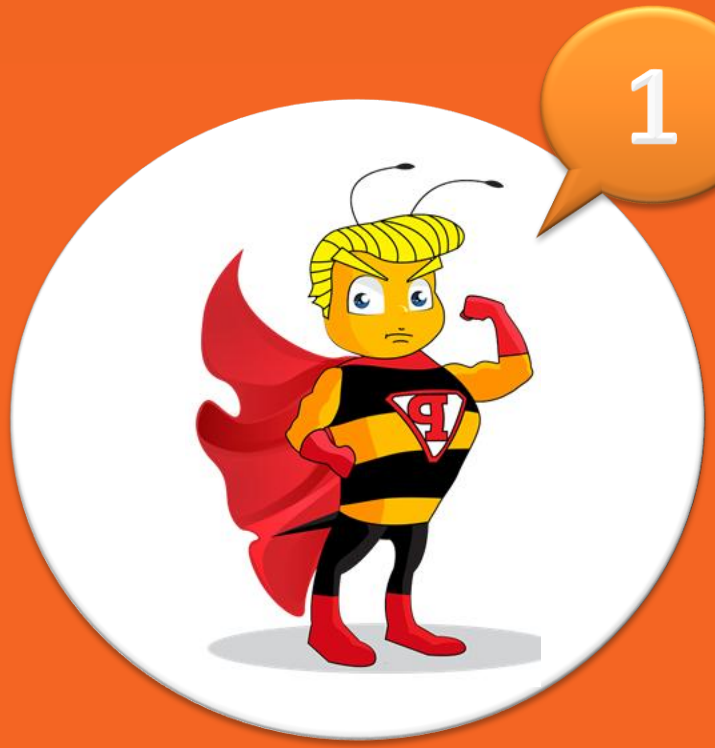
FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate



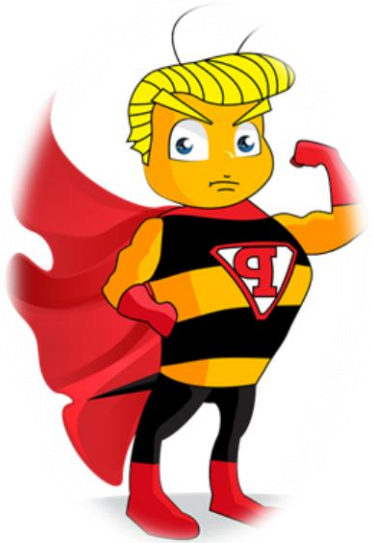
JPAREPOSITORY API - @QUERY

SLIDE 7.1

www.poly.edu.vn

AGENDA

- PHẦN I: TRUY VẤN TÙY BIẾN VỚI @QUERY
  - GIỚI THIỆU @QUERY
  - THAM SỐ TRUY VẤN
  - TÙY BIẾN SẮP XẾP VÀ PHÂN TRANG
  - TRUY VẤN TỔNG HỢP
- PHẦN II:
  - @NAMEDQUERY
  - TRUY VẤN ĐẶC THÙ VỚI SQL
  - NGÔN NGỮ DSL (DOMAIN SPECIFIC LANGUAGE)



@Query Introduction

TRUY VẤN NÂNG CAO

- JpaRepository API cung cấp các phương thức truy vấn và thao tác dữ liệu rất đa dạng và phong phú. Tuy nhiên trong thực tế những yêu cầu truy vấn phức tạp có độ tùy biến cao mà những gì đã cung cấp chưa thể giải quyết được.
  - Ví dụ: Truy vấn những sản phẩm có giá trong một phạm vi nào đó (từ min đến max). Với yêu cầu này thì không có phương thức nào của JpaRepository giúp chúng ta thực hiện được.
- Để tăng cường khả năng truy vấn, JpaRepository cung cấp 2 giải pháp truy vấn linh hoạt
  - @Query
  - DSL (Domain Specific Language)

GIỚI THIỆU @QUERY VÀ DSL

- Tìm hiểu @Query và DSL qua các ví dụ "Truy vấn các sản phẩm có giá trong khoản từ min đến max"

```
public interface ProductDAO extends JpaRepository<Product, Integer> {
        @Query("FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2")
        List<Product> findByUnitPrice(double min, double max);

        List<Product> findByUnitPriceBetween(double min, double max);
}
```

- Cả 2 phương thức được bổ sung vào DAO cho cùng 1 kết quả để giải quyết yêu cầu ở trên.
  - findByUnitPrice() dựa vào JPQL của ***@Query*** để sinh mã và thực hiện
  - findBy***UnitPriceBetween***() dựa vào tên phương thức để sinh mã và thực hiện

@QUERY(VALUE, NAME, NATIVEQUERY)

- @Query được sử dụng để chú thích cho 1 phương thức truy vấn dữ liệu.
- Các đối số:
  - Value là câu lệnh JPQL hoặc SQL (nếu nativeQuery=true)
  - Name: tên của @NamedQuery đã được khai báo trong entity
  - *Chú ý: value và name không thể xuất hiện đồng thời*
- Ví dụ: Truy vấn các sản phẩm có giá > 5

```
// JPQL - Đầy đủ theo cú pháp
@Query(value="SELECT o FROM Product o WHERE o.unitPrice > 5" , nativeQuery=false)

// SQL – phụ thuộc vào hệ quản trị CSDL (MySQL Server có thể khác với SQL Server)
@Query(value="SELECT * FROM Products WHERE UnitPrice > 5", nativeQuery=true)
```

- Thay đổi cách viết @Query tùy thuộc vào cấu hình mặc định của JpaRepository

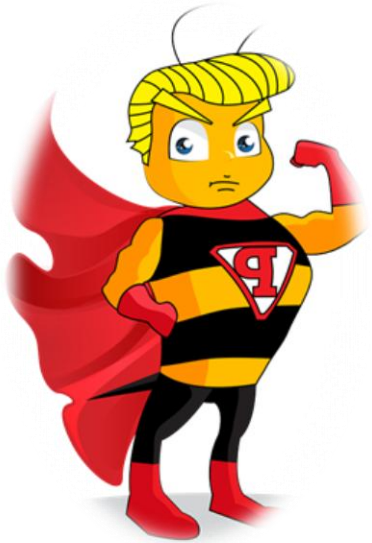
```
// JPQL
// Đầy đủ theo cú pháp
@Query(value="SELECT o FROM Product o WHERE o.unitPrice > 5" , nativeQuery=false)

// Bỏ nativeQuery vì false là giá trị mặc định
@Query(value="SELECT o FROM Product o WHERE o.unitPrice > 5");

// Bỏ SELECT vì mặc định SELECT chọn nguyên đối tượng
@Query(value="FROM Product o WHERE o.unitPrice > 5")

// Value là thuộc tính mặc định nên có thể bỏ
@Query("FROM Product o WHERE o.unitPrice > 5")

// SQL – phụ thuộc vào hệ quản trị CSDL (MySQL Server có thể khác với SQL Server)
@Query(value="SELECT * FROM Products WHERE UnitPrice > 5", nativeQuery=true)
```

SỬ DỤNG PHƯƠNG THỨC TRUY VẤN TÙY BIẾN VỚI @QUERY

```
public interface ProductService  extends CrudService<Product, Integer>{
        List<Product> findByPrice(double minPrice, double maxPrice);
}
```

```
@Service("productService")
public class ProductServiceImpl implements ProductService {
        @Autowired
        ProductDAO dao;

        @Override
        public List<Product> findByPrice(double min, double max) {
                return dao.findByUnitPrice(min, max);
        }
}
```

```
public interface ProductDAO extends JpaRepository<Product, Integer> {
        @Query("FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2")
        List<Product> findByUnitPrice(double min, double max);
}
```



JPQL Parameters

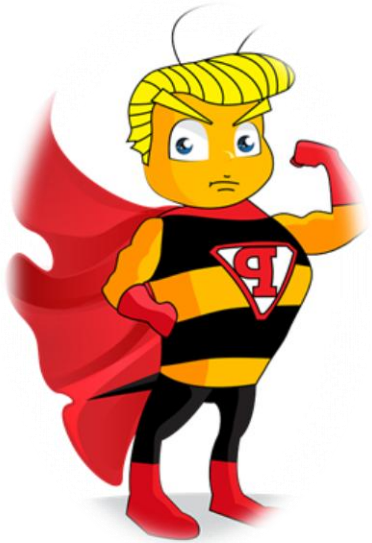
THAM SỐ VÀ TRUYỀN THAM SỐ CHO CÂU LỆNH TRUY VẤN

- JPQL có thể chứa tham số theo 2 hình thức là vị trí và tên tham số
  - Vị trí: FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2
  - Tên: FROM Product o WHERE o.unitPrice BETWEEN :min AND :max
- Truyền giá trị cho các tham số

```
// Đối số đầu tiên của phương thức được truyền vào ?1 của JPQL/SQL
@Query("FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2")
List<Product> findByUnitPrice(double min, double max);

// Sử dụng @Param để chỉ định đối số của phương thức được truyền vào tham số của JPQL/SQL
@Query("FROM Product o WHERE o.unitPrice BETWEEN :min AND :max")
List<Product> findByUnitPrice(@Param("min") double min, @Param("max") double max);
```

MỘT SỐ TRUY VẤN KHÁC

- FROM Product p WHERE p.name LIKE **?1**
  - **List<Product>** findByKeyword(String **keyword**)
- FROM Product p WHERE p.name LIKE **:keyword**
  - **List<Product>** findByKeyword(@Param("**keyword**") String keyword)
- FROM Product p WHERE p.price BETWEEN **?1** AND **?2**
  - **List<Product>** findByPrice(double **min**, double **max**)
- FROM Product p WHERE p.price BETWEEN **:min** AND **:max**
  - **List<Product>** findByPrice(@Param("**min**")double min, @Param("**max**")double max)
- SELECT p.name FROM Product p WHERE p.price BETWEEN **?1** AND **?2**
  - **List<String>** findNamesByPrice(double **min**, double **max**)
- SELECT min(p.price) FROM Product p WHERE p.price BETWEEN **?1** AND **?2**
  - **double** findMin(double **min**, double **max**)



TÙY BIẾN SẮP XẾP VÀ PHÂN TRANG

TRUY VẤN CÓ SẮP XẾP

- Câu lệnh JPQL hỗ trợ mệnh đề ORDER BY để bạn thực hiện việc sắp xếp kết quả truy vấn
- Để tùy biến việc sắp xếp theo ý riêng thì bạn chỉ cần bổ sung đối số Sort vào phương thức truy vấn.

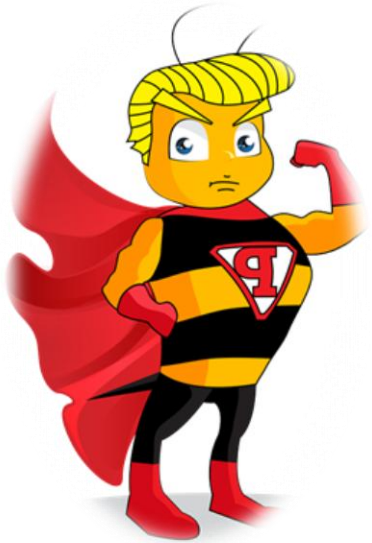
```
@Query("FROM Product o WHERE o.unitPrice < ?1 ORDER BY o.unitPrice DESC")
List<Product> findByUnitPrice(double max);

@Query("FROM Product o WHERE o.unitPrice < ?1")
List<Product> findByUnitPrice(double max, Sort sort);
```

TRUY VẤN CÓ PHÂN TRANG

- Để thực hiện phân trang với phương thức truy vấn tùy biến bạn cần
  - 1. Bổ sung đối số Pageable vào phương thức truy vấn
  - 2. Khai báo kiểu kết quả trả về là Page<T>

```
@Query("FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2")
Page<Product> findByUnitPrice(double min, double max, Pageable pageable);
```

- Tiêm dao vào Spring Bean và sử dụng
  - Page<Product> page = dao.findByUnitPrice(5.0, 9.5, PageRequest.of(1, 8));



TRUY VẤN TỔNG HỢP

CASE STUDY

- Hãy tổng hợp số liệu bán hàng từ OrderDetail để cung cấp các thông tin có cấu trúc như sau:

LOẠI HÀNG	DOANH THU	SỐ LƯỢNG BÁN

- Chúng ta cần câu lệnh JPQL như sau

```
SELECT d.product.category,
    sum(d.unitPrice * d.quantity),
    sum(d.quantity)
FROM OrderDetail d
GROUP BY d.product.category
```

- Với câu lệnh truy vấn này thì kết quả sẽ là **List<Object[]>**, mỗi phần tử của List là một mảng đối tượng **[Category, Double, Long]**

TRUY VẤN TỔNG HỢP

Kết quả truy vấn: Mỗi phần tử của List là 1 mảng gồm 3 phần tử (Category, Amount, Quantity) => Bí ẩn, khó nhớ, khó lập trình.

CÁCH 1:

```
@Query("SELECT o.product.category, "
        + " sum(o.unitPrice * o.quantity), "
        + " sum(o.quantity) "
        + " FROM OrderDetail o "
        + " GROUP BY o.product.category")
List<Object[]> getReportItems();
```

CÁCH 2:

```
@Query("SELECT o.product.category AS category, "
        + " sum(o.unitPrice * o.quantity) AS totalAmount, "
        + " sum(o.quantity) AS totalQuantity "
        + " FROM OrderDetail o "
        + " GROUP BY o.product.category")
List<ReportItem> getReportItems();
```

```
public interface ReportItem {
    Category getCategory();
    long getTotalQuantity();
    double getTotalAmount();
}
```

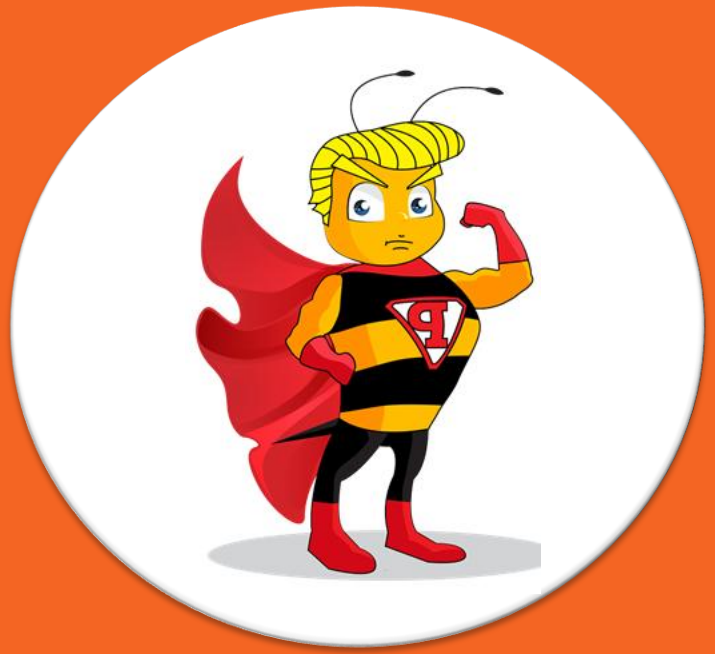
Kết quả truy vấn: Mỗi phần tử của List là 1 đối tượng ReportItem => được công cụ hỗ trợ gợi nhớ

FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate



JpaRepository API - DSL

Slide 7.2

www.poly.edu.vn

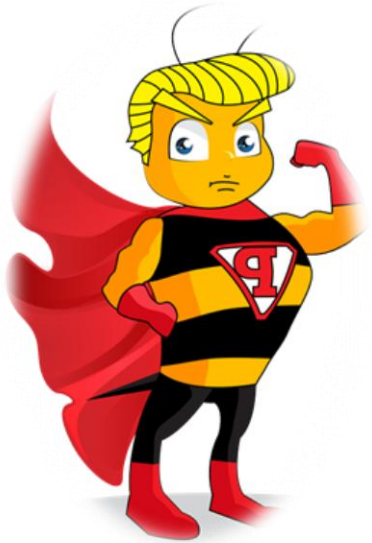
TRUY VẤN SỬ DỤNG @NAMEDQUERY

```
@Query(name = "findByKeyword")
List<Product> findByKeywords(String keywords);
```

```
@NamedQueries({
        @NamedQuery(name = "findByPrice",
                query = "FROM Product o WHERE o.unitPrice BETWEEN ?1 AND ?2"),
        @NamedQuery(name = "findByKeyword",
                query = "FROM Product o WHERE o.name LIKE ?1")
})
@Entity
@Table(name = "Products")
public class Product {...}
```

TRUY VẤN VỚI CÂU LỆNH SQL ĐẶC THÙ

- JpaRepository cho phép sử dụng câu lệnh SQL thay vì JPQL để truy vấn dữ liệu.
  - Ưu điểm: Không cần học thêm JPQL, tận dụng được các hàm có sẵn trong hệ quản trị CSDL
  - Nhược điểm: Khó năng cấp, mở rộng vì SQL phụ thuộc vào hệ quản trị CSDL
- Ví dụ
  - @Query(**value** = "SELECT * FROM Products WHERE Name LIKE ?1", **nativeQuery** = **true**)
  - List<Product> findByKeyword(String keyword);
  - @Query(value = "SELECT * FROM Products WHERE Price BETWEEN ?1 AND ?2", **nativeQuery = true**)
  - List<Product> findByPrice(double min, double max);
  - @Query(value = "SELECT COUNT(*) FROM Products WHERE Price BETWEEN ?1 AND ? 2", **nativeQuery = true**)
  - Long countByPrice(double min, double max);



DSL

QUERY DSL (DOMAIN SPECIFIC LANGUAGE)

- Ngoài @Query(), JpaRepository còn đơn giản hóa việc truy vấn dữ liệu bằng cách khai báo tên phương thức đúng cú pháp quy định mà không cần phải viết câu lệnh JPQL.
- JpaRepository dựa vào tên của phương thức truy vấn để sinh ra JPQL và mã thực thi truy vấn.
- Cú pháp đặt tên phương thức truy vấn dựa vào một miền dữ liệu cụ thể nói trên được gọi là DSL.
- Ví dụ:
  - Với @Query

    ```
    @Query("FROM Product p WHERE p.name LIKE ?1")
    List<Product> searchProduct(String name)
    ```

  - DSL

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

CÚ PHÁP ĐẶT TÊN PHƯƠNG THỨC TRUY VẤN

- JpaRepository xác định miền dữ liệu dựa vào T trong biểu thức tổng quát kế thừa của interface JpaRepository<**T**, ID>
- **findByNameLike**(x)
  - **findBy_**, **getBy_**, **countBy_** là các tiền tố truy vấn được chấp nhận trong JpaRepository
  - **Name** là tên thuộc tính của miền dữ liệu
  - **Like** là toán tử áp dụng cho thuộc tính đã được chỉ ra kế trước
- Ví dụ
  - findBy**PriceIsNull**() ~ findBy**PriceNull**()
  - findBy**PriceEqual**(double max) ~ findBy**Price**(double max)
  - findBy**PriceLessThan**(double max)
  - findBy**PriceBetween**(double min, double max)
  - findBy**PriceLessThanEqualAndNameLike**(double max, String name)

CÁC QUY ƯỚC CỦA DSL

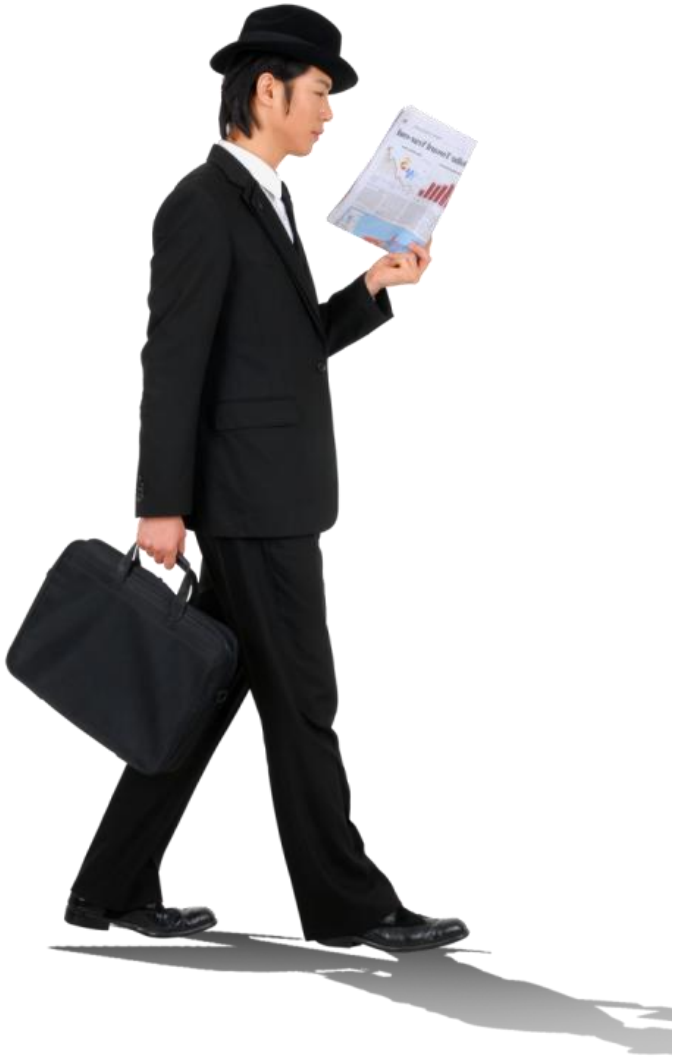
TỪ KHÓA	VÍ DỤ	JPQL TƯƠNG ĐƯƠNG
AND	*findBy*Lastname**And**Firstname	...*WHERE* x.lastname **=** ?1 **AND** x.firstname **=** ?2
OR	*findBy*Lastname**Or**Firstname	...*WHERE* x.lastname **=** ?1 **OR** x.firstname **=** ?2
IS, EQUALS	*findBy*Firstname**Equals**	...*WHERE* x.firstname **=** ?1
BETWEEN	*findBy*StartDate**Between**	...*WHERE* x.startDate **BETWEEN** ?1 and ?
LESSTHAN	*findBy*Age**LessThan**	...*WHERE* x.age **<** ?1
LESSTHANEQUAL	*findBy*Age**LessThanEqual**	...*WHERE* x.age **<=** ?1
GREATERTHAN	*findBy*Age**GreaterThan**	...*WHERE* x.age **>** ?1
GREATERTHANEQUAL	*findBy*Age**GreaterThanEqual**	...*WHERE* x.age **>=** ?1
AFTER	*findBy*StartDate**After**	...*WHERE* x.startDate **>** ?1
BEFORE	*findBy*StartDate**Before**	...*WHERE* x.startDate **<** ?1
ISNULL	*findBy*Age**IsNull**	...*WHERE* x.age **IS NULL**
ISNOTNULL, NOTNULL	*findBy*Age**[Is]NotNull**	...*WHERE* x.age **IS NOT NULL**

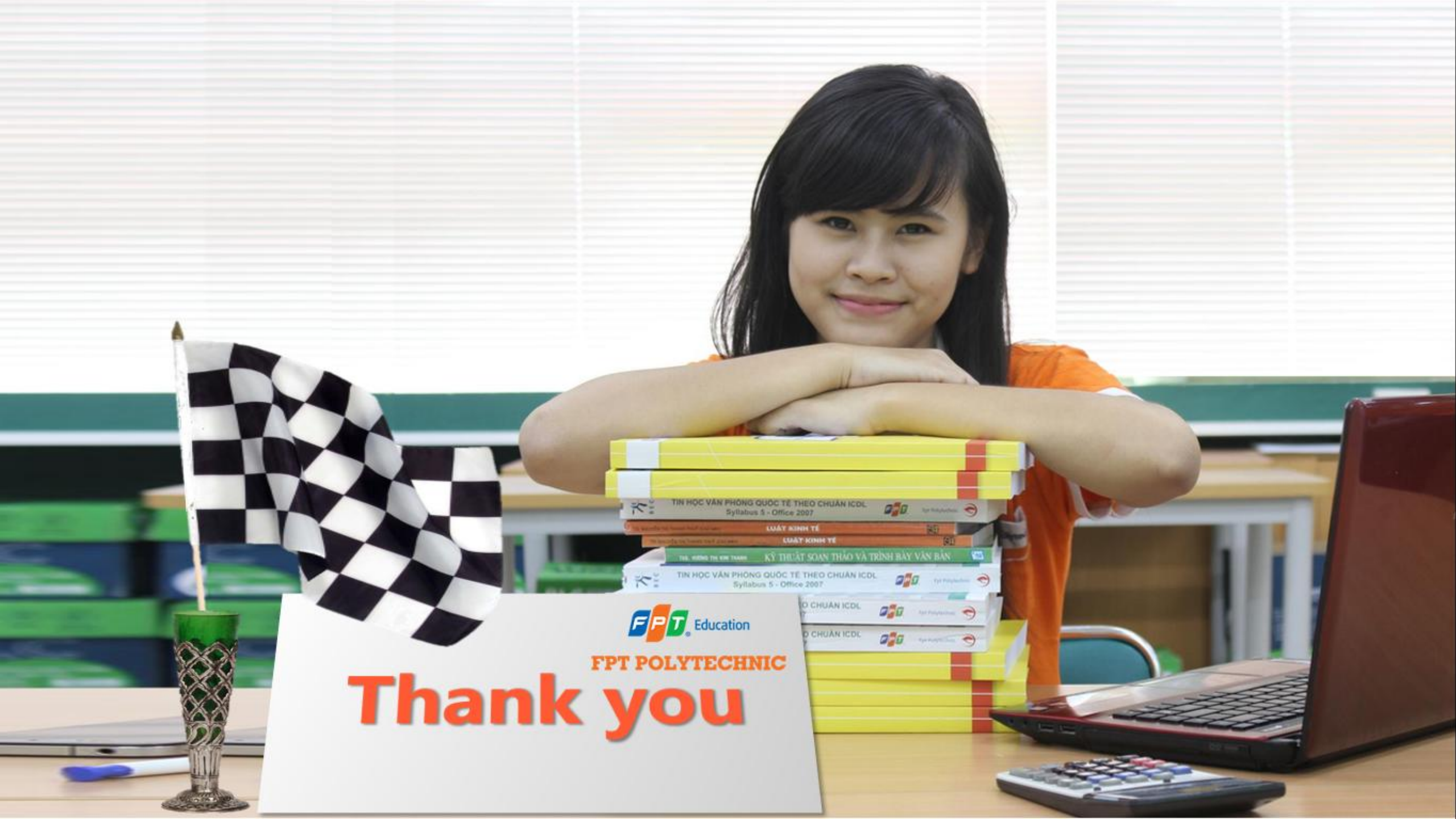
CÁC QUY ƯỚC CỦA DSL

TỪ KHÓA	VÍ DỤ	JPQL TƯƠNG ĐƯƠNG
Like	*findBy*Firstname**Like**	...*WHERE* x.firstname **LIKE** ?1
NotLike	*findBy*Firstname**NotLike**	...*WHERE* x.firstname **NOT LIKE** ?1
StartingWith	*findBy*Firstname**StartingWith**	...*WHERE* x.firstname **LIKE** ?1[%]
EndingWith	*findBy*Firstname**EndingWith**	...*WHERE* x.firstname **LIKE** [%]?1
Containing	*findBy*Firstname**Containing**	...*WHERE* x.firstname **LIKE** [%]?1[%]
OrderBy	*findBy*Age**OrderBy**Lastname**Desc**	...*WHERE* x.age = ?1 **ORDER BY** x.lastname **DESC**
Not	*findBy*Lastname**Not**	...*WHERE* x.lastname **<>** ?1
In	*findBy*Age**In**(Collection ages)	...*WHERE* x.age **IN** ?1
NotIn	*findBy*Age**NotIn**(Collection ages)	...*WHERE* x.age **NOT IN** ?1
True	*findBy*Active**True**()	...*WHERE* x.active **= true**
False	*findBy*Active**False**()	...*WHERE* x.active **= false**
IgnoreCase	*findBy*Firstname**IgnoreCase**	...*WHERE* **upper**(x.firstame) **= upper**(?1)

SUMMARY

- ☑ PHẦN I: TRUY VẤN TÙY BIẾN VỚI @QUERY
  - ☑ GIỚI THIỆU @QUERY
  - ☑ THAM SỐ TRUY VẤN
  - ☑ TÙY BIẾN SẮP XẾP VÀ PHÂN TRANG
  - ☑ TRUY VẤN TỔNG HỢP
- ☑ PHẦN II:
  - ☑ @NAMEDQUERY
  - ☑ TRUY VẤN ĐẶC THÙ VỚI SQL
  - ☑ NGÔN NGỮ DSL (DOMAIN SPECIFIC LANGUAGE)




FPT Education
FPT POLYTECHNIC
Thank you