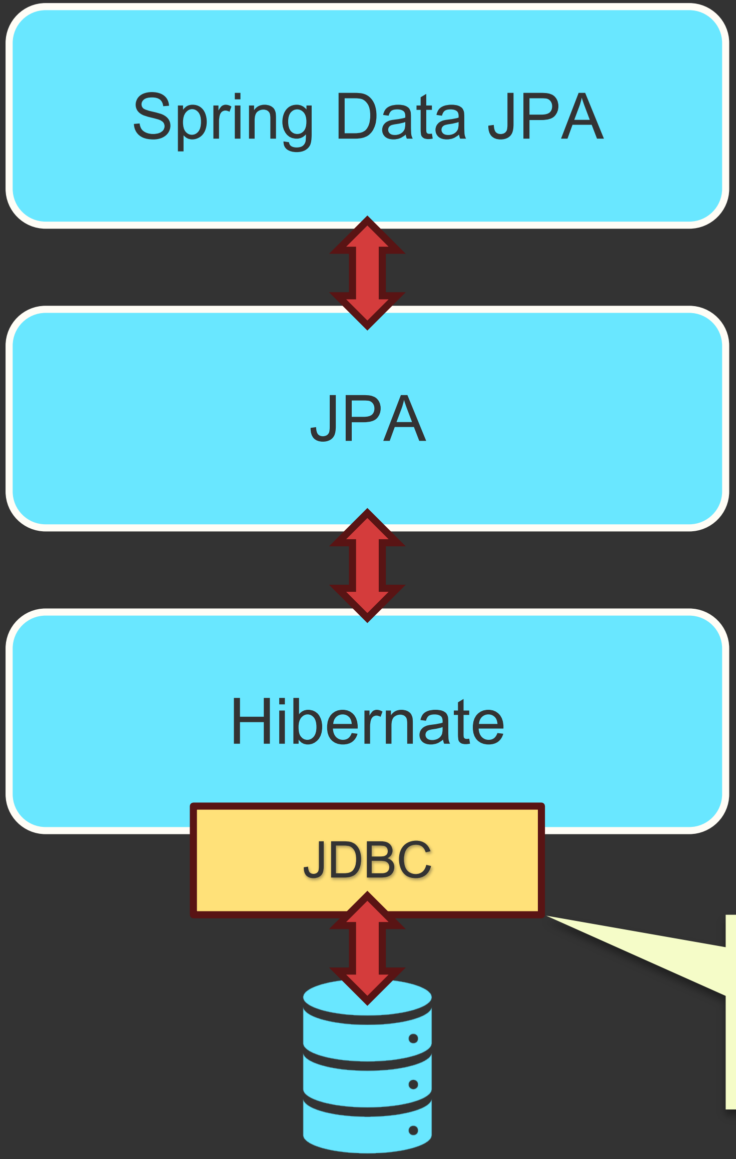
# 3.3

# Spring Data JPA & Dynamic Query Methods



Spring Data JPA

JPA

Hibernate

JDBC

Built on top of JPA and helps building DAO methods using Repository

Just a specification as to how the Relation and Objects mapping shout take place

Hibernate is the implementation built on JPA specification

Queries generated by hibernate get executed using JDBC

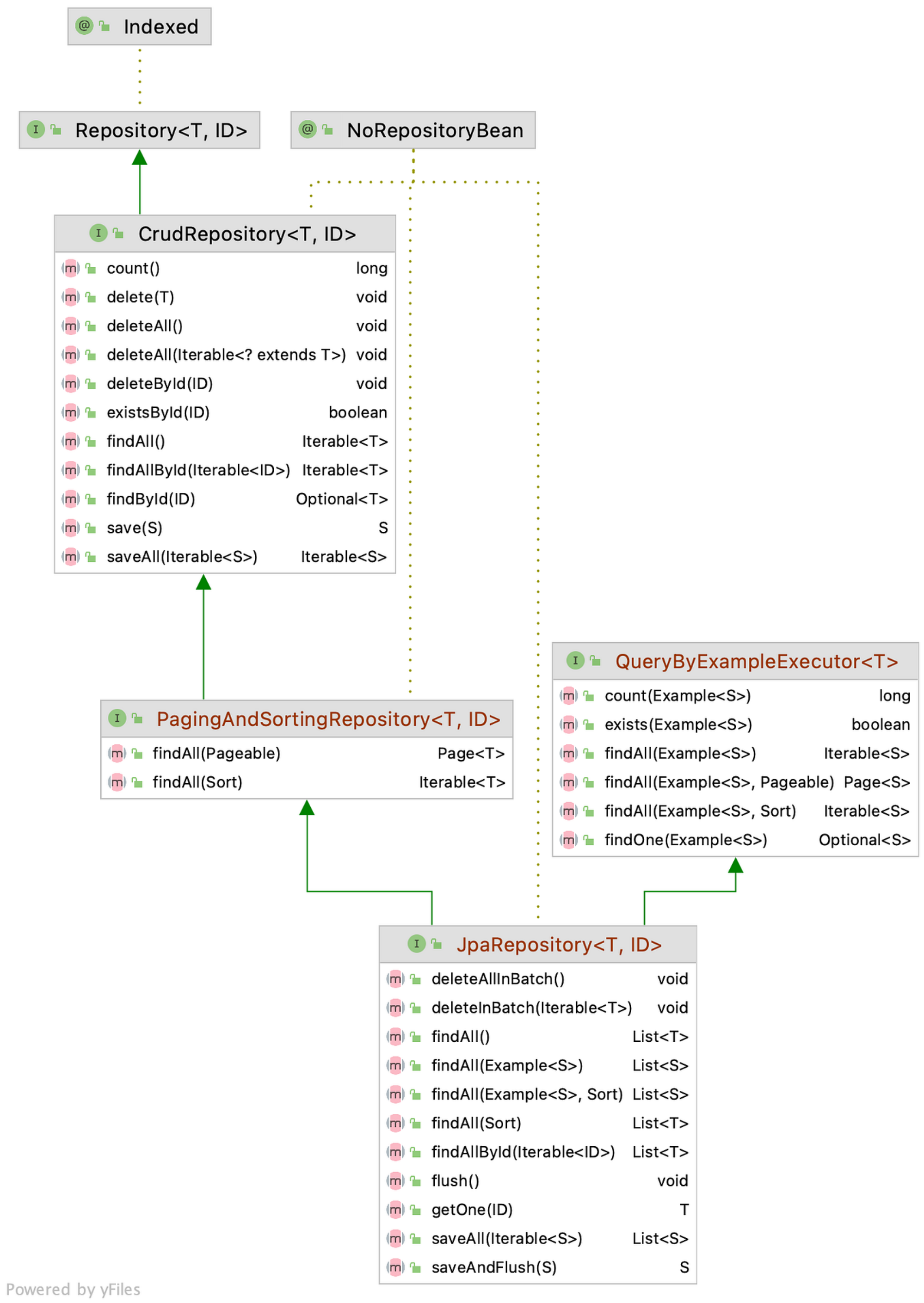
# Spring Data JPA

Spring Data JPA is a part of the larger Spring Data family.

It builds on top of JPA, providing a higher-level and more convenient abstraction for data access.

Spring Data JPA makes it easier to implement JPA-based repositories by providing boilerplate code, custom query methods, and various utilities to reduce the amount of code you need to write.

# Spring Data JPA Repositories



Indexed

Repository<T, ID>

NoRepositoryBean

CrudRepository<T, ID>

| | |
|---|---|
| count() | long |
| delete(T) | void |
| deleteAll() | void |
| deleteAll(Iterable<? extends T>) | void |
| deleteById(ID) | void |
| existsById(ID) | boolean |
| findAll() | Iterable<T> |
| findAllById(Iterable<ID>) | Iterable<T> |
| findById(ID) | Optional<T> |
| save(S) | S |
| saveAll(Iterable<S>) | Iterable<S> |

PagingAndSortingRepository<T, ID>

| | |
|---|---|
| findAll(Pageable) | Page<T> |
| findAll(Sort) | Iterable<T> |

QueryByExampleExecutor<T>

| | |
|---|---|
| count(Example<S>) | long |
| exists(Example<S>) | boolean |
| findAll(Example<S>) | Iterable<S> |
| findAll(Example<S>, Pageable) | Page<S> |
| findAll(Example<S>, Sort) | Iterable<S> |
| findOne(Example<S>) | Optional<S> |

JpaRepository<T, ID>

| | |
|---|---|
| deleteAllInBatch() | void |
| deleteInBatch(Iterable<T>) | void |
| findAll() | List<T> |
| findAll(Example<S>) | List<S> |
| findAll(Example<S>, Sort) | List<S> |
| findAll(Sort) | List<T> |
| findAllById(Iterable<ID>) | List<T> |
| flush() | void |
| getOne(ID) | T |
| saveAll(Iterable<S>) | List<S> |
| saveAndFlush(S) | S |

# Key features of Spring Data JPA

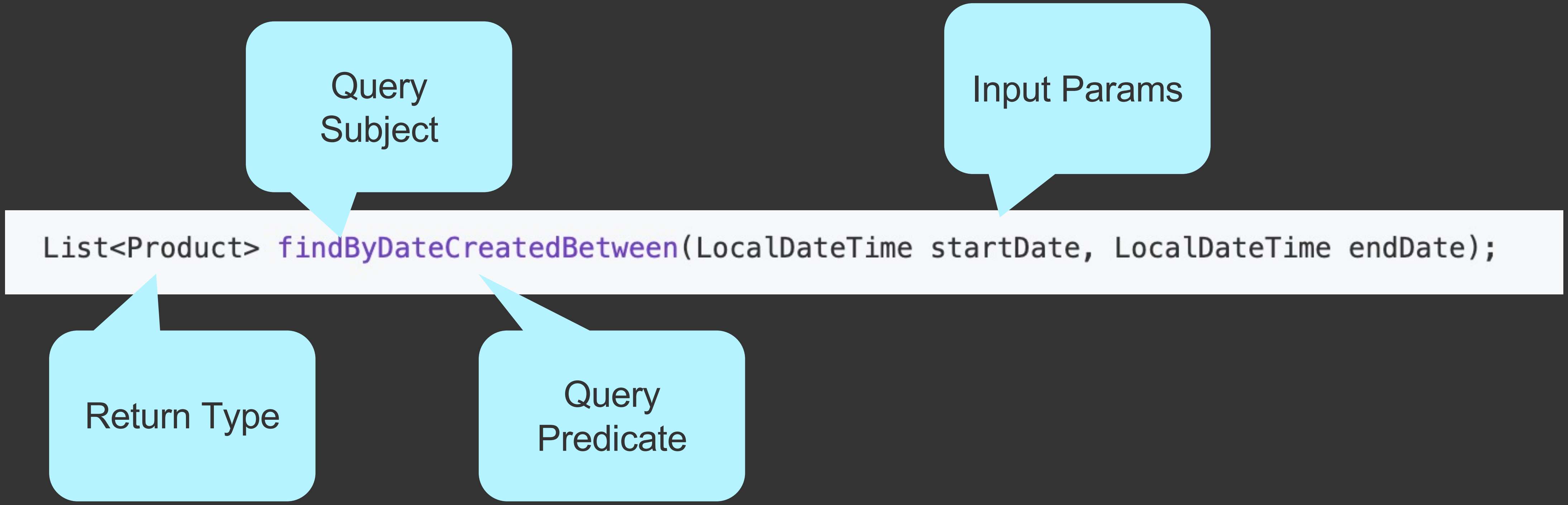
1. Repository Abstraction: Provides a Repository interface with methods for common data access operations.

2. Custom Query Methods: Allows defining custom query methods by simply declaring method names.

3. Pagination and Sorting: Offers built-in support for pagination and sorting.

4. Query Derivation: Automatically generates queries from method names.

# Rules for Creating Query Methods



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

Query Subject

Input Params

Return Type

Query Predicate

# Rules for Method Names

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

# Rules for Method Names

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*

5. Checkout the Official Doc for more info: [https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html](https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html)