



HIBERNATE

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



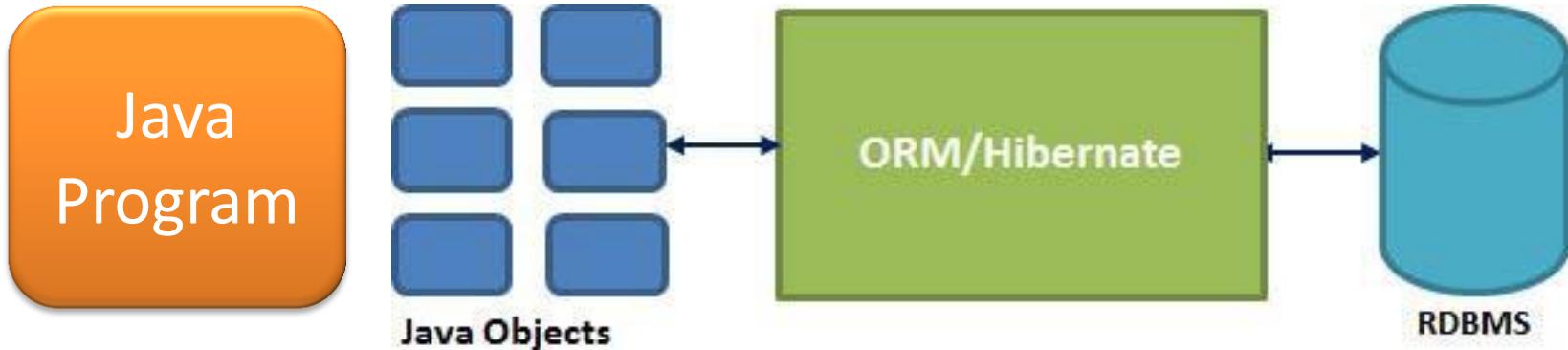
MỤC TIÊU

- Mô tả được cơ chế hoạt động của Hibernate framework
- Khai báo được thư viện phụ thuộc
- Ánh xạ thực thể với Annotation
- Cấu hình được CSDL
- Lập trình thao tác được với CSDL
- Lập trình truy vấn được các thực thể
- Khai thác được thực thể kết hợp
- Sử dụng được câu lệnh HQL

TỔNG QUAN VỀ HIBERNATE



HIBERNATE OVERVIEW



- 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

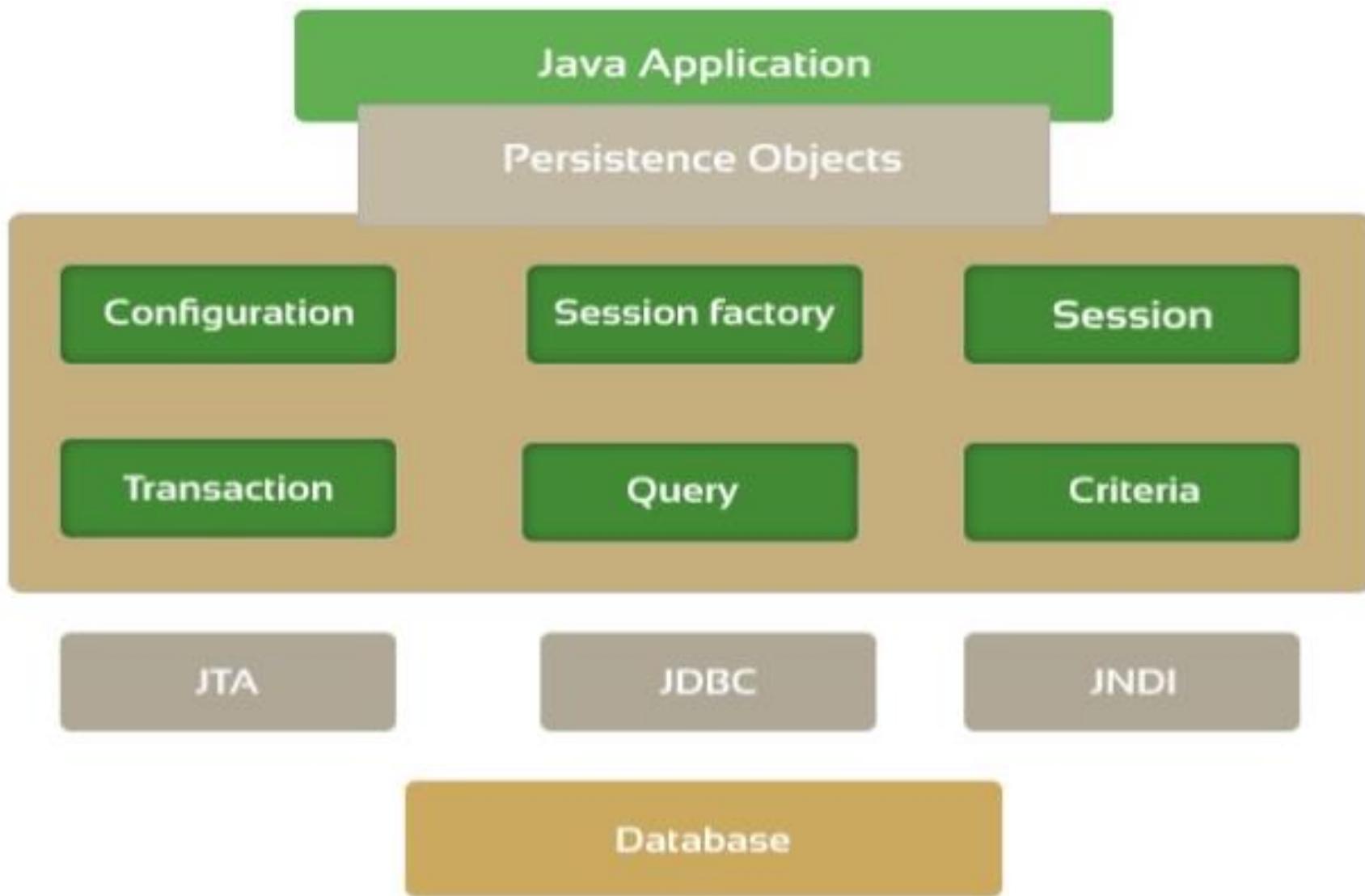


LỢI ÍCH CỦA HIBERNATE

- 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



HIBERNATE ARCHITETURE



CẤU HÌNH VÀ ÁNH XẠ THỰC THỂ



THƯ VIỆN PHỤ THUỘC

CORE

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

POOL

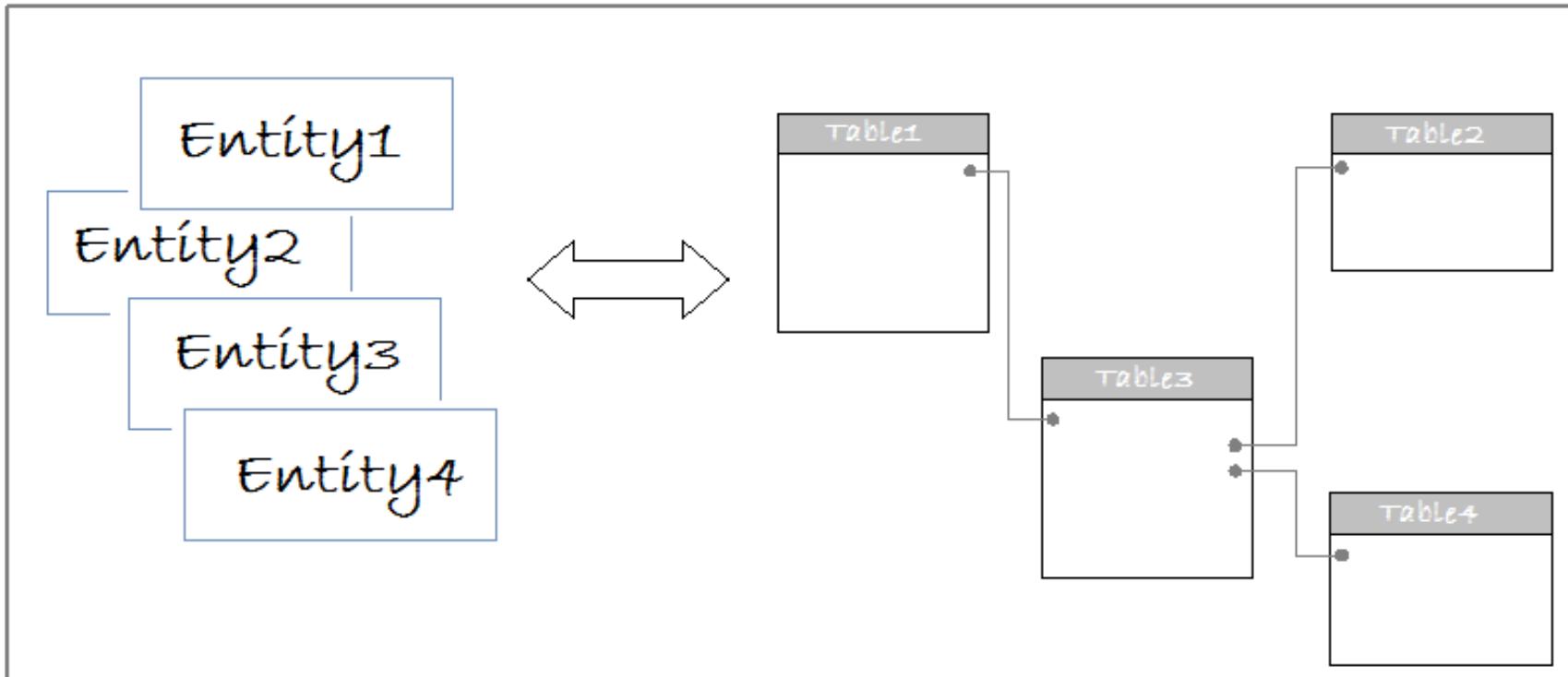
```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-c3p0</artifactId>
    <version>5.2.12.Final</version>
</dependency>
```

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

@Entity
Category \longleftrightarrow **@Entity**
Product

Categories

	Id
	Name

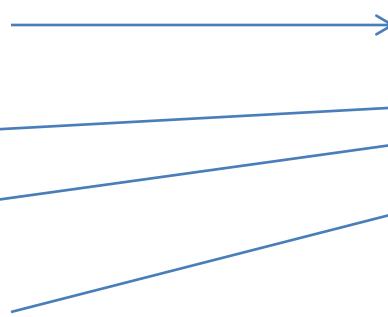
Products

	Id
	Name
	UnitPrice
	Quantity
	ProductDate
	CategoryId
	Available



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

```
@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

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



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 <ul style="list-style-type: none">✓ 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)

LẬP TRÌNH HIBERNATE



TIỆN ÍCH HIBERNATE UTIL

```
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();
Transaction tran = session.beginTransaction();
try {
    session.save(entity);
    tran.commit();
}
catch (Exception e) {
    tran.rollback();
}
HibernateUtil.shutdown();
```

☞ **save()/persist()**

☞ **update()/merge()**

☞ **saveOrUpdate()**

☞ **delete()/remove()**



LẬP TRÌNH TRUY VĂN

List<Category>

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

Double

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

List<Object[]>

```
String hql = "SELECT name, unitPrice FROM Product";  
TypedQuery<Object[]> query = session.createQuery(hql, Object[].class);  
List<Object[]> list = query.getResultList();
```



● TypedQuery.getResultList()

- ★ SELECT p FROM Product p ~ FROM Product
 - => 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[]>

● TypedQuery.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



HQL – HIBERNATE QUERY LANGUAGE

- HQL tương tự SQL nhưng được sử dụng để truy vấn **đối tượng**

- ★ Entity thay vì ~~Table~~

- ★ Property (getters/setters) thay vì ~~Column~~

- Ví dụ

- ★ SELECT **p.category** FROM **Product** p WHERE **p.id**=1028

- *p.category* là *p.getCategory()*

- *Product* là *Entity Class*

- *p.id* là *p.getId()*



HQL EXAMPLES

FROM Category WHERE name LIKE '%on%' ← **List<Category>**

FROM Product ORDER BY unitPrice DESC ← **List<Product>**

SELECT p.category FROM Product p WHERE p.id=1028

SELECT p.category.id, ← **List<Category>**
 count(p), sum(p.unitPrice*p.quantity), min(unitPrice)

FROM Product p

GROUP BY p.category.id ← **List<Object[]>**



PARAMETERS AND PAGINATION

```
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);
```

← **Parametters**

← **Pagination**

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

- **setParameter(int, Object)**
 - ★ Cung cấp giá trị tham số theo vị trí (based zero): 0?, 1?...
- **setParameter(String, Object)**
 - ★ Cung cấp giá trị tham số theo tên: :min, :max...
- **setFirstResult(int)**
 - ★ Chỉ ra vị trí bắt đầu
- **setMaxResults(int)**
 - ★ Chỉ ra số lượng tối đa



PARAMETERS AND PAGINATION

Parameters

`setParameter()`

`setXyz()`

Pagination

`setFirstResult()`

`setMaxResults()`



QUERY AN ENTITY

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

```
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);
```



QUERY AN ENTITY

Find()

Load()

Session

Refresh()

Get()



QUERY ASSOCIATED ENTITIES

● 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();
```



ADVANTAGES OF ASSOCIATED ENTITIES

- Thực thể kết hợp là thực thể có liên quan đến thực thể hiện tại.
- Kết hợp **@ManyToOne** thay khóa ngoại 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 API

createQuery(): TypedQuery

beginTransaction(): Transaction

TypedQuery API

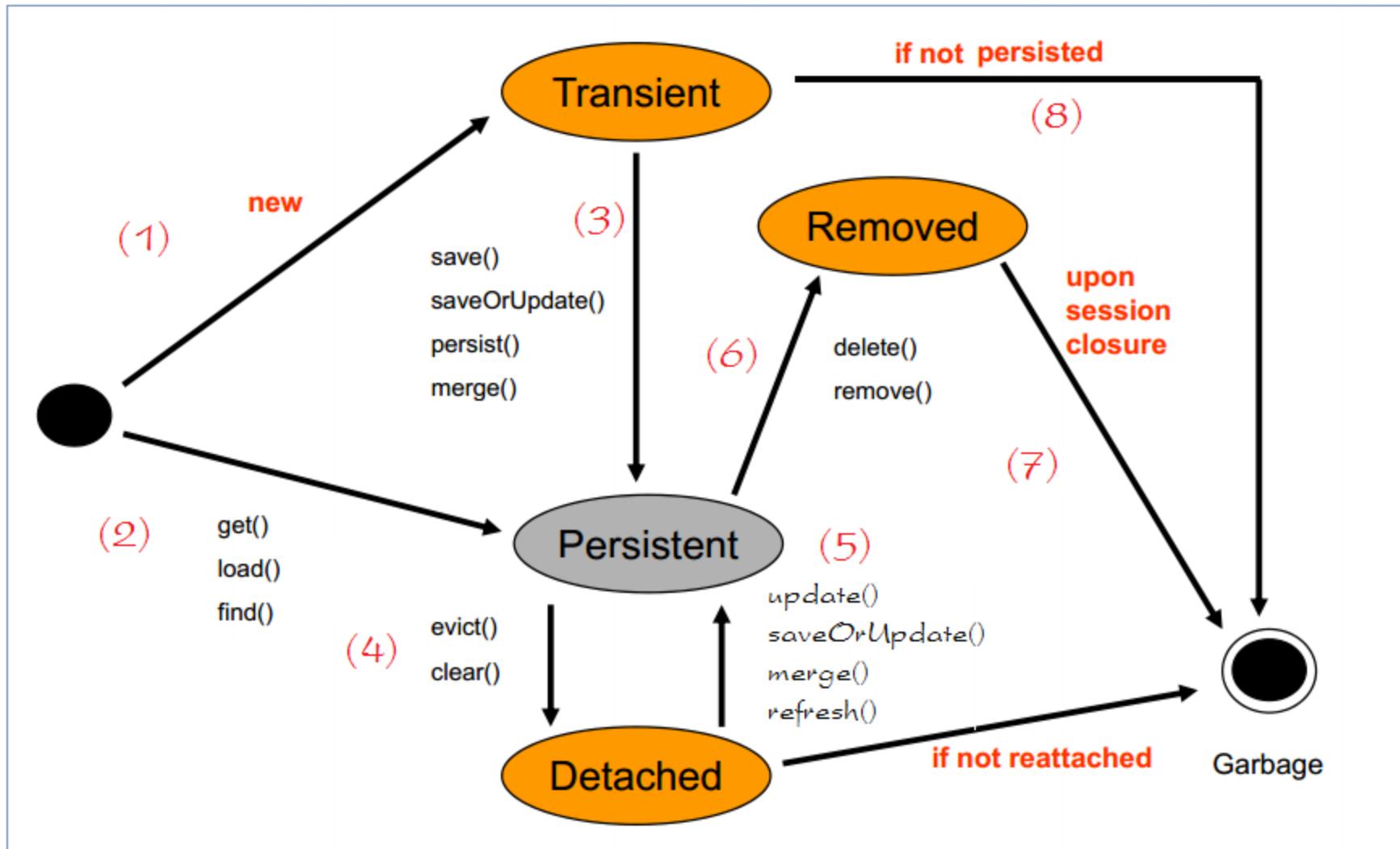
`getResultSet(): List<Type>`
`getSingleResult(): Type`
`setParameter(String, Object)`
`setFirstResult(startIndex)`
`setMaxResults(size)`

`begin()`
`rollback()`
`commit()`

Transaction API



ENTITY LIFECYCLE





HQL DETAILS

● 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