



Spring Boot



# **Spring Data JPA**

@Query(JPQL)

Nguyễn Nghiệm



















- QUERY(JPQL)
  - QUERY(SQL, NATIVEQUERY)
  - SORT & PAGINATE INTEGRATION



2





- ☐ Các phương thức truy vấn sẵn có trong JpaRepository<T, ID>
  - findAll(): List<T>
  - findAll(Sort): List<T>
  - findAll(Pageable): Page<T>
  - getById(ID): T
  - findById(ID): Optional<T>
  - count(): long
  - existsById(ID): boolean
- Truy vấn thy biến theo yêu cầu?
  - **♦** findByXyz([params]): List<T>
  - ❖ getByXyz([params]): T





### **CUSTOM QUERY WITH JPQL & SQL**

```
public interface ProductDAO extends JpaRepository<Product, Integer>{
    @Query(value="SELECT * FROM Products p" +
        " INNER JOIN Categories c ON c.Id=p.CategoryId" +
        " WHERE c.id = ?1", nativeQuery=true)
    List<Product> findByCategoryId(Integer id);
}
```

```
List<Product> list = dao.findByCategoryId(1005)
```

```
public interface ProductDAO extends JpaRepository<Product, Integer>{
    @Query("SELECT p FROM Product p WHERE p.category.id = ?1")
    List<Product> findByCategoryId(Integer id);
}
```

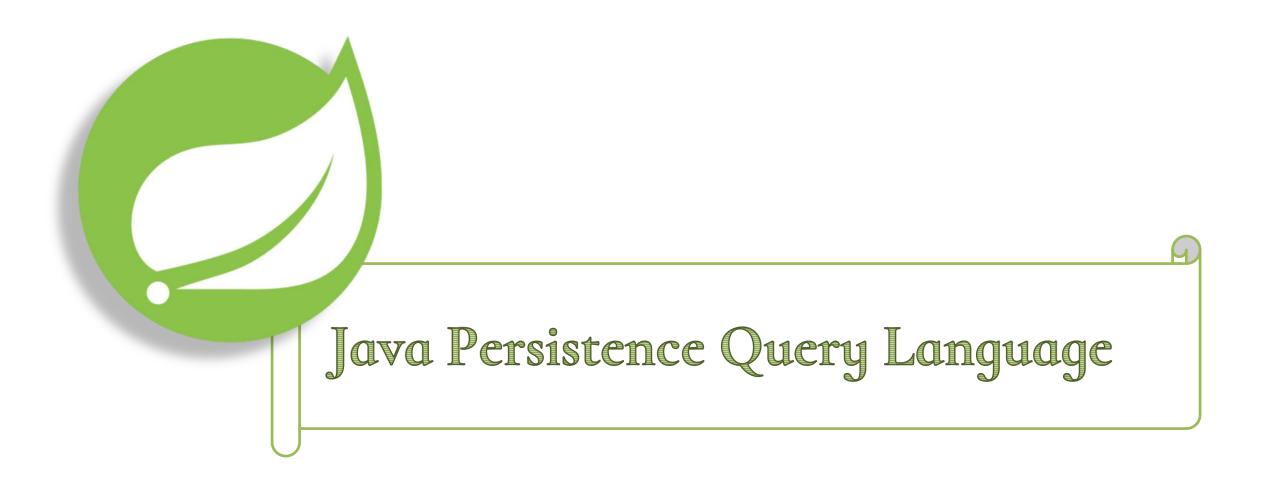


### **SQL** = **STRUCTURE QUERY LANGUAGE**

```
SELECT * FROM Products p
INNER JOIN Categories c ON c.Id=p.CategoryId
WHERE c.Name LIKE ?1
(keyword) => List<Product>
```

# JPQL = JAVA PERSISTENT QUERY LANGUAGE

SELECT p FROM Product p
WHERE p.category.name LIKE ?1
(keyword) => List<Product>





### JPAREPOSITORY - @QUERY(JPQL)

```
public interface ProductDAO extends JpaRepository<Product, Integer>{
   @Query("SELECT p FROM Product p "
              + " WHERE p.unitPrice BETWEEN ?1 AND ?2")
   List<Product> findByPrice(double min, double max);
   @Query("SELECT p FROM Product p WHERE p.category.id=:cid")
   List<Product> findByCategoryId(@Param("cid") Integer id);
   @Query("SELECT p FROM Product p "
                                                     @Query(JPQL)
              + " WHERE p.name LIKE ?1 "
                                                     Type Method([arguments])
              + " OR p.category.name LIKE ?1 "
              + " OR p.category.nameVn LIKE ?1")
   List<Product> findByKeyword(String keyword);
```



### JPQL - JAVA PERSISTENT QUERY LANGUAGE

#### ■ Syntax

```
SELECT expr
FROM Entity e
WHERE <condition>
GROUP BY <expr>
ORDER BY <expr> [<direction>]
```

- ☐ Tham số
  - ❖ Vị trí: ?1
  - \*Tên: @Param("name") và ":name"
- ☐ Toán tử, toán hạng, hàm
  - Toán tử: tương tự SQL
  - \*Toán hạng: entity và property
  - ❖ Hàm: JPA

```
SELECT p
FROM Product p
WHERE p.unitPrice BETWEEN ?1 AND ?2
ORDER BY p.unitPrice DESC
(min, max) => List<Product>
```

```
SELECT a
FROM Account a
WHERE a.username=?1 AND a.password=?2
(username, password) => Account
```

```
SELECT p.category.name, sum(p.quantity)
FROM Product p
WHERE year(p.productDate)=:year
GROUP BY p.category.name
ORDER BY sum(p.quantity)
(year) => List<Object[]>
```

# Java

### JPQL - JAVA PERSISTENT QUERY LANGUAGE

- @Query("SELECT e FROM Product e WHERE e.name LIKE ?1 OR e.category.name LIKE ?1")
- **List<Product>** findByKeyword(String keyword)
- @Query("SELECT e FROM Account e WHERE e.email = :email")
- **Account** findByEmail(@Param("email") String email)
- @Query("SELECT e FROM Product e WHERE e.unitPrice BETWEEN ?1 AND ?2")
- **List<Product>** findByPrice(double min, double max)
- @Query("SELECT e FROM Order e WHERE e.account.username=?1")
- **List<Order>** findByUsername(String username)
- @Query("SELECT e.name FROM Product e WHERE year(e.productDate)=?1")
- **List<String>** findProductNamesByYear(int year)
- @Query("SELECT min(e.unitPrice) FROM Product e WHERE e.category.id=?1")
- double findMinPriceByCategoryId(int categoryId)



## TRUY VẤN THUỘC TÍNH

```
SELECT p.name, p.unitPrice
FROM Product p

WHERE p.orderDetails IS EMPTY

() => List<Object[]>
```

```
SELECT p.category.name,

sum(p.quantity * p.unitPrice * (1–p.discount),

sum(p.quantity)

FROM Product p

GROUP BY p.category.name

() => List<Object[]>
```

```
SELECT month(d.order.orderDate),
sum(d.quantity * d.unitPrice * (1–d.discount),
sum(d.quantity),
min(d.unitPrice),
max(d.unitPrice),
avg(d.unitPrice)
FROM OrderDetail d
GROUP BY month(d.order.orderDate)
ORDER BY month(d.order.orderDate)
```

```
SELECT d.product.category.name,
sum(d.quantity * d.unitPrice * (1–d.discount),
sum(d.quantity),
min(d.unitPrice),
max(d.unitPrice),
avg(d.unitPrice)

FROM OrderDetail d
GROUP BY d.product.category.name
```

```
SELECT d.order.account.username,

sum(d.quantity * d.unitPrice * (1-d.discount),

sum(q.quantity),

min(d.unitPrice),

max(d.unitPrice),

avg(d.unitPrice)

FROM OrderDetail d

GROUP BY d.order.account.username

ORDER BY sum(d.quantity)
```

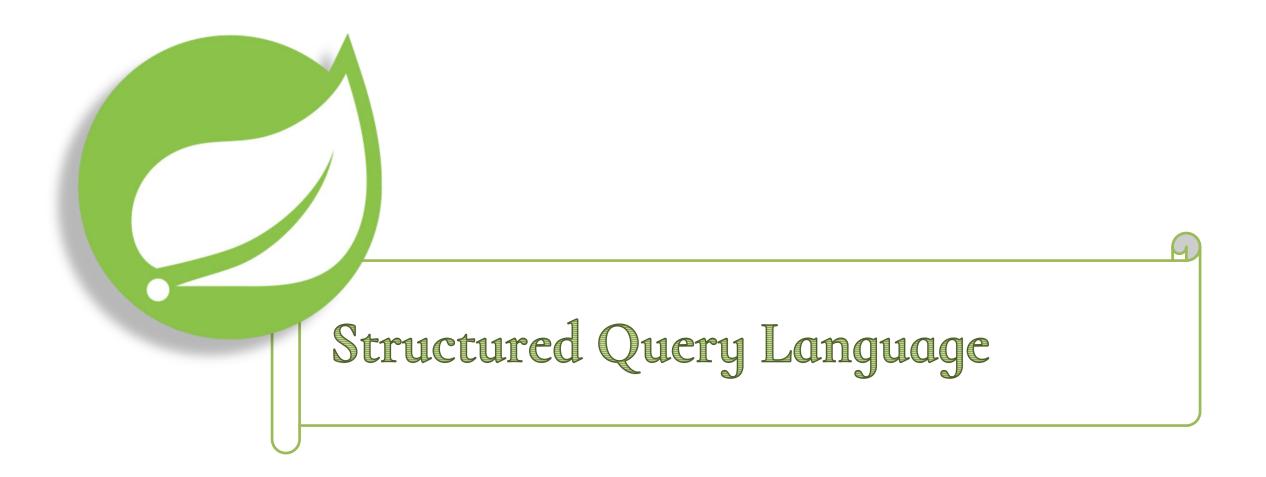


# **S**ử DỤNG ENTITY THAY **O**BJECT[]

```
SELECT p.name, p.unitPrice
FROM Product p
WHERE p.orderDetails IS EMPTY
() => List<Object[]>
```

```
SELECT new MyEntity(p.name, p.unitPrice)
FROM Product p
WHERE p.orderDetails IS EMPTY
() => List<MyEntity>
```

```
@AllArgsConstructor
@Data
@Entity
public class MyEntity{
   @Id
   String name;
   Double price;
```





### @Query(value=SQL, nativeQuery=true)

```
@Query(value=SQL, nativeQuery=true)
([arguments]) => Type
public interface ProductDAO extends JpaRepository<Product, Integer>{
   @Query(value="SELECT * FROM Products "
            + "WHERE UnitPrice BETWEEN ?1 AND ?2", nativeQuery=true)
   List<Product> findByPrice(double min, double max);
   @Query(value="SELECT p.* FROM Products p
            + "INNER JOIN Categories c ON c.Id=p.CategoryId
            + " WHERE p.CategoryId=:cid", nativeQuery=true)
   List<Product> findByCategoryId(@Param("cid") Integer id);
```



### Truy vấn với câu lệnh SQL đặc thù

```
SELECT * FROM Products WHERE Name LIKE :keyword
SELECT * FROM Products WHERE UnitPrice BETWEEN ?1 AND ?2
SELECT COUNT(*) FROM Products WHERE UnitPrice BETWEEN ?1 AND ?2
SELECT c.Name,
   sum(d.Quantity * d.UnitPrice * (1-d.Discount)),
   sum(d.Quantity),
   min(d.UnitPrice),
   max(d.UnitPrice),
   avq(d.UnitPrice)
FROM OrderDetails d
   INNER JOIN Products p ON p.Id=d.ProductId
   INNER JOIN Categories c ON c.Id=p.CategoryId
GROUP BY c.Name
ORDER BY sum(d.Quantity)
```







- ☐ Thêm *Sort* hoặc *Pageable* vào đối số của phương thức truy vấn để thực hiện sắp xếp hoặc phân trang.
- □Đối số Sort hoặc Pageable luôn là đối số cuối cùng của phương thức truy vấn.



- **☑** @Query(JPQL)
- ☑ @Query(value=SQL, nativeQuery=true)
- **☑** Sort & Paginate

