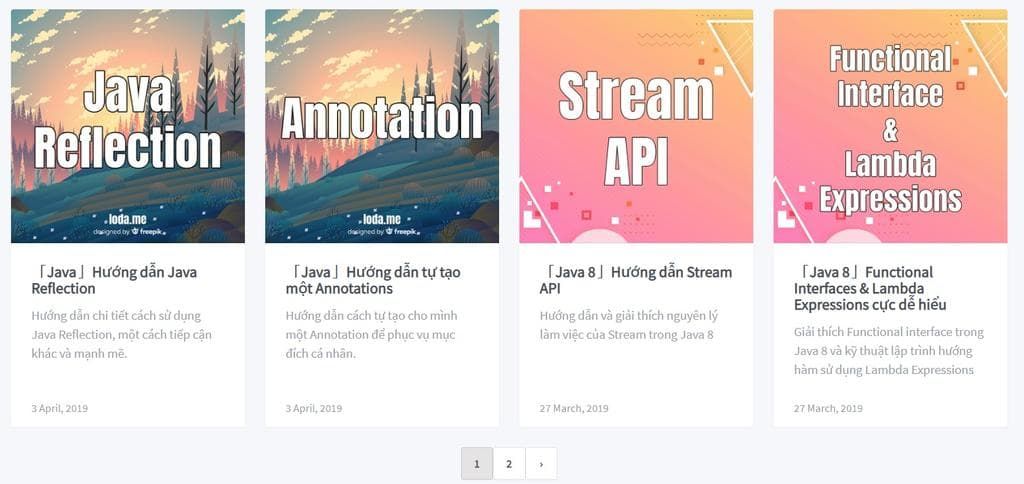
**Giới thiệu**

Điều này có lẽ ai cũng biết, đó là đa phần chúng ta không lấy toàn bộ dữ liệu từ Database lên, mà chỉ lấy một số lượng nhất định, và chia nó thành nhiều trang.

Mà cái blog này cũng là ví dụ điển hình:



Tôi chia dữ liệu thành nhiều trang khác nhau, với mỗi trang thì sẽ lấy ra các bài viết cần thiết.

Tạo sao cần dùng phân trang? vì nó giúp tiết kiệm băng thông và tăng tốc xử lý, ngoài ra nó cũng giảm thiểu việc hiển thị các thông tin thừa không cần thiết.

Chúng ta sẽ tìm hiểu các làm việc này bằng JPA Pageable

Nếu chưa biết Spring JPA thì xem tại đây:

1. [Hibernate là gì?](https://loda.me/hibernate-la-gi-loda1554623701594)
2. [Spring JPA](https://loda.me/spring-boot-11-huong-dan-spring-boot-jpa-my-sql-loda1558687596060)

**Cài đặt**

Nhớ thêm spring-boot-starter-data-jpa vào dependencies của bạn.

Trong phần này, tôi xài H2 database để demo. (h2 là dạng memory database, nó sẽ lưu trong ram và khi tắt chương trình nó sẽ mất sạch)

*pom.xml*

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>me.loda.spring</groupId>

<artifactId>spring-boot-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-learning</name>

<description>Everything about Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--spring mvc, rest-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!--spring jpa-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- h2 database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

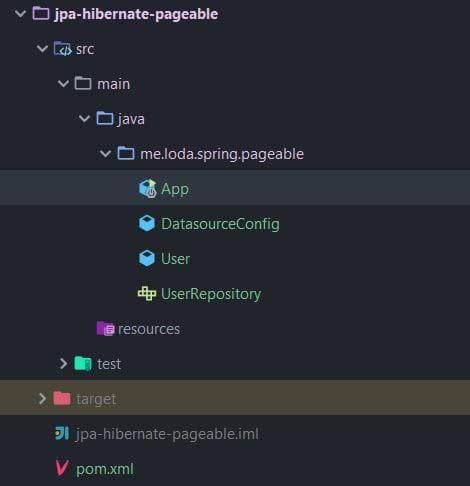
</plugin>

</plugins>

</build>

</project>

Cấu trúc thư mục:



**Tạo Model và Repository**

Tạo ra class User và insert sẵn 100 User vào Database.

*User.java*

@Entity

@Data

@Builder

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

}

*UserRepository.java*

public interface UserRepository extends JpaRepository<User, Long> {

}

*DatasourceConfig.java*

@Configuration

@RequiredArgsConstructor

public class DatasourceConfig {

// inject bởi RequiredArgsConstructor

private final UserRepository userRepository;

// Chỉ áp dụng trong demo :D

@PostConstruct

public void initData() {

// Insert 100 User vào H2 Database sau khi

// DatasourceConfig được khởi tạo

userRepository.saveAll(IntStream.range(0, 100)

.mapToObj(i -> User.builder()

.name("name-" + i)

.build())

.collect(Collectors.toList())

);

}

}

**Pageable**

Ukie, giả sử bây giờ chúng ta có 100 record trong DB rồi.

Để có thể query lấy dữ liệu theo dạng Page, **Spring Data JPA** hỗ trợ chúng ta bằng đối tượng Pageable.

Hàm findAll(Pageable pageable) là có sẵn và trả về đối tượng Page<T>

// Lấy ra 5 user đầu tiên

// PageRequest.of(0,5) tương đương với lấy ra page đầu tiên, và mỗi page sẽ có 5 phần tử

// PageRequest là một đối tượng kế thừa Pageable

Page<User> page = userRepository.findAll(PageRequest.of(0, 5));

Để lấy ra 5 User tiếp theo, chúng ta có thể làm theo 2 cách:

// tận dụng đối tượng Page trước đó

page.nextPageable()

// Sử dụng PageRequest mới

PageRequest.of(1, 5)

**Sorting**

Bạn có thể query dạng Page kèm theo yêu cầu sorting theo một trường nào đó.

Page<User> page = userRepository.findAll(PageRequest.of(1, 5, Sort.by("name").descending()));

**Note**

Ngoài ra, bạn hoàn toàn có thể tự custom để hàm trả về Page<T>, Slice<T>, List<T>.

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findAllByNameLike(String name, Pageable pageable);

}

**Ví dụ DEMO**

@SpringBootApplication

@RequiredArgsConstructor

public class App {

public static void main(String[] args) {

SpringApplication.run(App.class, args);

}

private final UserRepository userRepository;

@Bean

CommandLineRunner run() {

return args -> {

// Lấy ra 5 user đầu tiên

// PageRequest.of(0,5) tương đương với lấy ra page đầu tiên, và mỗi page sẽ có 5 phần tử

Page<User> page = userRepository.findAll(PageRequest.of(0, 5));

// In ra 5 user đầu tiên

System.out.println("In ra 5 user đầu tiên: ");

page.forEach(System.out::println);

// Lấy ra 5 user tiếp theo

page = userRepository.findAll(page.nextPageable());

System.out.println("In ra 5 user tiếp theo: ");

page.forEach(System.out::println);

System.out.println("In ra số lượng user ở page hiện tại: " + page.getSize());

System.out.println("In ra tổng số lượng user: " + page.getTotalElements());

System.out.println("In ra tổng số page: " + page.getTotalPages());

// Lấy ra 5 user ở page 1, sort theo tên

page = userRepository.findAll(PageRequest.of(1, 5, Sort.by("name").descending()));

System.out.println("In ra 5 user page 1, sắp xếp theo name descending:");

page.forEach(System.out::println);

// Custom method

List<User> list = userRepository.findAllByNameLike("name-%", PageRequest.of(0, 5));

System.out.println(list);

};

}

}

Output:

In ra 5 user đầu tiên:

User(id=1, name=name-0)

User(id=2, name=name-1)

User(id=3, name=name-2)

User(id=4, name=name-3)

User(id=5, name=name-4)

In ra 5 user tiếp theo:

User(id=6, name=name-5)

User(id=7, name=name-6)

User(id=8, name=name-7)

User(id=9, name=name-8)

User(id=10, name=name-9)

In ra số lượng user ở page hiện tại: 5

In ra tổng số lượng user: 100

In ra tổng số page: 20

In ra 5 user page 1, sắp xếp theo name descending:

User(id=95, name=name-94)

User(id=94, name=name-93)

User(id=93, name=name-92)

User(id=92, name=name-91)

User(id=91, name=name-90)

Custom Method

[User(id=1, name=name-0), User(id=2, name=name-1), User(id=3, name=name-2), User(id=4, name=name-3), User(id=5, name=name-4)]

**Kết**

Pageable là một cách tiếp cận rất tiện lợi, tuy nhiên bạn cần biết rằng các hàm sử dụng Pageable sẽ thực hiện query 2 lần xuống DB. Một lần là để lấy ra tổng số lượng bản ghi, một lần là query lấy ra page bạn cần. Cái này tôi sẽ nói kĩ hơn ở bài sau để chúng ta tối ưu hiệu năng