[Bài đọc] Spring Data JPA

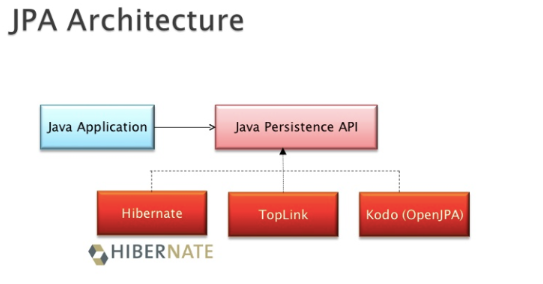
**JPA – Java Persistence API**

JPA là một bộ đặc tả tiêu chuẩn của Java, trong đó mô tả các thao tác quản lý dữ liệu quan hệ trong các ứng dụng sử dụng Java.

JPA bao gồm:

* Bộ API, được định nghĩa trong gói javax.persistence
* Java Persitence Query Language (Ngôn ngữ truy vấn Java Persistence)
* Object/relational metadata

Bởi vì JPA chỉ là một bộ các đặc tả, do đó để sử dụng được thì cần tới một framework triển khai các đặc tả đó, chẳng hạn như là Hibernate.



**Entity**

Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng, chẳng hạn Customer, Product, Student… có thể là các entity.

Entity thường là POJO (Plain Old Java Object) – nghĩa là các lớp đơn giản, chỉ dùng để mô tả dữ liệu chứ không xử lý các logic nghiệp vụ.

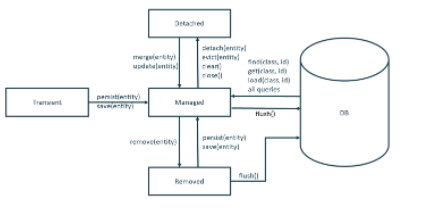
Entity sẽ được ánh xạ (mapping) tới một bảng trong CSDL. Chẳng hạn, lớp Customer có thể ánh xạ tới bảng customers trong CSDL, lớp Student có thể ánh xạ tới bảng students trong CSDL.

Trong JPA, một entity cần tuân thủ:

* Được gắn với annotation javax.persistence.Entity
* Có một constructor là public và không có tham số
* Không được khai báo final
* Các biến đối tượng cần được khai báo là private, protected hoặc ở mức package-private

**Persistence** **Context**

Persistence Context là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu.



*Hình: Vòng đời của Entity*

## ****Entity Manager****

Entity Manager là đối tượng quản lý các entity. Interface EntityManager thực hiện các công việc sau:

* Khai báo các phương thức để tương tác với persistence context
* Tạo hoặc xoá các thể hiện của entity
* Tìm kiếm entity theo khoá chính
* Thực thi các câu lệnh truy vấn lên entity

Có 2 cách để lấy về EntityManager:

Cách 1: Container-managed: Đối tượng EntityManager được quản lý bởi container.Tiêm đối tượng EntityManager sử dụng annotation @PersistenceContext:

@PersistenceContext  
EntityManager em;

Cách 2: Application-managed: Đối tượng EntityManager được quản lý bởi ứng dụng.Tiêm đối tượng EntityManagerFactory sử dụng annotation @PersistenceUnit:

@PersistenceUnit  
EntityManagerFactory emf;

Tạo đối tượng EntityManager thông qua EntityManagerFactory:

EntityManager em = emf.createEntityManager();

### **Tìm entity theo id**

EntityManager cung cấp phương thức find() để tìm một entity theo id. Ví dụ:

    @PersistenceContext

    EntityManager em;

    public void enterOrder(int custID, Order newOrder) {

        Customer cust = em.find(Customer.class, custID);

        cust.getOrders().add(newOrder);

        newOrder.setCustomer(cust);

    }

### **Lưu một entity**

EntityManager cung cấp phương thức persist() để lưu một entity. Ví dụ:

   @PersistenceContext

    EntityManager em;

    public void saveCustomer(Customer customer) {

        em.persist(customer);

    }

### **Cập nhật một entity**

   @PersistenceContext

    EntityManager em;

    public void updateCustomer(Customer customer) {

        em.merge(customer);

    }

Chúng ta có thể đơn giản hoá việc lưu và cập nhật một entity trong cùng một phương thức, ví dụ:

   @PersistenceContext

    EntityManager em;

    public void saveCustomer(Customer customer) {

        if(customer.getId() != null){

            em.merge(customer);

        } else {

            em.persist(customer);

        }

    }

### **Xoá một entity**

EntityManager cung cấp phương thức remove() để xoá một entity. Ví dụ:

    @PersistenceContext

    EntityManager em;

    public void deleteCustomer(Long id) {

        Customer customer = em.find(id);

        if(customer != null){

            em.remove(customer);

        }

    }

### **Câu lệnh truy vấn động**

EntityManager cung cấp phương thức createQuery() để tạo các câu lệnh truy vấn động. Ví dụ:

@Override

**public** List<Product> findAll() {

    TypedQuery<Product> query = **em**.createQuery(**"select p from Product p"**, Product.**class**);

**return** query.getResultList();

}

### **Câu lệnh truy vấn tĩnh**

Trong các lớp entity, chúng ta có thể định nghĩa sẵn các câu lệnh truy vấn. Ví dụ, đoạn mã sau đây định nghĩa một câu lệnh truy vấn có tên là findAllProductWithName để tìm các sản phẩm có tên mong muốn.

@Entity

@Table(name = **"products"**)

@NamedQuery(name=**"findAllProductWithName"**,

        query=**"SELECT c FROM Product c WHERE c.name LIKE :name"**)

**public class** Product {

    @Id

    @GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long **id**;

    @Column

**private** String **name**;

    @Column

**private float price**;

}

Sau đó, chúng ta sử dụng phương thức createNamedQuery của EntityManager để tạo câu truy vấn. Ví dụ:

@PersistenceContext

**private** EntityManager **em**;

@Override

**public** List<Product> findByName(String name) {

    Query query = **em**.createNamedQuery(**"findAllProductWithName"**, Product.**class**);

    query.setParameter(**"name"**, name);

**return** query.getResultList();

}

### **Truyền tham số cho câu lệnh truy vấn**

Có thể truyền tham số cho câu lệnh truy vấn theo tên của tham số đó. Trong ví dụ ở trên, chúng ta truyền giá trị của tham số name theo tên của tham số đó. Tên của các tham số bắt đầu bằng dấu hỏi (?).

Cũng có thể truyền tham số cho câu lệnh truy vấn theo số thứ tự của tham số đó trong câu truy vấn. Vị trí của các tham số bắt đầu bằng dấu hỏi (?). Số thứ tự đầu tiên là 1. Ví dụ:

Query query = **em**.createQuery(

**"SELECT st FROM Student st WHERE st.sroll= ?1 AND st.scourse= ?2"**);

query.setParameter(1, 100);

query.setParameter(2, **"MCA"**);

### **Quản lý transaction**

Chúng ta có thể sử dụng UserTransaction để quản lý các giao dịch. UserTransaction cung cấp các phương thức begin(), commit(), rollback() để quản lý các giao dịch. Ví dụ:

@PersistenceContext

EntityManagerFactory **emf**;

EntityManager **em**;

@Resource  
UserTransaction **utx**;

**void** doSomeTransaction(){

**em** = **emf**.createEntityManager();

**try** {

**utx**.begin();

**em**.persist(SomeEntity);

**em**.merge(AnotherEntity);

**em**.remove(ThirdEntity);

**utx**.commit();

    } **catch** (Exception e) {

**utx**.rollback();

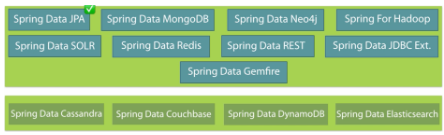
    }

}

Trong đoạn mã trên, phương thức begin() bắt đầu một giao dịch. Phương thức commit() sẽ ghi nhận kết quả giao dịch vào CSDL nếu tất cả các thao tác đều thành công. Nếu có thao tác nào đó không thành công thì phương thức rollback() sẽ được gọi và giao dịch đó sẽ bị huỷ bỏ.

## ****Spring Data JPA****

Spring Data JPA giúp cho việc sử dụng JPA trong ứng dụng Spring trở nên dễ dàng hơn.  Spring Data JPA là một phần của dự án Spring Data. Có thể sử dụng Spring Data JPA với các framework như Hibernate, OpenJPA, EclipseLink…



*Hình: Spring Data*

## Cấu hình Spring Data JPA

Để bắt đầu sử dụng Spring Data JPA, chúng ta cần khai báo các dependency. Ví dụ, chúng ta muốn sử dụng Hibernate, các dependency có thể là:

compile **group**: **'org.springframework.data'**, **name**: **'spring-data-jpa'**, **version**: **'1.11.12.RELEASE'**

compile **group**: **'org.springframework'**, **name**: **'spring-orm'**, **version**: **'4.3.17.RELEASE'**

compile **group**: **'mysql'**, **name**: **'mysql-connector-java'**, **version**: **'8.0.11'**

compile **group**: **'org.hibernate'**, **name**: **'hibernate-core'**, **version**: **'5.3.0.Final'**

compile **group**: **'org.hibernate'**, **name**: **'hibernate-entitymanager'**, **version**: **'5.3.0.Final'**

### **Cấu hình DataSource:**

@Bean

**public** DataSource dataSource(){

    DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

    dataSource.setDriverClassName(**"com.mysql.cj.jdbc.Driver"**);

    dataSource.setUrl(**"jdbc:mysql://localhost:3306/cms"**);

    dataSource.setUsername( **"root"** );

    dataSource.setPassword( **"123456"** );

**return** dataSource;

}

### **Cấu hình EntityManager:**

@Bean

**public** EntityManager entityManager(EntityManagerFactory entityManagerFactory) {

**return** entityManagerFactory.createEntityManager();

}

@Bean

**public** LocalContainerEntityManagerFactoryBean entityManagerFactory() {

    LocalContainerEntityManagerFactoryBean em = **new** LocalContainerEntityManagerFactoryBean();

    em.setDataSource(dataSource());

    em.setPackagesToScan(**new** String[]{**"com.codegym.pms.model"**});

    JpaVendorAdapter vendorAdapter = **new** HibernateJpaVendorAdapter();

    em.setJpaVendorAdapter(vendorAdapter);

    em.setJpaProperties(additionalProperties());

**return** em;

}