



JAVA PERSISTENCE QUERY LANGUAGE

LẬP TRÌNH JAVA #4 (P4.1)

- Giới thiệu JPQL
- Cấu trúc lệnh JPQL
- Hàm JPQL



- ❑ **JPQL** = Java Persistence Query Language
- ❑ JPQL là ngôn ngữ truy vấn đối tượng (thực thể) có cú pháp tương tự SQL nhưng thay vì **Table** và **Column** thì sử dụng **Entity** và **Property** (getters/setters)
 - ❖ **Entity** thay vì **Table**
 - ❖ **Property** thay vì **Column**

- ❑ Ví dụ: Truy vấn video có mã yêu thích là 1234

SELECT o.video FROM Favorite o WHERE o.id=1234

- ❖ Trong đó:
 - **Favorite** hiểu là Entity Class
 - **o.video** hiểu là **o.getVideo()**
 - **o.id** hiểu là **o.getId()**

❑ SELECT

```
SELECT ... FROM ...
[WHERE ...]
[GROUP BY ... [HAVING ...]]
[ORDER BY ...]
```

❑ UPDATE

```
UPDATE ... SET ... [WHERE ...]
```

❑ DELETE

```
DELETE FROM ... [WHERE ...]
```

THỰC HIỆN TRUY VẤN VỚI JPQL

Sử dụng getResultList() vì kết quả là List<User> (nhiều)

```
String jpql = "SELECT o FROM User o"; // Kết quả: List<User>
TypedQuery<User> query = em.createQuery(jpql, User.class);
List<User> list = query.getResultList();
```

```
String jpql = "SELECT o FROM User o WHERE o.id='TeoNV"'; // Kết quả: User
TypedQuery<User> query = em.createQuery(jpql, User.class);
User user = query.getSingleResult();
```

Sử dụng getSingleResult() vì kết quả là User (một)



DEMOSTATION

- ❑ Kết quả truy vấn với JPQL có thể
 - ❖ Một: T
 - ❖ Nhiều: List<T>
- ❑ Để nhận diện được T hay List<T> thì phải căn cứ vào câu lệnh JPQL, đặc biệt mệnh đề SELECT và WHERE.
 - ❖ Dựa vào SELECT để biết T
 - ❖ Dựa vào WHERE để biết một hay nhiều
- ❑ Ví dụ: **SELECT o FROM User o WHERE o.id='TeoNV'**
 - ❖ SELECT o, với o là bí danh của User nên xác định được T là User
 - ❖ WHERE o.id='TeoNV' với id là khóa chính nên kết quả nếu có chỉ là 1
 - ❖ => Vì vậy kết quả của việc thực hiện câu lệnh này là User

- SELECT o FROM User o**
 - ❖ *Truy vấn tất cả User. Kết quả: List<User>*
- SELECT o FROM User o WHERE o.id='TeoNV'**
 - ❖ *Truy vấn User có id là TeoNV. Kết quả: User*
- SELECT o.id, o.password FROM User o**
 - ❖ *Truy vấn id và password tất cả user. Kết quả: List<Object[]>*
- SELECT o.video FROM Favorite o WHERE o.id=1234**
 - ❖ *Truy vấn các video id có yêu thích. Kết quả: Video*
- SELECT DISTINCT o.video.id FROM Favorite o**
 - ❖ *Truy vấn các video id có yêu thích. Kết quả: List<String>*
- SELECT o FROM Favorite o WHERE year(o.likeDate) BETWEEN 2020 AND 2030**
 - ❖ *Truy vấn các yêu thích trong khoảng thời gian từ 2020 đến 2030. Kết quả: List<Favorite>*
- SELECT o.user.id, count(o) FROM Favorite o GROUP BY o.user.id ORDER BY count(o) DESC**
 - ❖ *Truy vấn các user id và số lượng video đã thích. Kết quả: List<Object[]>*

```
String jpql = "SELECT o FROM User o WHERE o.id='TeoNV"'; // Kết quả: User  
TypedQuery<User> query = em.createQuery(jpql, User.class);  
User user = query.getSingleResult();
```

```
String jpql = "SELECT count(o) FROM User o"; // Kết quả: Long  
TypedQuery<Long> query = em.createQuery(jpql, Long.class);  
Long count = query.getSingleResult();
```

```
String jpql = "SELECT o.id, o.password FROM User o"; // Kết quả: List<Object[ ]>  
TypedQuery<Object[ ]> query = em.createQuery(jpql, Object[ ].class);  
List<Object[ ]> list = query.getResultList();
```

```
String jpql = "SELECT o.video FROM Favorite o WHERE o.user.id='TeoNV"'; // List<Video>
TypedQuery<Video> query = em.createQuery(jpql, Video.class);
List<Video> list = query.getResultList();
```

```
String jpql = "SELECT o.video FROM Favorite o WHERE o.id=1234"; // Video
TypedQuery<Video> query = em.createQuery(jpql, Video.class);
Video video = query.getSingleResult();
```

```
String jpql = "SELECT DISTINCT o.video.id FROM Favorite o"; // List<String>
TypedQuery<String> query = em.createQuery(jpql, String.class);
List<String> list = query.getResultList();
```

❑ Câu lệnh:

- ❖ JPQL: SELECT o.id, o.password FROM User o
- ❖ Kết quả: List<Object[]>

@NoArgsConstructor
@AllArgsConstructor
@Data

❑ Câu lệnh:

- ❖ JPQL: SELECT new Auth(o.id, o.password)
FROM User o

@Entity
public class Auth {
 @Id
 String username;
 String password;
}

- ❖ Kết quả: List<Auth>
- ❖ Trong đó: Auth là 1 Entity Class không map với bất kỳ bảng nào trong CSDL

- ❑ JPQL cho phép sử dụng các toán tử số học, so sánh và quan hệ thông thường như trong SQL
 - ❖ Số học: +, -, *, /
 - ❖ So sánh: >, >=, <, <=, !=
 - ❖ Quan hệ: AND, OR, NOT
- ❑ Toán tử đặc biệt
 - ❖ [NOT] IN
 - ❖ [NOT] BETWEEN
 - ❖ IS [NOT] NULL
 - ❖ [NOT] LIKE
 - ❖ IS [NOT] EMPTY

- SELECT o FROM User o WHERE o.email **LIKE** '%@fpt.edu.vn'
 - ❖ *Truy vấn các User có email kết thúc bởi @fpt.edu.vn*
- SELECT o FROM Favorite o
 - WHERE year(o.likeDate) **BETWEEN** 2020 **AND** 2030 ORDER BY o.likeDate DESC
 - ❖ *Truy vấn và sắp xếp giảm dần các Favorite thực hiện từ 2020 đến 2030*
- SELECT o FROM Video o WHERE o.description **IS NOT NULL**
 - ❖ *Truy vấn các video ko có mô tả*
- SELECT o FROM Favorite o WHERE o.user.id **IN** ('PheoNC', 'NoPT')
 - ❖ *Truy vấn các Favorite của Chí Phèo (PheoNC) và Thị Nở (NoPT)*
- SELECT count(o) FROM Video o WHERE o.favorites **IS EMPTY**
 - ❖ *Truy vấn các Video chưa được yêu thích*

- ❑ JPQL cung cấp tập hợp các hàm phong phú đủ hỗ trợ bạn xử lý mọi vấn đề ngay trong khi truy vấn mà không cần phải viết mã phức tạp trong Java.
- ❑ Hàm xử lý chuỗi
 - ❖ Length(), Trim(), Lower(), Upper(), Substring(), Concat(), Locate()...
- ❑ Hàm xử lý thời gian
 - ❖ Year(), Month(), Day(), Hour(), Minute(), Second(), Current_Date(), Current_Time(), Current_Timestamp()
- ❑ Hàm tổng hợp số liệu
 - ❖ Sum(), Count(), Min(), Max(), Avg(), Size()
- ❑ Các hàm khác
 - ❖ Sqrt(), str(), cast(), ceil(), floor()

- ❑ Length(String): int
 - ❖ *Lấy độ dài chuỗi*
- ❑ Trim(): String
 - ❖ *Cắt bỏ khoảng trắng 2 đầu chuỗi*
- ❑ Lower(String): String
 - ❖ *Đổi sang ký tự thường*
- ❑ Upper(String): String
 - ❖ *Đổi sang ký tự hoa*
- ❑ Substring(String, int, int): String
 - ❖ *Lấy chuỗi con*
- ❑ Concat(String, String): String
 - ❖ *Ghép chuỗi*
- ❑ Locate(String, String): int
 - ❖ *Tìm vị trí xuất hiện của chuỗi con*

Ví dụ: Truy vấn họ tên *in hoa*, email *in thường* của những User **họ Nguyễn**:

```
SELECT upper(o.fullname), lower(o.email)  
FROM User o  
WHERE locate(trim(o.fullname), 'Nguyễn')
```

- Year(Date): int
 - ❖ *Lấy năm 4 chữ số*
- Month(Date): int
 - ❖ *Lấy tháng (1..12)*
- Day(Date): int
 - ❖ *Lấy ngày (1..31)*
- Hour(Date): int
 - ❖ *Lấy giờ (0..23)*
- Minute(Date): int
 - ❖ *Lấy phút (0..59)*
- Second(Date): int
 - ❖ *Lấy giây (0..59)*
- Current_Date(): Date
 - ❖ *Lấy thời gian hiện tại (chỉ ngày-tháng-năm)*
- Current_Time(): Date
 - ❖ *Lấy thời gian hiện tại (chỉ giờ-phút-giây)*
- Current_Timestamp(): Date
 - ❖ *Lấy thời gian hiện tại (đầy đủ)*

Ví dụ: Truy vấn tiêu đề và tháng yêu thích trong vòng 5 năm đổ lại:

```
SELECT o.video.title AS Title, month(o.likeDate)  
AS Month
```

```
FROM Favorite o
```

```
WHERE (year(current_date)) -  
year(o.likeDate)) > 5
```

- ❑ Sum(Double): Double
 - ❖ Tổng
- ❑ Count(T): int
 - ❖ Đếm
- ❑ Min(T): T
 - ❖ Nhỏ nhất
- ❑ Max(T): T
 - ❖ Lớn nhất
- ❑ Avg(Double): Double
 - ❖ Trung bình
- ❑ Size(List<T>): int
 - ❖ Số lượng

Ví dụ: *Tổng hợp số lượt thích, ngày đầu tiên và ngày cuối cùng được thích của từng video:*

```
SELECT
    o.video.title AS Title,
    count(o) AS LikeCount,
    min(o.likeDate) AS FirstDate
    max(o.likeDate) AS LastDate
FROM Favorite o
GROUP BY o.video.title
```

- ❑ **Sqrt(Double): Double**
 - ❖ Tính căn bậc 2
- ❑ **ceil(Double): Double**
 - ❖ Lấy số nguyên trên
- ❑ **floor(Double): Double**
 - ❖ Lấy số nguyên dưới
- ❑ **str(Object): String**
 - ❖ Chuyển đổi sang chuỗi
- ❑ **cast(T1 AS T2)**
 - ❖ Ép kiểu

```
SELECT
    count(o) AS ItemCount,
    cast(ceil(count(o)/8.0) AS int) AS
    PageCount
FROM Video o
```



DEMOSTATION

- ❑ Trong JPA EntityManager đã cung cấp các phương thức
 - ❖ Merge(Object): cập nhật
 - ❖ Remove(Object): xóa
- ❑ JPQL cũng cho phép UPDATE và DELETE với JPQL hỗ trợ thực hiện cập nhật hoặc xóa một lúc nhiều thực thể.
- ❑ Ví dụ 1:
 - ❖ **UPDATE User o SET o.id='poly', o.name='FPT Polytechnic' WHERE o.id='TeoNV'**
- ❑ Ví dụ 2:
 - ❖ **DELETE FROM User o WHERE o.id IN ('PheoNC', 'NoPT')**

```
String jpql = "DELETE FROM User o WHERE o.id IN :ids";
Query query = em.createQuery(jpql);
query.setParameter("ids", List.of("PheoNC", "NoPT"));
int deletedItemCount = query.executeUpdate();
```

```
String jpql = "UPDATE User o SET o.id=:nid, o.fullname=:name WHERE o.id=:oid";
Query query = em.createQuery(jpql);
query.setParameter("nid", "poly");
query.setParameter("name", "FPT Polytechnic");
query.setParameter("oid", "TeoNV");
int updatedItemCount = query.executeUpdate();
```



DEMOSTATION

- ✓ Giới thiệu JPQL
- ✓ Cấu trúc lệnh JPQL
- ✓ Hàm JPQL





Cảm ơn