**BẢNG ĐÁNH GIÁ ĐIỂM THUYẾT TRÌNH**

**MÔN LẬP TRÌNH JAVA NÂNG CAO**

**LỚP CTK44-PM**

|  |  |
| --- | --- |
| **Thông tin đề tài thuyết trình** | **Thông tin sinh viên thực hiện đánh giá** |
| Tên đề tài: Spring Data JPA cơ bản và nâng cao | MSSV của sinh viên đánh giá: 2012353 |
| Điểm tổng kết: Ngày trình bày: 05/05/2023 | Họ tên sinh viên & ký xác nhận: Đoàn Cao Nhật Hạ |
| ***Thông tin sinh viên thuyết trình*** |
| MSSV: 2011423  Họ tên: Ngô Hoài Phong |

1. **ĐÁNH GIÁ THUYẾT TRÌNH**

**Thang điểm: Kém 0-5; Trung bình: 5-7; Khá 7-8; Tốt 8-9; Xuất sắc: 9-10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bảng chấm điểm nhóm thuyết trình** | | | | | | |
| Các chủ đề chính có được xác định? (A) | Slide thuyết trình có thiết kế tốt (B) | Giao tiếp với người nghe bằng ánh mắt (C) | Giao tiếp với người nghe bằng giọng nói (D) | Khả năng trình bày nội dung (E) | Thảo luận & Hỏi đáp (F) | Điểm trung bình = (A+B+C+D+E+F)/6 |
| **9** | **8** | **9** | **8,5** | **9** | **8,5** | **8,7** |

1. **BÀI TẬP NẮM KIẾN THỨC**

Dựa vào nội dung vừa trình bày, trả lời các câu hỏi sau:

1. Trình bày tóm tắt nội dung anh/ chị đã nắm được về chủ đề vừa trình bày?

* **Mô hình MVC**

+ Model: Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Chẳng hạn như biểu tượng hay là một nhân vật trong game.

+ View: Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.

+ Controller: Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các update tương ứng.

* **JPA**

Spring Boot JPA là một bản ghi chi tiết của Java để quản lý dữ liệu quan hệ trong các ứng dụng Java. Nó cho phép chúng ta truy cập và lưu trữ dữ liệu giữa các object/class Java và database quan hệ. JPA tuân theo Object-Relation Mapping (ORM). Nó là một tập hợp các interface. Nó cũng cung cấp một API EntityManager runtime để xử lý các câu query và giao dịch trên các object dựa trên database. Nó sử dụng ngôn ngữ truy vấn hướng đối tượng độc lập nền tảng JPQL (Java Persistent Query Language).JPA bao gồm ba lĩnh vực:The Java Persistence APIObject-Relational metadataBản thân API, được định nghĩa trong persistence package.JPA không phải là một framework. Nó định nghĩa một khái niệm có thể được thực thi bởi bất kỳ framework nào.

* **Kiến trúc cơ bản của JPA**

Diagram

Description automatically generated

Diagram

Description automatically generated

+ JPA là một nguồn để lưu trữ các entity doanh nghiệp dưới dạng các entity quan hệ. Nó chỉ ra cách định nghĩa POJO là một entity và cách quản lý các entity quan hệ.Hình sau mô tả kiến trúc các cấp độ class của JPA, cùng các class và interface cốt lõi mà được định nghĩa trong package javax persistence. Kiến trúc JPA chứa các đơn vị sau đây:Persistence: là một class chứa các phương thức tĩnh để lấy một instance EntityManagerFactory.EntityManagerFactory: là một class Factory của EntityManager. Nó tạo và quản lý nhiều instance EntityManager.EntityManager: Nó là một giao diện. Nó kiểm soát các hoạt động liên tục trên các đối tượng. Nó hoạt động cho phiên bản Truy vấn.Entity: Các entity là các đối tượng lưu trữ lâu dài như một bản ghi trong database.Persistence Unit: định nghĩa một tập hợp tất cả các class entity. Trong một ứng dụng, các instance EntityManager quản lý nó. Tập hợp các class entity đại diện cho dữ liệu được chứa trong một kho dữ liệu duy nhất.

+ **EntityTransaction**: có mối quan hệ 1:1 với các class EntityManager. Đối với mỗi EntityManager, các hoạt động được duy trì bởi class EntityTransaction.Query (Lệnh truy vấn): là một interface được thực hiện bởi mỗi nhà cung cấp JPA để có được các object quan hệ đáp ứng các tiêu chí.

+ **Mối quan hệ của JPA**: Các class và interface chúng ta đã thảo luận ở trên duy trì một mối quan hệ.

* Mối quan hệ giữa EntityManager và EntiyTransaction là 1:1. Có một instance EntityTransaction cho mỗi hoạt động EntityManager.Mối quan hệ giữa EntityManageFactory và EntiyManager là một-nhiều. Nó là một class factory chứa các instance EntityManager.Mối quan hệ giữa EntityManager và Query là một-nhiều. Chúng ta có thể thực hiện bất kỳ số lượng truy vấn nào bằng cách sử dụng một instance của class EntityManager.Mối quan hệ giữa EntityManager và Entity là một-nhiều. Một instance EntityManager có thể quản lý nhiều entity.
* **So sánh giữa Sping boot và  java servlet :**

+ Giống nhau : Spring Boot và Servlet đều hỗ trợ JPA và có thể được sử dụng để phát triển các ứng dụng web sử dụng JPA.

+ Spring Boot cung cấp một số tính năng hỗ trợ JPA, giúp phát triển ứng dụng dễ dàng hơn. Một số tính năng của Spring Boot bao gồm:

+ Dependency management: Spring Boot có thể quản lý các phụ thuộc (dependencies) của ứng dụng, giúp cho việc cấu hình và triển khai trở nên dễ dàng hơn.

+ Auto-configuration: Spring Boot có thể tự động cấu hình JPA và các thành phần khác của ứng dụng mà không cần phải viết nhiều mã.

+ Spring Data JPA: Spring Boot cung cấp Spring Data JPA, một thư viện hỗ trợ việc truy vấn cơ sở dữ liệu dễ dàng hơn với JPA.

+ Servlet cũng hỗ trợ JPA, tuy nhiên, nó không có các tính năng tương tự như Spring Boot để hỗ trợ JPA. Việc sử dụng Servlet để phát triển ứng dụng sử dụng JPA có thể đòi hỏi nhiều thời gian và công sức hơn so với sử dụng Spring Boot.

* **Các bước sử dụng JPA**

+ Bước 1: Chuẩn bị dependency trong file pom.xml

Text

Description automatically generated

+ Bước 2: Cấu hình connection kết nối database trong file application.properties.

Text

Description automatically generated

+ Bước 3: Bước 3 - Chuẩn bị entity. Mapping table Department trong database thành các class Java.

Text

Description automatically generated

+ Bước 5: Tạo file DepartmentQueryCreationService Service

Service có nhiệm vụ thực hiện các nghiệp vụ của ứng dụng . Đồng thời nhúng bean Repository để gọi tầng Persistence.

Text

Description automatically generated

+ Bước 6 - Tạo file DepartmentAnnotationRepository sử dụng JPA

Tầng này có nhiệm vụ thao tác lấy dữ liệu. Các cách lấy dữ liệu sẽ được giới thiệu riêng ở bài khác.

Text

Description automatically generated

+ Bước 7 - Luồng đi của ứng dụng trên như sau:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* **Object-Relation Mapping (ORM)**

+ Trong ORM, ánh xạ (mapping) các đối tượng Java tới các bảng database và ngược lại được gọi là Object-Relation Mapping. Ánh xạ ORM hoạt động như một cầu nối giữa database quan hệ (bảng và bản ghi) và ứng dụng Java (các class và object).Trong hình sau, layer ORM là một layer chuyển đổi. Nó điều chỉnh ngôn ngữ của đồ thị object sang ngôn ngữ của SQL và các bảng quan hệ.

+ Layer ORM tồn tại giữa ứng dụng và database. Nó chuyển đổi các class và object Java để chúng có thể được lưu trữ và quản lý trong database quan hệ. Theo mặc định, tên class trở thành tên của bảng và các trường trở thành cột. Khi một ứng dụng thiết lập, mỗi hàng trong bảng tương ứng với một object.

* **Tính năng JPA**

+ Có các tính năng sau của JPA:

* Là một repository mạnh mẽ và object-mapping abstraction tùy chỉnh .
* Hỗ trợ cho cross-store persistence. Tức là một entity có thể được lưu trữ một phần trong MySQL và Neo4j (Hệ thống quản lý cơ sở dữ liệu đồ thị).
* Tự động tạo các truy vấn từ ngay chính tên các phương thức truy vấn.
* Các lớp dựa vào tên miền cung cấp các thuộc tính cơ bản.
* Hỗ trợ kiểm tra quản lý minh bạch.
* Khả năng tích hợp mã repository tùy chỉnh.
* Dễ dàng tích hợp với Spring Framework với namespace tùy chỉnh.
* Các kỉ thuật nâng cao trong JPA

**+ Query Creation:** Trong Spring JPA, có một cơ chế giúp chúng ta tạo ra các câu Query mà không cần viết thêm code.

* Cơ chế này xây dựng Query từ tên của method.
* Ví dụ:

Graphical user interface, text, application, email

Description automatically generated

* Khi chúng ta đặt tên method là: findByAtk(int atk)

* Thì Spring JPA sẽ tự định nghĩa câu Query cho method này, bằng cách xử lý tên method. Vậy là chúng ta đã có thể truy vấn dữ liệu mà chỉ mất thêm 1 dòng code.
* Cơ chế xây dựng Query từ tên method này giúp chúng ta tiết kiệm thời gian với những query có logic đơn giản, và cũng đặc biệt hữu ích là nó giống ngôn ngữ con người thường nói hơn là SQL. (human-readable)

**+ Query:** Với cách sử dụng @Query, bạn sẽ có thể sử dụng câu truy vấn JPQL (Hibernate) hoặc raw SQL.

Ví dụ :

public interface UserRepository extends JpaRepository<User, Long> {

    // Khi được gắn @Query, thì tên của method không còn tác dụng nữa

    // Đây là JPQL

    @Query("select u from User u where u.emailAddress = ?1")

    User myCustomQuery(String emailAddress);

    // Đây là Native SQL

    @Query(value = "select \* from User u where u.email\_address = ?1", nativeQuery = true)

    User myCustomQuery2(String emailAddress);

}

* NameQuery: để tạo câu lệnh Select. Để gọi được câu lệnh @NamedQuery(name = “Employee.fetchByLastNameLength”) thì ở JPA Repository, ta phải có phương thức (fetchByLastNameLength) giống y như vậy.

+ Ví dụ

@Entity

@Table(name = "employee", schema="spring\_data\_jpa\_example")

@NamedQuery(name = "Employee.fetchByLastNameLength",

        query = "SELECT e FROM Employee e WHERE CHAR\_LENGTH(e.lastname) =:length "

)

public class Employee {

    @Id

    @Column(name = "id")

    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    private Long id;

    @Column(name = "firstname")

    private String firstName;

    @Column(name = "lastname")

    private String lastname;

 }

@Repository

public interface EmployeeRepository extends JpaRepository<Employee,Long>, EmployeeRepositoryCustom {

    List<Employee> fetchByLastNameLength(@Param("length") Long length);

}

+ Order in Query để sắp xếp dữ liệu theo chiều tăng hoặc giảm dần.

Ví dụ

departmentRepository.findAll(new Sort(Sort.Direction.ASC, "name"));

+  Annotation @Modifying để cập nhật dữ liệu.

@Modifying

@Query("update User u set u.status = :status where u.name = :name")

int updateUserSetStatusForName(@Param("status") Integer status,

  @Param("name") String name);

* Kết quả trả về là số lượng dòng đã được cập nhật trong Database. Chúng ta cũng có thể sử dụng Native Query để cập nhật như sau

@Modifying

@Query(value = "update Users u set u.status = ? where u.name = ?",

  nativeQuery = true)

int updateUserSetStatusForNameNative(Integer status, String name);

+  Annotation @Modifying để chèn dữ liệu.

Ví dụ:

@Modifying

@Query(

  value =

    "insert into Users (name, age, email, status) values (:name, :age, :email, :status)",

  nativeQuery = true)

void insertUser(@Param("name") String name, @Param("age") Integer age,

  @Param("status") Integer status, @Param("email") String email);

+ Pageable để phân trang

Ví dụ

Pageable first10PageWithTwoElements = PageRequest.of(0, 10);

Page<Product> allProducts = productRepository.findAll(first10PageWithTwoElements);

public interface ProductRepository extends PagingAndSortingRepository<Product, Integer> {

    List<Product> findAllByPrice(double price, Pageable pageable);

}

1. Anh/ chị hãy đưa ra các nhận xét và góp ý cho sinh viên vừa trình bày?

* Cần tương tác với người nghe hơn