Creating Database Queries From Method Names

By Ramesh Fadatare (Java Guides)

Query generation from method names

Spring Data JPA query methods are the most powerful methods, we can create query methods to select records from the database without writing SQL queries. Behind the scenes, Spring Data JPA will create SQL queries based on the query method and execute the query for us.

findByName(String name)

```
select id, name, description, active, image_url, price, sku from products where name="product name"
```

We can create query methods for repository using Entity fields

Creating query methods is also called finder methods (findBy, findAll …)

Query creation/generation from method names works

We write Query Method using Spring Data JPA

```
public interface UserRepository extends Repository<User, Long> {

  List<User> findByEmailAddressAndLastname(String emailAddress, String lastname);
}
```

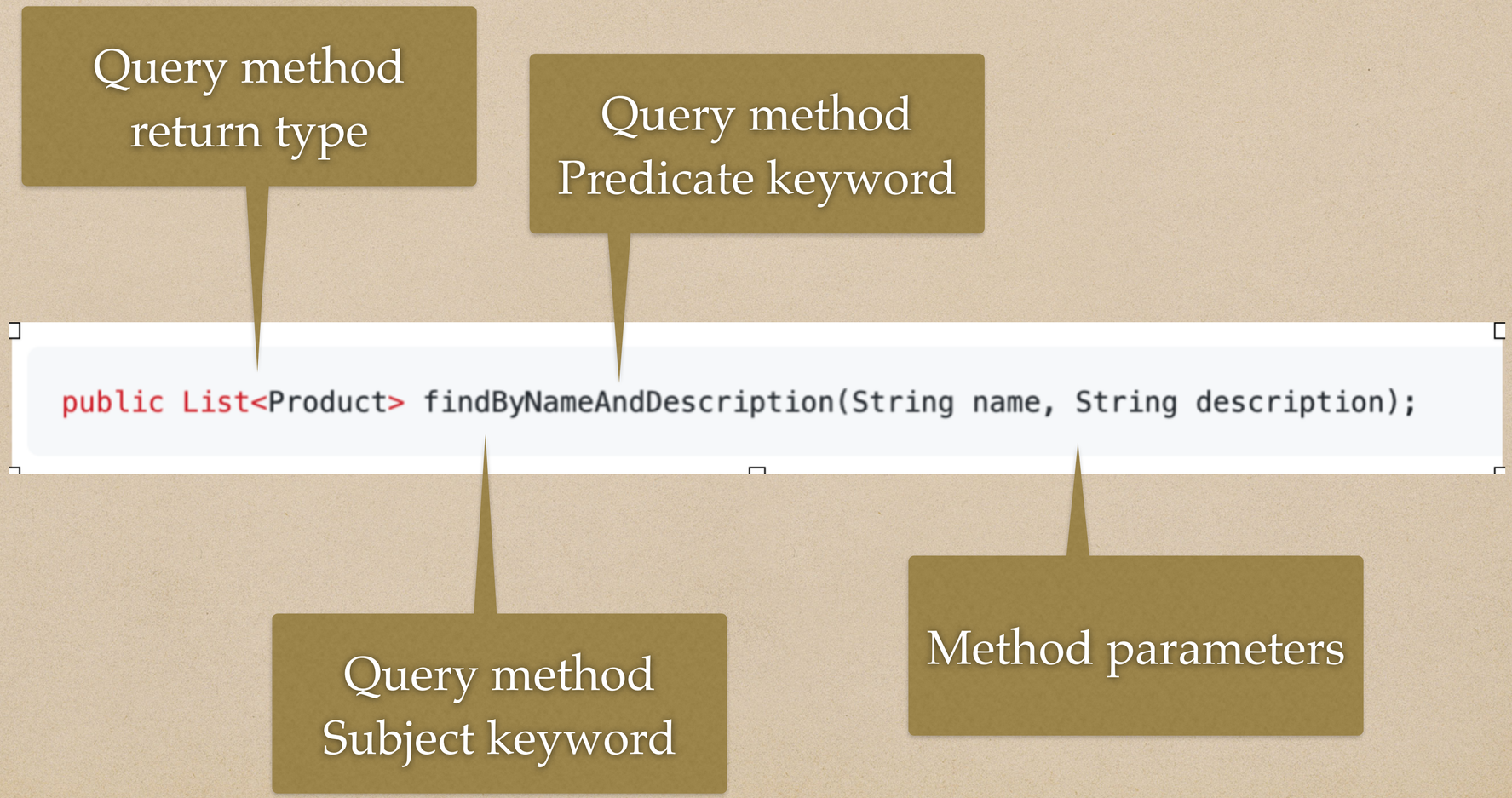
Spring Data JPA parse Query method and creates JPA Criteria

Parsing query method names is divided into subject and predicate. The first part (find…By, exists…By) defines the subject of the query, the second part forms the predicate.

JPA criteria creates JPQL query and executes it

```
select u from User u where u.emailAddress = ?1 and u.lastname = ?2
```

Query Method Structure



Rules to create query methods

1. The name of our query method must start with one of the following prefixes: find…By, read…By, query…By, count…By, and get…By.

Examples: findByName, readByName, queryByName, getByName

2. If we want to limit the number of returned query results, we can add the First or the Top keyword before the first By word.

Examples: findFirstByName, readFirst2ByName, findTop10ByName

3. If we want to select unique results, we have to add the Distinct keyword before the first By word.

Examples: findDistinctByName or findNameDistinctBy

4. Combine property expression with AND and OR.

Examples: findByNameOrDescription, findByNameAndDescription

Returning Values From Query Methods

A query method can return only one result or more than one result.

1. if we are writing a query that should return only one result, we can return the following types:

- ***Basic type*. Our query method will return the found basic type or *null*.**
- ***Entity*. Our query method will return an entity object or *null*.**
- **Guava / Java 8 *Optional<T>*. Our query method will return an *Optional* that contains the found object or an empty *Optional*.**

```
public Product findByName(String name);
public Optional<Product> findById(long id);
```

2. if we are writing a query method that should return more than one result, we can return the following types:

- ***List<T>*. Our query method will return a list that contains the query results or an empty list.**
- ***Stream<T>*. Our query method will return a *Stream* that can be used to access the query results or an empty *Stream*.**

```
List<Product> findByPriceGreaterThan(BigDecimal price);
Stream<Product> findByPriceLessThan(BigDecimal price);
```

Query or finder methods examples

```java
@Entity
@Table(name="products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
            generator = "product_generator")
    @SequenceGenerator(name = "product_generator",
            sequenceName = "product_sequence_name",
            allocationSize = 1)

    @Column(name = "id")
    private Long id;

    @Column(name = "sku")
    private String sku;

    @Column(name = "name")
    private String name;

    @Column(name = "description")
    private String description;

    @Column(name = "price")
    private BigDecimal price;
```

```java
public interface ProductRepository extends JpaRepository<Product, Long> {
    public Optional<Product> findById(long id);
    public List<Product> findByNameOrDescription(String name, String description);
    public Product findDistinctByName(String title);
    List<Product> findByPriceGreaterThan(BigDecimal price);
    List<Product> findByPriceLessThan(BigDecimal price);
    List<Product> findByNameContaining(String name);
    List<Product> findByDateCreatedBetween(LocalDateTime startDate, LocalDateTime endDate);
    List<Product> findByPriceBetween(BigDecimal startPrice, BigDecimal endPrice);
    List<Product> findByNameLike(String name);
    List<Product> findByNameIn(List<String> names);
    List<Product> findFirst2ByName(String name);
}
```

Query method - Find by single field name

Write a query method to find or retrieve a product by name

```
select * from products where name = "product 1";
```

```
public Product findByName(String name);
```

Write a query method to find or retrieve a product by id

```
select * from products where id = 1;
```

```
public Optional<Product> findById(long id);
```

Query method - Find by multiple field names

Write a query method to find or retrieve a product by name or description

```
select * from products where name = "product 1" or description = "product 1 desc";
```

```
public List<Product> findByNameOrDescription(String name, String description);
```

Write a query method to find or retrieve a product by name and description

```
select * from products where name = "product 1" and description = "product 1 desc";
```

```
public List<Product> findByNameAndDescription(String name, String description);
```

Query method - Find by Distinct

Write a query method to find or retrieve a unique product by name

```
select distinct id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
   name="product 1";
```

```
public Product findDistinctByName(String title);
```

Query method - Find by GreaterThan

Write a query method to find or retrieve products whose price is greater than given price as method parameter

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
  price > 100;
```

```
List<Product> findByPriceGreaterThan(BigDecimal price);
```

Query method - Find by LessThan

Write a query method to find or retrieve products whose price is less than given price as method parameter

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
  price < 200;
```

```
List<Product> findByPriceLessThan(BigDecimal price);
```

Query method - Find by Containing

Write a query method to find or retrieve filtered products that match the given text (contains check)

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
  name like '%product%';
```

```
List<Product> findByNameContaining(String name);
```

Query method - Find by Like

Write a query method to find or retrieve products for a specified pattern in a column (SQL LIKE condition)

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
  name like '%product%';
```

```
List<Product> findByNameLike(String name);
```

Query method - Between

Write a query method to find or retrieve products based on the price range (start price and end price)

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
   price between 100 and 300
```

```
List<Product> findByPriceBetween(BigDecimal startPrice, BigDecimal endPrice);
```

Query method - Between

Write a query method to find or retrieve products based on the start date and end date

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
   date_created between '2022-01-28 22:54:15' and '2022-01-28 22:59:23'
```

```
List<Product> findByDateCreatedBetween(LocalDateTime startDate, LocalDateTime endDate);
```

Query method - In

Write a query method to find or retrieve products based on multiple values (specify multiple values in a SQL where clause)

```
select id, active, date_created, description,image_url,last_updated,name, price,sku
from
   products
where
   name in ('product 1', 'product 2')
```

```
List<Product> findByNameIn(List<String> names);
```

Query method - Limiting Query Results

Spring Data JPA supports keywords 'first' or 'top' to limit the query results.
Example: findFirstByName(), findTop5BySku()

An optional numeric value can be appended after 'top' or 'first' to limit the maximum number of results to be returned (e.g. findTop3By....). If this number is not used then only one entity is returned.

There's no difference between the keywords 'first' and 'top'.

When Should We Use Query Methods

This query generation strategy has the following benefits:

- **Creating simple queries is fast.**
- **The method name of our query method describes the selected value(s) and the used search condition(s).**

This query generation strategy has the following weaknesses:

- **The features of the method name parser determine what kind of queries we can create. If the method name parser doesn't support the required keyword, we cannot use this strategy.**
- **The method names of complex query methods are long and ugly.**
- **There is no support for dynamic queries.**