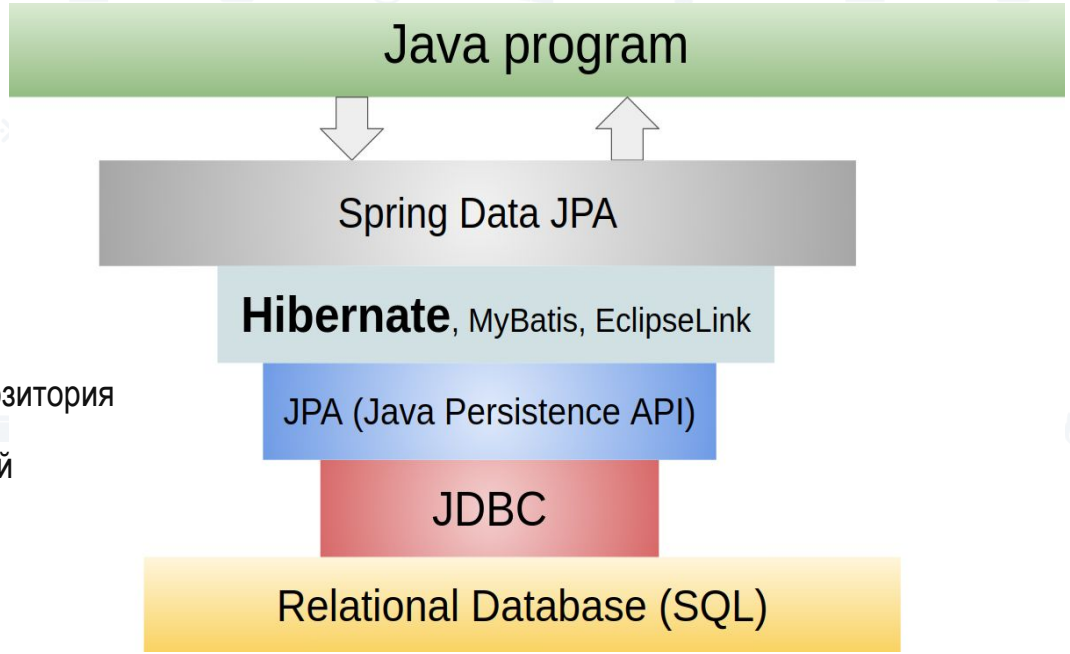
Lesson 35

27.02.2023

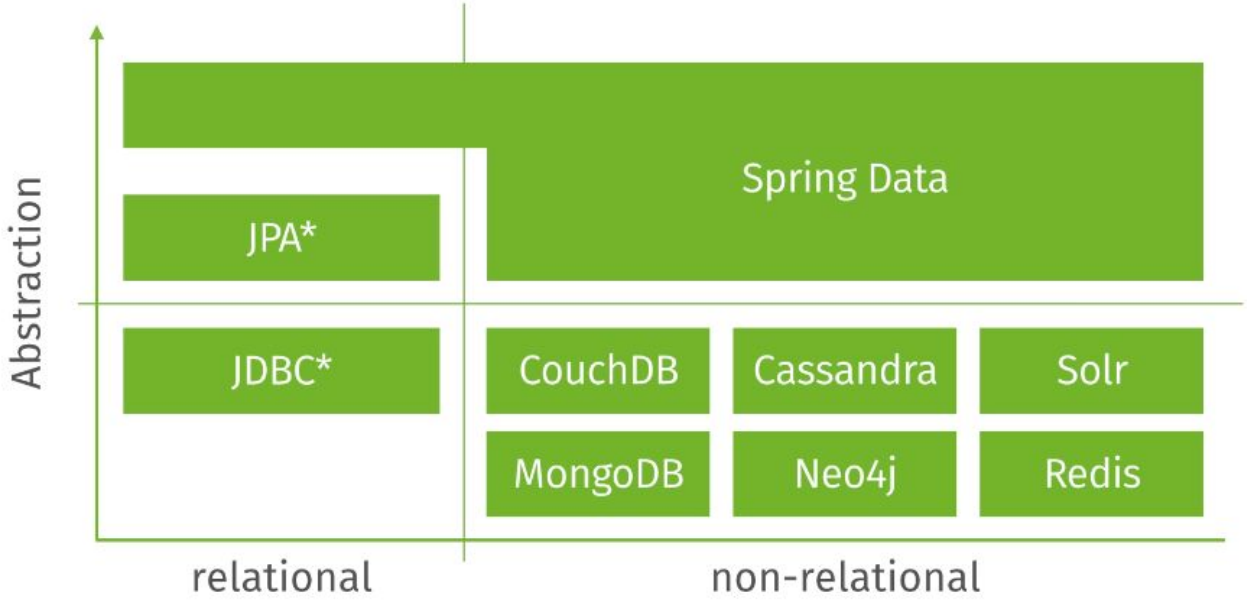
Spring Data — дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение, в каких то случаях для этого будет достаточно объявить интерфейс и метод в нем, без имплементации.

Содержит:

1. Spring Repository
2. Методы запросов из имени метода
3. Конфигурация и настройка
4. Специальная обработка параметров
5. Пользовательские реализации для репозитория
6. Пользовательский Базовый Репозиторий
7. Методы запросов — Query



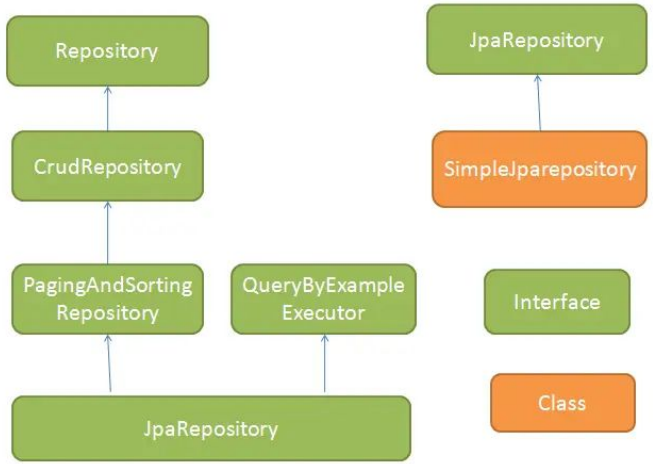
Spring Data


Abstraction
Spring Data
JPA*
JDBC*
CouchDB
Cassandra
Solr
MongoDB
Neo4j
Redis
relational
non-relational

Репозитории Spring Data JPA - это интерфейсы, которые вы можете определить для доступа к данным. Запросы JPA создаются автоматически из имен ваших методов. Например, интерфейс CityRepository может объявить метод findAllByState(String state), чтобы найти все города (city) в данном штате (state).

Для более сложных запросов вы можете аннотировать ваш метод аннотацией Query в Spring Data.

Хранилища Spring Data обычно берутся из интерфейсов Repository или CrudRepository. Если вы используете автоконфигурацию, поиск в репозиториях происходит из пакета, содержащего ваш основной класс конфигурации (тот, который аннотирован @EnableAutoConfiguration или @SpringBootApplication).



Методы запросов из имени метода

Запросы к сущности можно строить прямо из имени метода. Для этого используется механизм префиксов **find…By, read…By, query…By, count…By, и get…By**, далее от префикса метода начинает разбор остальной части. Вводное предложение может содержать дополнительные выражения, например, Distinct. Далее первый By действует как разделитель, чтобы указать начало фактических критериев. Можно определить условия для свойств сущностей и объединить их с помощью And и Or.

Keyword	Sample	JPQL snippet
And	`findByLastnameAndFirstname`	`… where x.lastname = ?1 and x.firstname = ?2`
Or	`findByLastnameOrFirstname`	`… where x.lastname = ?1 or x.firstname = ?2`
Is,Equals	`findByFirstname`,`findByFirstnameIs`,`findByFirstnameEquals`	`… where x.firstname = ?1`
Between	`findByStartDateBetween`	`… where x.startDate between ?1 and ?2`
LessThan	`findByAgeLessThan`	`… where x.age < ?1`
LessThanEqual	`findByAgeLessThanEqual`	`… where x.age <= ?1`
GreaterThan	`findByAgeGreaterThan`	`… where x.age > ?1`
GreaterThanEqual	`findByAgeGreaterThanEqual`	`… where x.age >= ?1`
After	`findByStartDateAfter`	`… where x.startDate > ?1`
Before	`findByStartDateBefore`	`… where x.startDate < ?1`

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<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> ages)	… 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)

Специальная обработка параметров

В методах запросов, в их параметрах можно использовать специальные параметры Pageable, Sort, а также ограничения Top и First.

Например вот так можно взять вторую страницу (индекс с -0), размером в три элемента и сортировкой по firstName, предварительно указав в методе репозитория параметр Pageable, также будут использованы критерии из имени метода — «Искать по FirstName начиная с % „

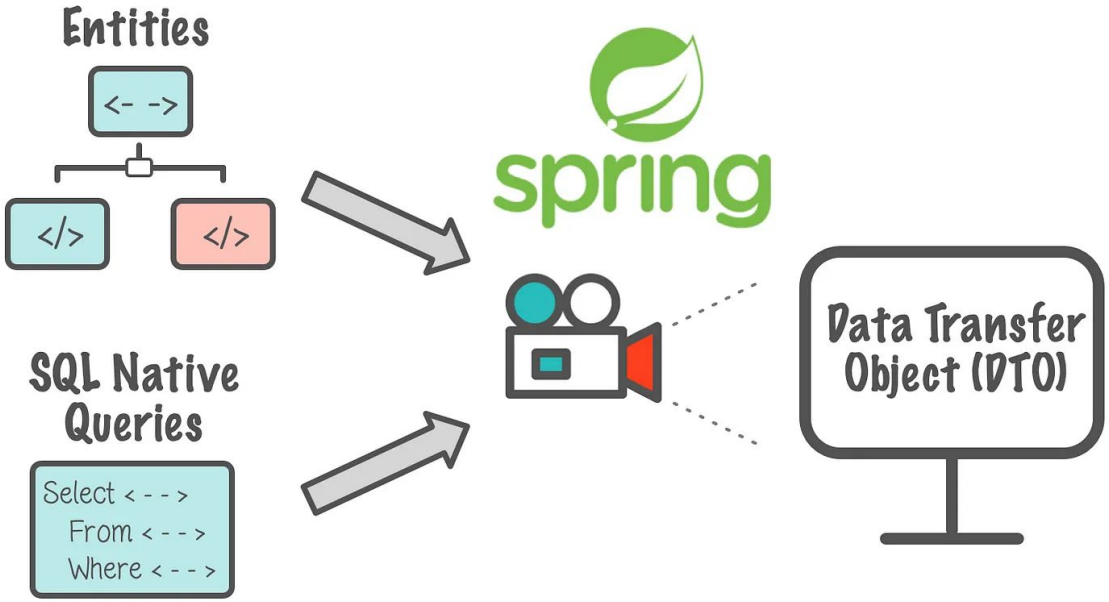
```
@Repository
public interface CustomizedEmployeesCrudRepository extends CrudRepository<Employees, Long> {
    List<Employees> findByFirstNameStartsWith(String firstNameStartsWith, Pageable page);
}

public void testFindByFirstNameStartsWithOrderByFirstNamePage() {
    List<Employees> list = employeesCrudRepository
    .findByFirstNameStartsWith("A", PageRequest.of(1,3, Sort.by("firstName")));
    list.forEach(e -> System.out.println(e.getFirstName() + " " +e.getLastName()));
}
```

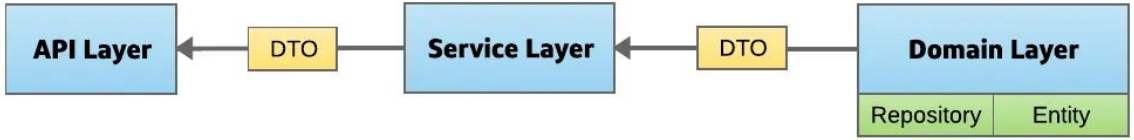
Методы запросов — Query

Ранее я писал, что если нужен специфичный метод или его реализация, которую нельзя описать через имя метода, то это можно сделать через некоторый Customized интерфейс (CustomizedEmployees) и сделать реализацию вычисления. А можно пойти другим путем, через указание запроса (HQL или SQL), как вычислить данную функцию. Для моего примера с getEmployeesMaxSalary, этот вариант реализации даже проще. Я еще усложню его входным параметром salary. Т.е. достаточно объявить в интерфейсе метод и запрос вычисления.

```
1  @Repository
2  public interface CustomizedEmployeesCrudRepository extends CrudRepository<Employees, Long>,
       CustomizedEmployees<Employees> {
3
4      @Query("select e from Employees e where e.salary > :salary")
5      List<Employees> findEmployeesWithMoreThanSalary(@Param("salary") Long salary, Sort sort);
6  }
```

Spring Data JPA Projection





Spring data architecture