Web-based programming with Java

HIBERNATE

ThS. Nguyễn Nghiệm
0913.745.789 - NghiemN@fpt.edu.vn

Mục tiêu

- Hiểu Hibernate framework
- Ánh xạ thực thể với Annotation
- Sử dụng Hibernate API để lập trình CSDL
- Sử dụng thực thể kết hợp

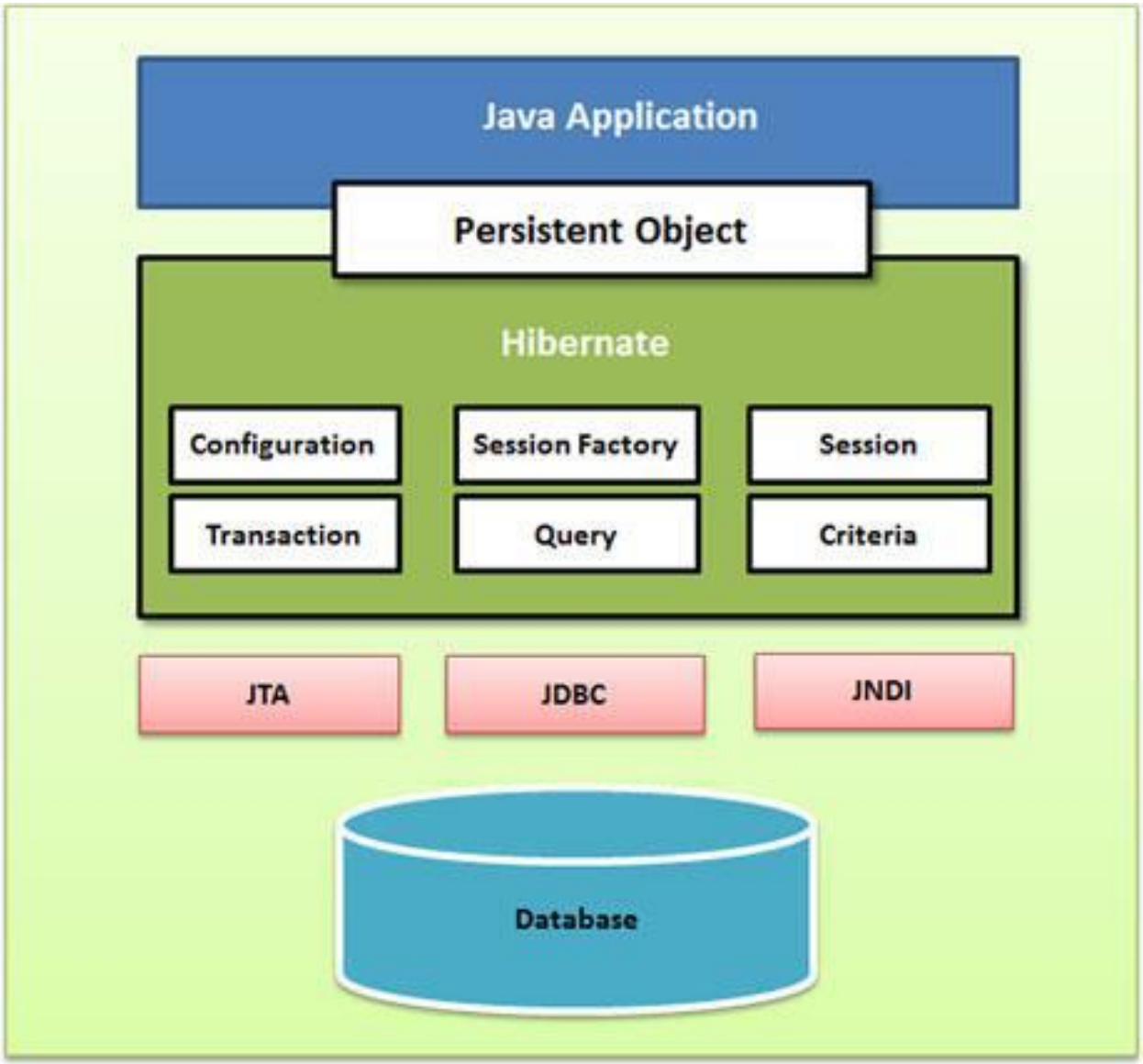
GIỚI THIỆU HIBERNATE

- Hibernate là framework lập trình cơ sở dữ liệu trong java phổ biến nhất hiện nay.
- Cho phép ánh xạ các lớp thực thể với các bảng của CSDL quan hệ bằng XML hoặc Annotation

Lợi ích của Hibernate

- Giảm công việc lập trình, tập trung vào xử lý nghiệp
- Không phụ thuộc CSDL
- Dễ dàng truy vấn các thực thể kết hợp
- Cung cấp các dịch vụ nền hỗ trợ quản lý giao dịch với CSDL ổn định, an toàn và hiệu quả hơn
- Hỗ trợ hầu hết các loại CSDL hiện nay

CÁC THÀNH PHẦN HIBERNATE



THƯ VIỆN HIBERNATE

CORE

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.12.Final</version>
</dependency>
```

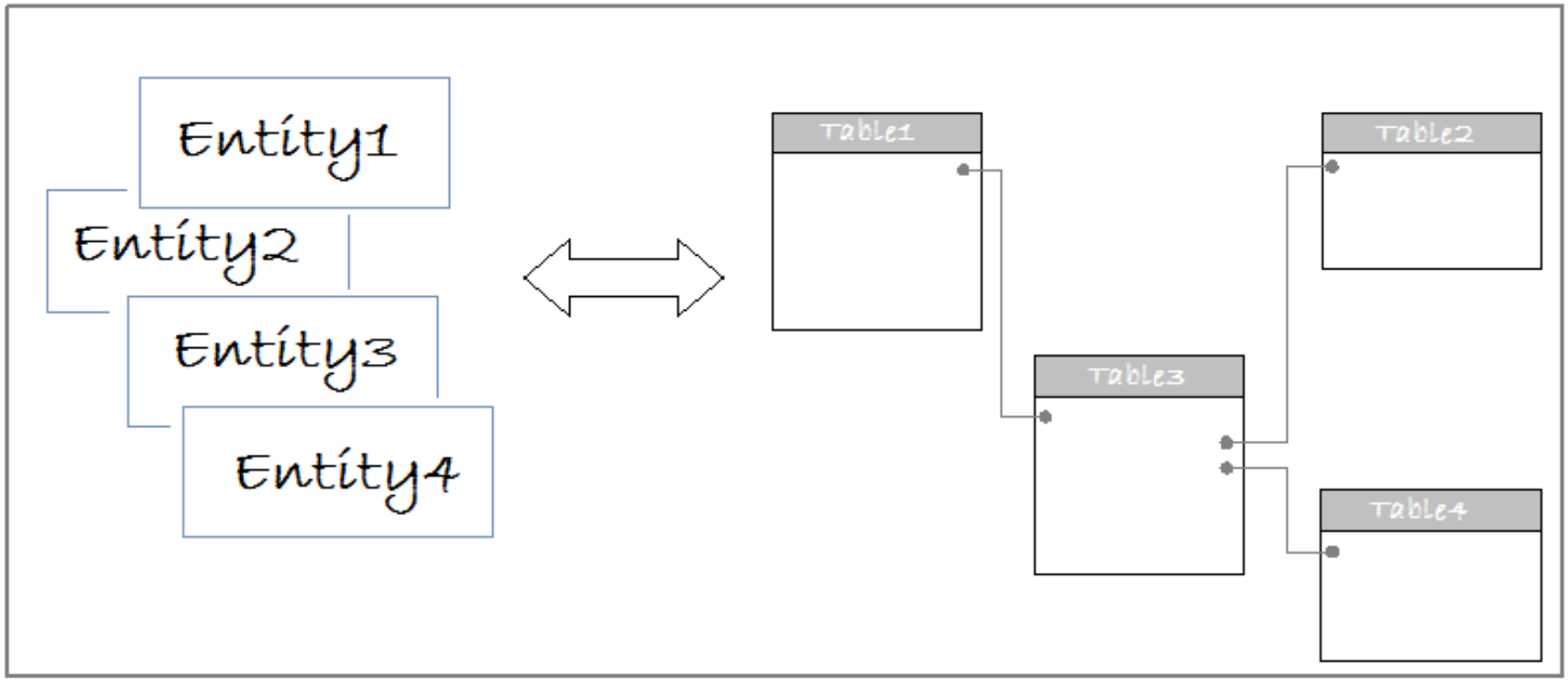
POOL

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-c3p0</artifactId>
    <version>5.2.12.Final</version>
</dependency>
```

VALIDATION

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.5.Final</version>
</dependency>
```

ENTITY MAPPING

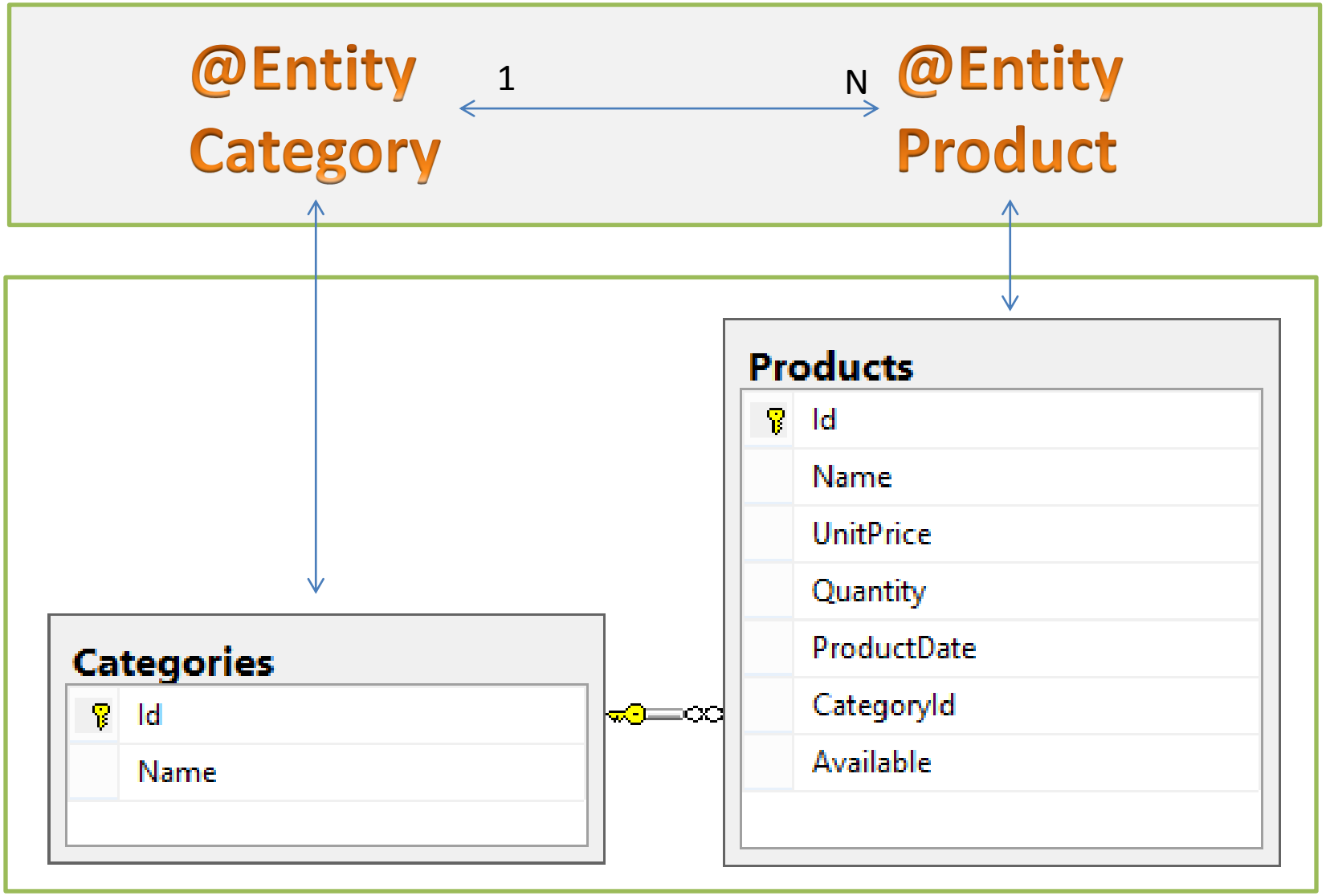


- Mô tả sự liên kết giữa các lớp thực thể và các bảng trong CSDL quan hệ.

ENTITY MAPPING

- Ánh xạ là mô tả liên kết giữa
  - Entity và Table
  - Field và Column
  - Association & Relationship
- Entity phải thỏa qui ước JavaBean
  - Public class
  - Constructor mặc định
  - Getters/Setters

Simple Mapping



CATEGORY MAPPING

```
@Entity
@Table(name="Categories")
public class Category {
    @Id
    @Column(name="Id")
    String id;
    @Column(name="Name")
    String name;

    public String getId() {
    public void setId(String id) {

    public String getName() {
    public void setName(String name) {
}
```

- @Entity
- @Table
- @Column
- @Id

MAPPING DATATYPE

Java type	ANSI SQL Type
int or java.lang.Integer	INTEGER
long or java.lang.Long	BIGINT
short or java.lang.Short	SMALLINT
double or java.lang.Double	FLOAT
java.math.BigDecimal	NUMERIC
char or java.lang.Character	CHAR(1)
java.lang.String	VARCHAR
byte or java.lang.Byte	TINYINT
boolean or java.lang.Boolean	BIT
byte[]	VARBINARY (or BLOB)

MAPPING CONVENTION

- ~~@Table~~
  - Nếu Entity và Table cùng tên
- ~~@Column()~~
  - Nếu Field và Column cùng tên
- @Temporal
  - Chọn kiểu thời gian
- @GeneratedValue
  - Tự tăng

```
@Entity
@Table(name="Products")
public class Product {
    @Id
    @GeneratedValue
    Integer id;
    String name;
    Double unitPrice;
    @Temporal(TemporalType.DATE)
    Date productDate;
    Integer quantity;
    Boolean available;
    String categoryId;
```

RELATIONSHIP MAPPING

```java
@Entity
@Table(name="Products")
public class Product {
    @Id
    @GeneratedValue
    Integer id;
    String name;
    Double unitPrice;
    @Temporal(TemporalType.DATE)
    Date productDate;
    Integer quantity;
    Boolean available;

    @ManyToOne
    @JoinColumn(name="CategoryId")
    Category category;
```

@OneToMany

```java
@Entity
@Table(name="Categories")
public class Category {
    @Id
    String id;
    String name;

    @OneToMany(mappedBy="category")
    List<Product> products;
```

@ManyToOne
@JoinColumn

ENTITY MAPPING ANNOTATIONS

Annotation	Mô tả	Ví dụ
@Entity	Chỉ ra class là thực thể	@Entity @Table(name="Courses") public class Course{...}
@Table	Ánh xạ lớp thực thể với bảng	
@Id	Chỉ ra cột khóa chính	@Id @GeneratedValue int id;
@GeneratedValue	Chỉ ra cột tự tăng	
@Column	Ánh xạ thuộc tính với cột	@Column(name = "Name") String name
@Temporal	Chỉ ra kiểu dữ liệu chuyển đổi	@Temporal(TemporalType.DATE) Date startDate
@ManyToOne	Ánh xạ quan hệ nhiều-1	@ManyToOne @JoinColumn(name="CategoryId") Category category;
@JoinColumn	Chỉ ra cột kết nối	
@OneToMany	Ánh xạ quan hệ 1-nhiều	@OneToMany(mappedBy="category") Collection<Product> products;

HIBERNATE.CFG.XML

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
        com.microsoft.sqlserver.jdbc.SQLServerDriver
    </property>
    <property name="hibernate.connection.url">
        jdbc:sqlserver://localhost;database=EShopV00
    </property>
    <property name="hibernate.connection.username">sa</property>
    <property name="hibernate.connection.password">***</property>
    <property name="hibernate.dialect">
        org.hibernate.dialect.SQLServer2012Dialect
    </property>

    <mapping class="com.entity.Category"/>
    <mapping class="com.entity.Product"/>
  </session-factory>
</hibernate-configuration>
```

HIBERNATE.CFG.XML

@name	Mô tả
hibernate.connection.driver_class	JDBC driver
hibernate.connection.url	Chuỗi kết nối đến CSDL
hibernate.connection.username	Mã đăng nhập CSDL
connection.password	Mật khẩu đăng nhập CSDL
connection.pool_size	Số kết nối tối đa trong hồ
show_sql	Xuất câu lệnh SQL khi truy vấn hoặc thao tác dữ liệu
hbm2ddl.auto	Chỉ ra cơ chế tự định nghĩa CSDL, thường dùng ✓ create-drop: tự tạo CSDL ✓ none: làm việc với CSDL đã tồn tại
hibernate.dialect	Chỉ ra phương pháp làm việc với CSDL của Hibernate (xem bảng bên dưới để khai báo cho đúng)

TIỆN ÍCH HIBERNATEUTIL

```
public class HibernateUtil {
	private static StandardServiceRegistry registry;
	private static SessionFactory sessionFactory;

	public static SessionFactory getSessionFactory() {
		if (sessionFactory == null) {
			try {
				registry = new StandardServiceRegistryBuilder()
					.configure("hibernate.cfg.xml").build();
				MetadataSources sources = new MetadataSources(registry);
				Metadata metadata = sources.getMetadataBuilder().build();
				sessionFactory = metadata.getSessionFactoryBuilder().build();
			} catch (Exception e) {
				shutdown();
			}
		}
		return sessionFactory;
	}
	public static void shutdown() {
		if (registry != null) {
			StandardServiceRegistryBuilder.destroy(registry);
		}
	}
}
```

LẬP TRÌNH THAO TÁC THỰC THỂ

```
Session session = HibernateUtil.getSessionFactory().openSession();

session.beginTransaction().begin();
try {
        session.save(entity);
        session.beginTransaction().commit();
}
catch (Exception e) {
        session.beginTransaction().rollback();
}
```

- save()/persist()
- update()/merge()
- saveOrUpdate()
- delete()/remove()

TRUY VẤN

```
Session session = HibernateUtil.getSessionFactory().openSession();

String sql = "FROM Product";
TypedQuery<Product> query = session.createQuery(sql, Product.class);
List<Product> list = query.getResultList();
```

getResultList()

- FROM Product ~ SELECT p FROM Product p
  - => List<Product>
- SELECT p.name FROM Product p
  - => List<String>
- SELECT p.unitPrice FROM Product p
  - => List<Double>
- SELECT p.name, p.unitPrice FROM Product p
  - => List<Object[]>

HIBERNATE QUERY LANGUAGE

- HQL tương tự SQL nhưng được sử dụng để truy vấn đối tượng, nghĩa là các thành phần bên trong HQL là Entity và Property thay vì Table và Column.
- Ví dụ
  - FROM **Category** WHERE **name** LIKE '%on%'
  - FROM **Product** ORDER BY **unitPrice** DESC
  - SELECT **p.category** FROM **Product** p WHERE **p.id**=1028
  - SELECT **p.category.id**, count(p), sum(**p.unitPrice***p.quantity), min(**unitPrice**) FROM **Product** p GROUP BY **p.category.id**

VÍ DỤ: THAM SỐ VÀ PHÂN TRANG

```
String sql = "FROM Product WHERE unitPrice BETWEEN :min AND :max";
TypedQuery<Product> query = session.createQuery(sql, Product.class);
query.setParameter("min", 5.0);
query.setParameter("max", 15.0);
query.setFirstResult(5);
query.setMaxResults(10);

List<Product> list = query.getResultList();
```

- setParameter(int, Object)
- setParameter(String, Object)
- setFirstResult(int)
- setMaxResults(int)

TRUY VẤN MỘT GIÁ TRỊ

```
String sql = "SELECT avg(unitPrice) FROM Product";
TypedQuery<Double> query = session.createQuery(sql, Double.class);
Double average = query.getSingleResult();
```

getSingleResult()

- SELECT **min(p.unitPrice)** FROM Product p
  - => Double
- FROM **Product p** WHERE p.id=1028
  - => Product
- SELECT **p.category** FROM Product p WHERE p.id=1028
  - => Category

TRUY VẤN MỘT THỰC THỂ

```
Product product = session.get(Product.class, 1028);

Product product = session.find(Product.class, 1028);

Product product = session.load(Product.class, 1028);

Product product = new Product();
product.setId(1028);
session.refresh(product);
```

- 4 cách truy vấn sản phẩm theo khóa chính có giá trị là 1028

TRUY VẤN THỰC THỂ KẾT HỢP

- Kết hợp N-1

Truy vấn Category kết hợp với Product có id là 1005

```
Product entity = (Product) session.get(Product.class, 1005);
Category category = entity.getCategory();
```

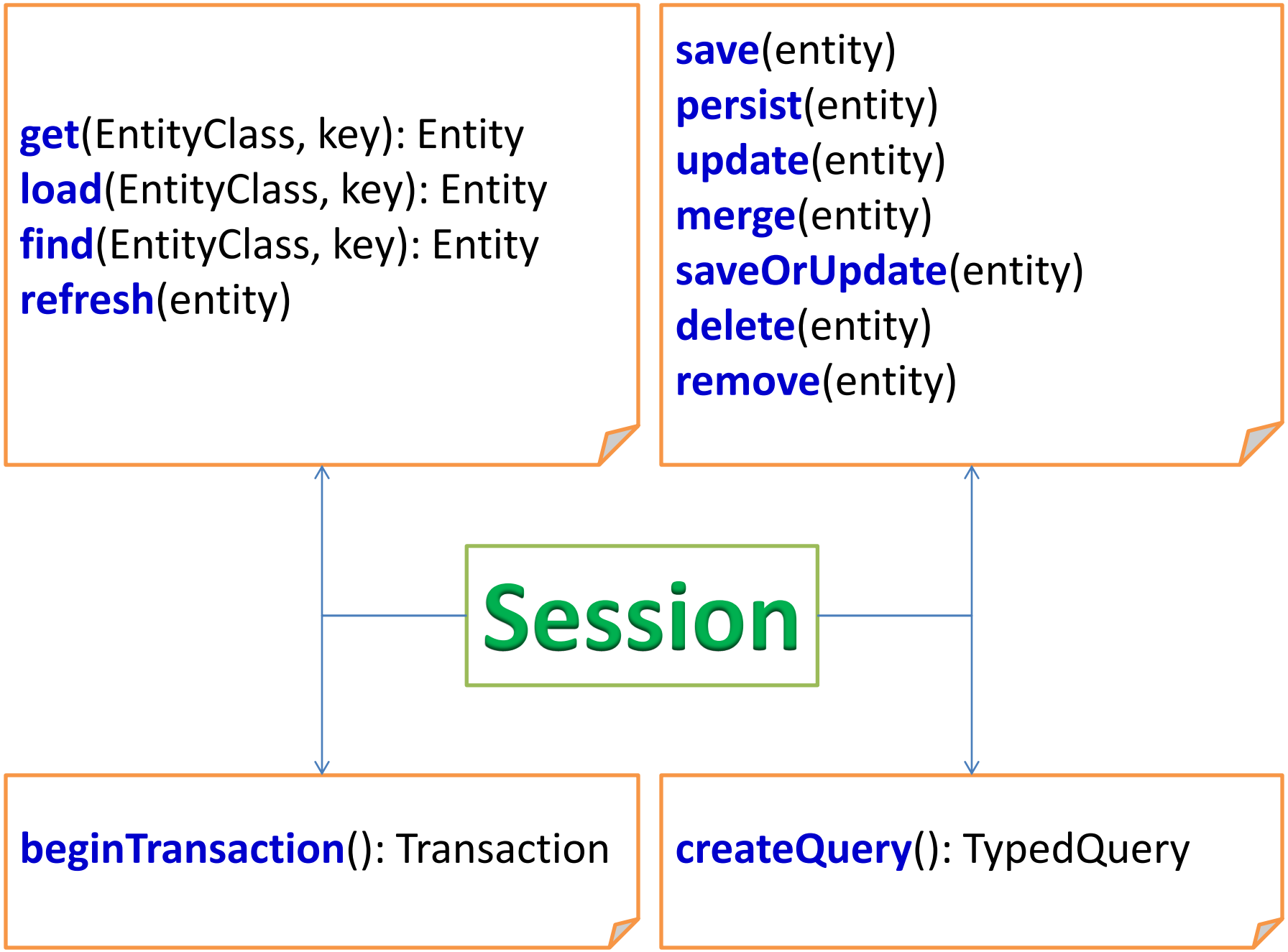
- Kết hợp 1-N

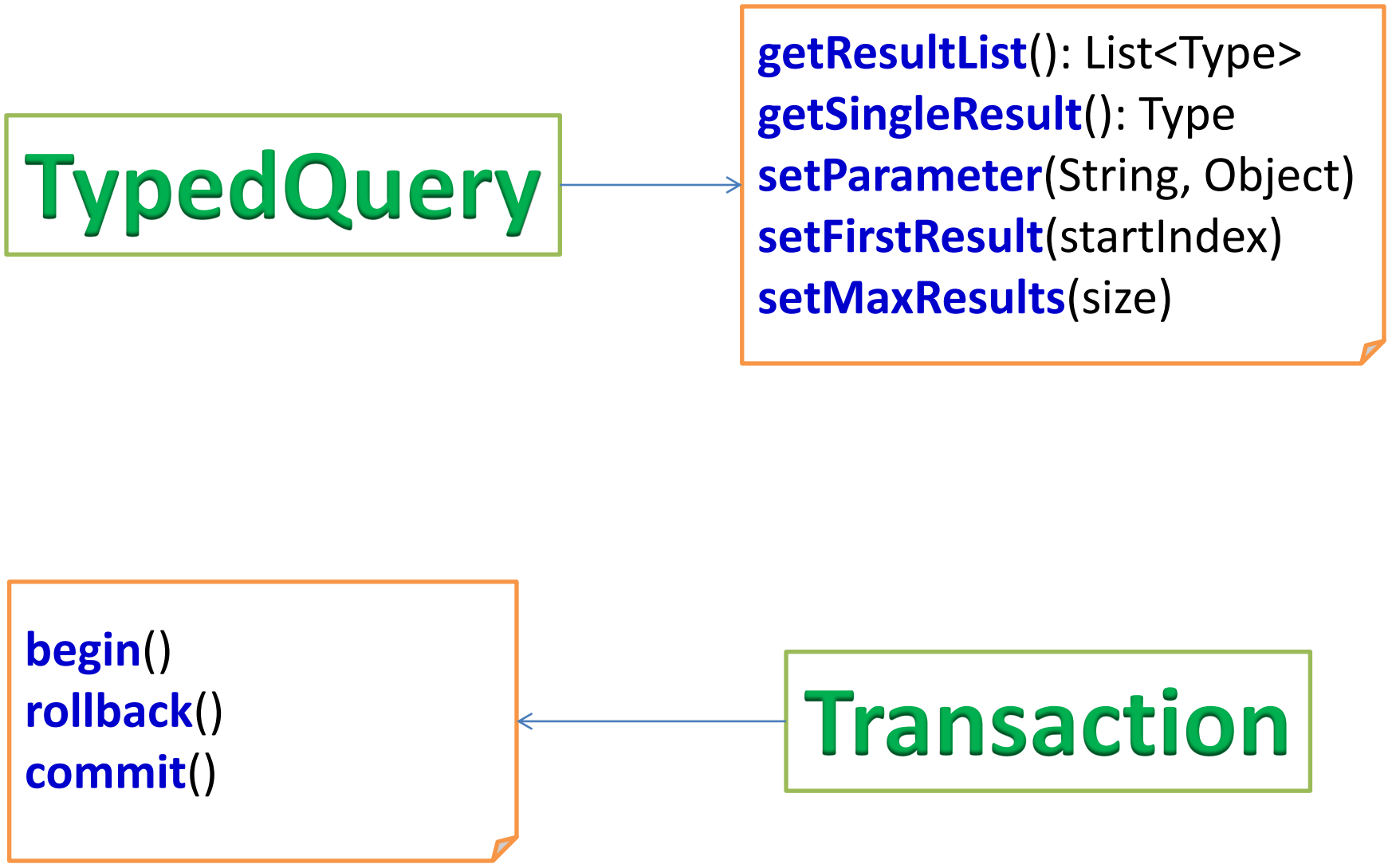
Truy vấn Products kết hợp với Category có id là MOB

```
Category entity = session.get(Category.class, "MOB");
List<Product> list = entity.getProducts();
```

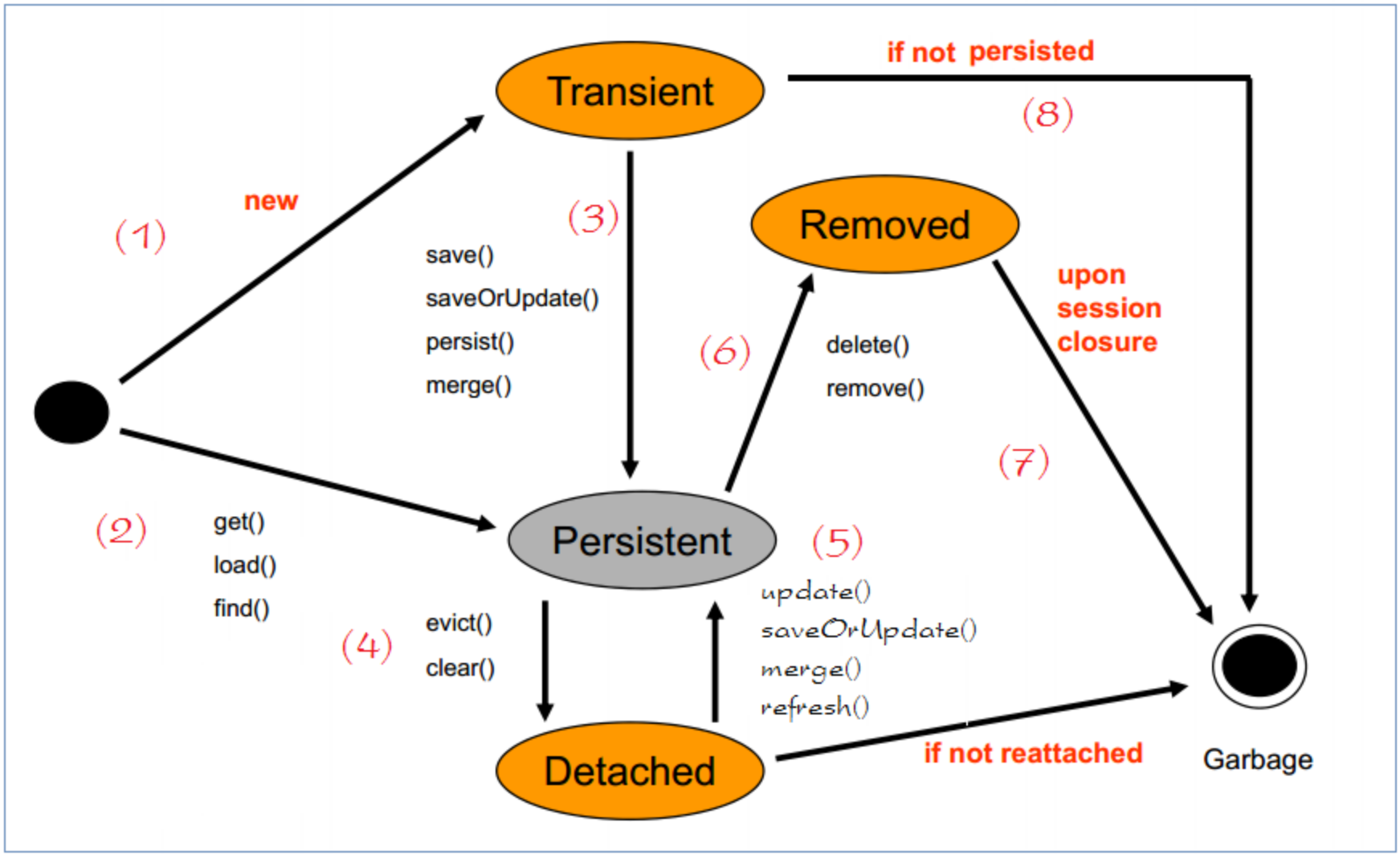
LỢI ÍCH CỦA THỰC THỂ KẾT HỢP

- Thực thể kết hợp là thực thể có liên quan đến thực thể hiện tại.
- Kết hợp @ManyToOne để có được nhiều thông tin hơn về Category thay vì chỉ có CategoryId
- Kết hợp @OneToMany để có ngay các sản phẩm của loại


get(EntityClass, key): Entity
load(EntityClass, key): Entity
find(EntityClass, key): Entity
refresh(entity)
save(entity)
persist(entity)
update(entity)
merge(entity)
saveOrUpdate(entity)
delete(entity)
remove(entity)
Session
beginTransaction(): Transaction
createQuery(): TypedQuery


TypedQuery
getResultList(): List<Type>
getSingleResult(): Type
setParameter(String, Object)
setFirstResult(startIndex)
setMaxResults(size)
begin()
rollback()
commit()
Transaction

ENTITY LIFECYCLE



NGÔN NGỮ HQL

- SELECT
  - <Properties>
- FROM
  - <Entity>
- WHERE
  - <Filter Condition>
- GROUP BY
  - <Group Expression>
- ORDER BY
  - <Order Expression> DESC|ASC

TOÁN TỬ

- ✓ Số học
- ✓ So sánh
- ✓ Quan hệ

TOÁN TỬ ĐẶC BIỆT

- ✓ NOT
- ✓ IN,
- ✓ BETWEEN
- ✓ IS NULL
- ✓ LIKE
- ✓ IS EMPTY

NGÔN NGỮ HQL

- FROM Product
  WHERE name **LIKE** ‘on%’

- FROM Product
  WHERE unitPrice **BETWEEN** 5 **AND** 10
  ORDER BY unitPrice DESC

- FROM Product
  WHERE name **IS NOT NULL**

- FROM Product p
  WHERE p.category.id **IN**(1, 3, 5, 7)

- FROM Category c
  WHERE c.products **IS EMPTY**

NGÔN NGỮ HQL

- SELECT

```
    p.category.id, avg(unitPrice)
FROM Product p
GROUP BY p.category.id
```

- SELECT

```
    p.category.id, avg(unitPrice)
FROM Product p
GROUP BY p.category.id
    HAVING avg(unitPrice) > 100
```

- ✓ Sum()
- ✓ Count()
- ✓ Min()
- ✓ Max()
- ✓ Avg()

HQL Functions

- Tổng hợp số liệu
  - Sum(), Count(), Min(), Max(), Avg(), Size()
- Xử lý chuỗi
  - Length(), Trim(), Lower(), Upper(), Substring(), Concat(), Locate()
- Xử lý thời gian
  - Year(), Month(), Day(), Hour(), Minute(), Second(), Current_Date(), Current_Time(), Current_Timestamp()
- Các hàm khác
  - Sqrt(), str(), cast()

NGÔN NGỮ HQL

- SELECT upper(c.name), size(c.products)
  FROM Category c

- FROM Product
  WHERE locate(name, 'on') >= 0

- FROM Product
  WHERE year(current_date())-year(productDate) > 5

- SELECT concat(name, str(unitPrice))
  FROM Product

TÓM TẮT

- Giới Thiệu
- Ánh xạ
  - Tập tin cấu hình
  - Xây dựng lớp thực thể
- Hibernate API
  - Truy vấn
    - Truy vấn thực thể
    - Truy vấn một thực thể theo khóa chính
  - Thao tác dữ liệu
    - Thêm mới
    - Cập nhật
    - Xóa
- Ngôn ngữ HQL