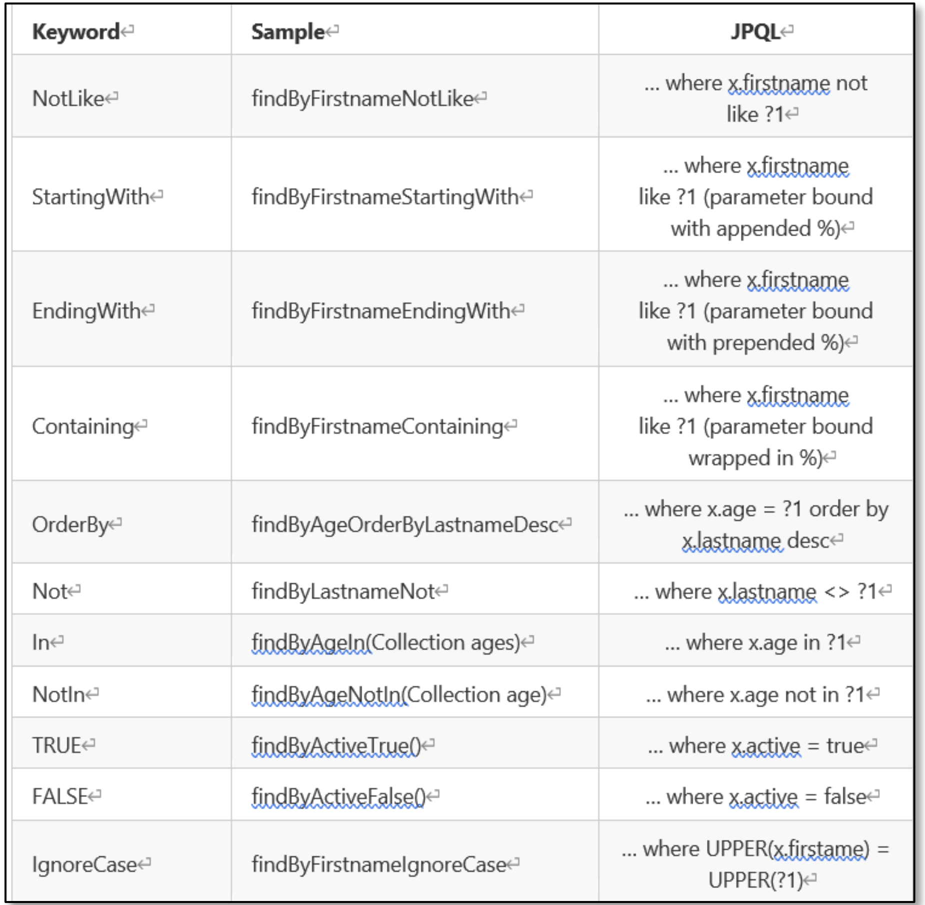
```java
/**
 * This program tests the PrimeCheckClient and its ability to
 * communicate with the PrimeCheckServerController.
 */
@SpringBootTest
@ContextConfiguration(classes = {
    Components.class,
    PrimeCheckClient.class,
    PrimeCheckController.class
})
public class PrimeCheckTest {
    /**
     * Debugging tag used by the logger.
     */
    private final String TAG = getClass().getSimpleName();

    /**
     * This object connects to the TestClient.  The {@code @Autowired}
     * annotation ensures this field is initialized via Spring
     * dependency injection, where an object receives another object
     * it depends on (e.g., by creating a {@link PrimeCheckClient}).
```

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

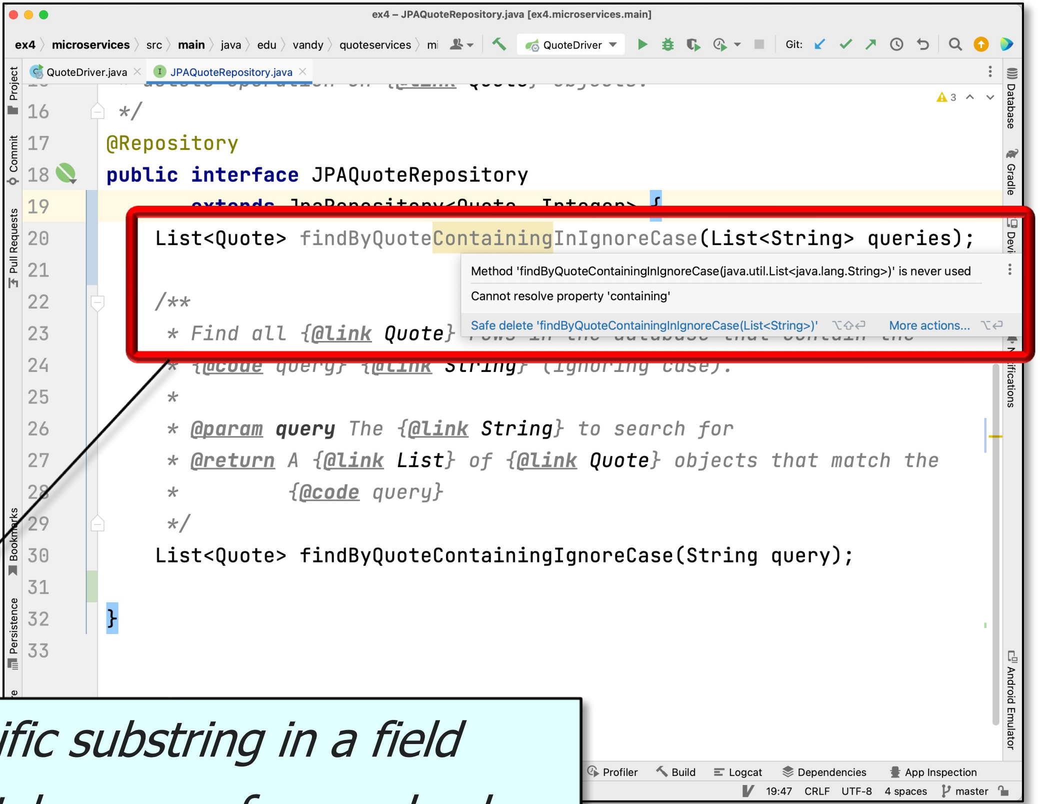


Keyword	Sample	JPQL
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… 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]

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

    /**
     * Find all {@link Quote} rows in the database that contain the
     * {@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);

}
```

Method 'findByQuoteContainingInIgnoreCase(java.util.List<java.lang.String>)' is never used

Cannot resolve property 'containing'

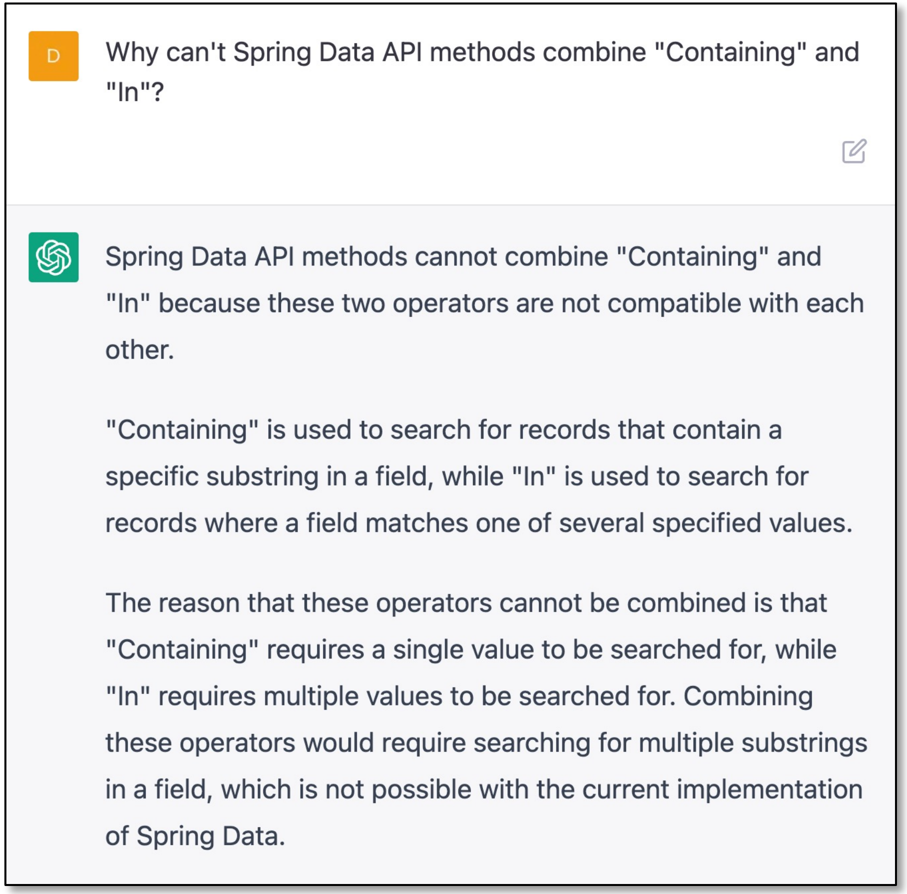
Safe delete 'findByQuoteContainingInIgnoreCase(List<String>)'   More actions...

- *"Containing" searches for a specific substring in a field*
- *"In" searches for a field that matches one of several values*

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



D

Why can't Spring Data API methods combine "Containing" and "In"?

Spring Data API methods cannot combine "Containing" and "In" because these two operators are not compatible with each other.

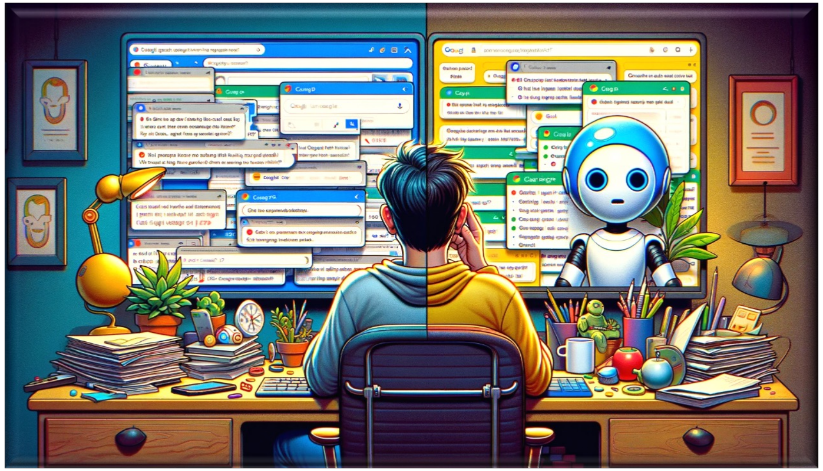
"Containing" is used to search for records that contain a specific substring in a field, while "In" is used to search for records where a field matches one of several specified values.

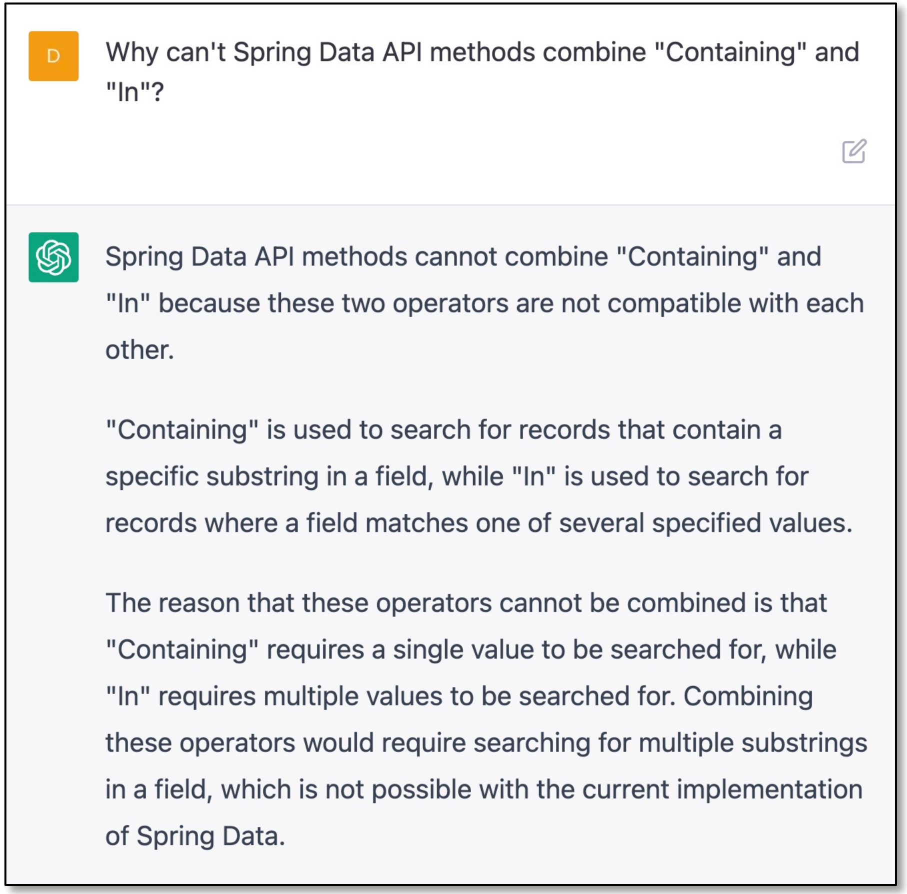
The reason that these operators cannot be combined is that "Containing" requires a single value to be searched for, while "In" requires multiple values to be searched for. Combining these operators would require searching for multiple substrings in a field, which is not possible with the current implementation of Spring Data.

See chat.openai.com/share/fab9ca17-6891-44da-a003-a15a2e4274ca

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





D

Why can't Spring Data API methods combine "Containing" and "In"?

Spring Data API methods cannot combine "Containing" and "In" because these two operators are not compatible with each other.

"Containing" is used to search for records that contain a specific substring in a field, while "In" is used to search for records where a field matches one of several specified values.

The reason that these operators cannot be combined is that "Containing" requires a single value to be searched for, while "In" requires multiple values to be searched for. Combining these operators would require searching for multiple substrings in a field, which is not possible with the current implementation of Spring Data.

Good luck finding this explanation with a conventional 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 MultiQueryRepositoryImpl class implements the 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 MultiQueryRepositoryImpl class implements the custom query

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

}
```

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 MultiQueryRepositoryImpl class implements the custom query

```
public class MultiQueryRepositoryImpl
       implements MultiQueryRepository {
   ...
  List<Quote> findAllByQuoteContainingIgnoreCaseAllIn
                 (List<String> queries) {
      var criteriaBuilder = mEntityManager.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 3)