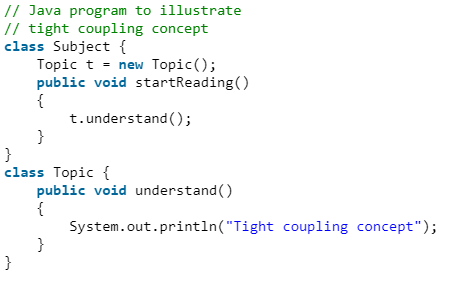
**Báo cáo thu hoạch**

1. **Tight coupled**

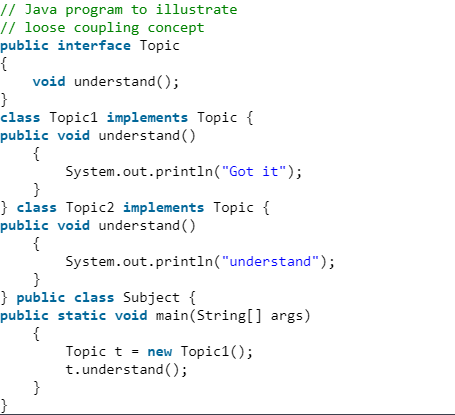
* Tight coupled có nghĩa là sự kết nối chặt chẽ. Trong Java, có thể hiểu là hai lớp thay đổi cùng nghĩa là 1 trong 2 lớp có sự kết nối chặt chẽ khi thay đổi thì lớp còn lại cũng phải thay đổi theo.



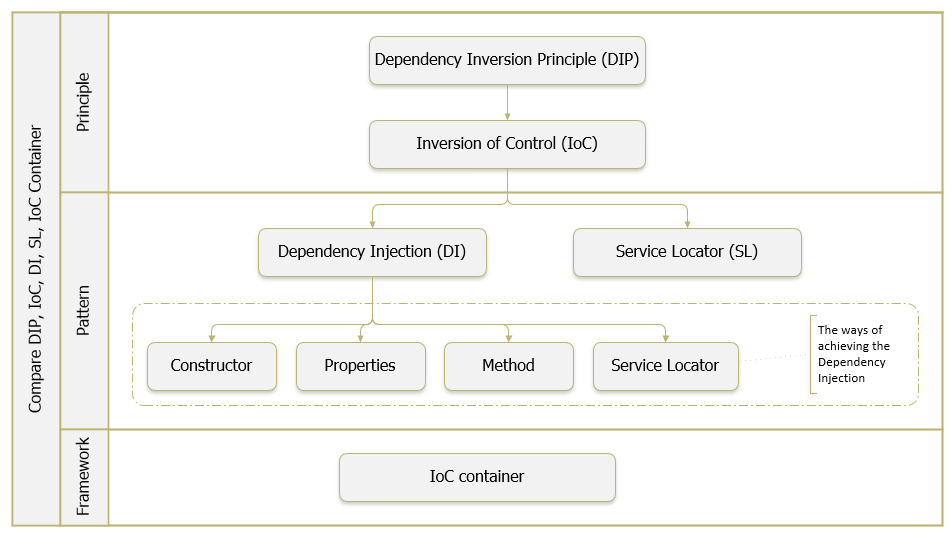
**Giải thích:** Trong chương trình trên, lớp Subject phụ thuộc vào lớp Topic hay lớp Subject được kết hợp chặt chẽ với lớp Topic, điều đó có nghĩa là nếu bất kỳ thay đổi nào trong lớp Topic thì lớp Subject phải thay đổi. Ví dụ, nếu phương thức understand() của lớp Topic thay đổi thành phương thức gotIt() thì Subject phải thay đổi phương thức startReading() sẽ gọi phương thức gotIt () thay vì gọi phương thức understand().

1. **Loosely coupled**

* Hiểu đơn giản, thì loosely coupled là sự kết nối lỏng đối lập với tight coulped và mối quan hệ giữa các đối tượng gần như không có sự phụ thuộc. Để khắc phục các vấn đề tight coupled giữa các đối tượng, Spring sử dụng cơ chế **Dependency Injection** với sự trợ giúp của mô hình POJO / POJI và thông qua tiêm phụ thuộc có thể đạt được khớp nối lỏng.



**Giải thích:** Trong ví dụ trên, các đối tượng Topic1 và Topic2 được ghép nối lỏng lẻo. Nó có nghĩa là Topic là một giao diện và chúng ta có thể đưa vào bất kỳ lớp nào trong số các lớp được triển khai tại thời điểm chạy và chúng tôi có thể cung cấp dịch vụ cho người dùng cuối.

1. **Dependency Injection**
   1. **Khái niệm**

* Dependency Injection (DI): là một Design Pattern, một cách để hiện thực Inversion of Control Pattern. DI chính là khả năng liên kết giữa các thành phần lại với nhau, các module phụ thuộc (dependency) sẽ được inject vào module cấp cao.
* Dependency injection thường có nghĩa là truyền một đối tượng vào phương thức phụ thuộc, như một tham số cho phương thức, chứ không phải để phương thức tạo đối tượng phụ thuộc. Điều đó có nghĩa là trong thực tế, phương thức này không phụ thuộc trực tiếp vào một implementation cụ thể; bất kỳ implementation nào cũng đáp ứng yêu cầu đều có thể được chuyển dưới dạng tham số.
* Ví dụ:
  + **Khi không sử dụng Dependency Injection**

Hãy xem xét ví dụ sau, khi mà chúng ta tạo trực tiếp đối tượng address của lớp Address bên trong lớp Student. Rõ ràng là đối tượng address sẽ bị phụ thuộc vào Student. Mỗi khi muốn thay đổi đối tượng này thì phải thay đổi đối tượng student.

public class Student {

private Address address;

public Student() {

address = new Address();

}

}

* **Sử dụng Dependency Injection**

Để đảo ngược tình thế trong ví dụ trên, nghĩa là làm cho student và address giảm phụ thuôc vào nhau, chúng ta có thể khởi tạo đối tượng address ở bên ngoài lớp Student và truyền vào thông qua Constructor hoặc phương thức Setter.

* + Thông qua Constructor

public class Student {

private Address address;

// constructor được sử dụng để truyền dependency

public Student(Address address) {

this.address = address;

}

}

* + Thông qua phương thức Setter

public class Student {

private Address address;

// phương thức setter được sử dụng để truyền dependency

public void setAddress(Address address) {

this.address = address;

}

// phương thức getter trả về đối tượng address

public Address getAddress() {

return address;

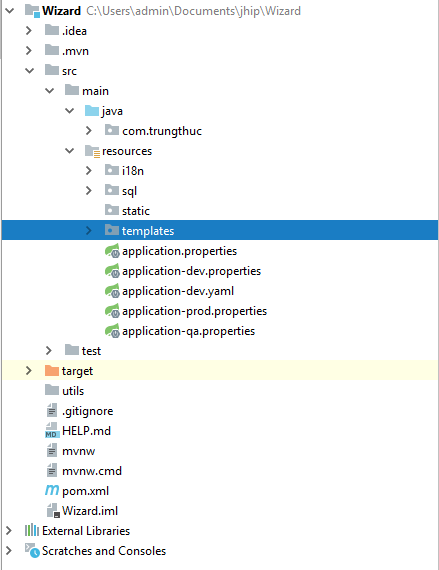
}

* 1. **Các dạng Dependency Injection**
* Constructor Injection: Các dependency sẽ được container truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất.
* Setter Injection: Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter.
* Interface Injection: Class cần inject sẽ implement 1 interface. Interface này chứa 1 hàm tên Inject. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm Inject của interface đó.
  1. **Ưu điểm và khuyết điểm của Dependency Injection**
  2. **Ưu điểm:**
* **Reduced dependencies**: giảm sự kết dính giữa các module.
* **Reusable:** code dễ bảo trì, dễ tái sử dụng, thay thế module. Giảm boiler-plate code do việc tạo các biến phụ thuộc đã được injector thực hiện.
* **Testable**: rất dễ test và viết Unit Test.
* **Readable**: dễ dàng thấy quan hệ giữa các module vì các dependecy đều được inject chuyển đổi thành tham số.
  1. **Khuyết điểm:**
* Khái niệm DI khá khó hiểu đối với người mới tìm hiểu.
* Sử dụng interface nên đôi khi sẽ khó debug, do không biết chính xác module nào được gọi.
* Các object được khởi tạo toàn bộ ngay từ đầu, có thể làm giảm performance.
* Có thể gặp lỗi ở run-time thay vì compile-time.

1. **Profiles**

Spring Profiles cung cấp một cách để tách biệt các phần trong cấu hình ứng dụng của bạn và làm cho nó chỉ khả dụng trong một số môi trường nhất định. Bất kỳ @Component hoặc @Configuration đều có thể được đánh dấu bằng @Profile để giới hạn khi được tải.

Các bạn hãy xem hình minh họa sau đây là có thể hiểu ngay Spring Profiles.



Chúng ta có thể thấy trong project có rất nhiều file application . Nhưng  chúng đảm nhiệm 1 chức năng khác nhau:

**application.properties**– Phần này bao gồm các chủ đề về cài đặt và đọc thuộc tính và cài đặt cấu hình và tương tác của chúng với các ứng dụng Spring Boot. File này là file mặc định.

**application-dev.properties**  
– Phần này bao gồm các cài đặt dành cho nhóm phát triển.

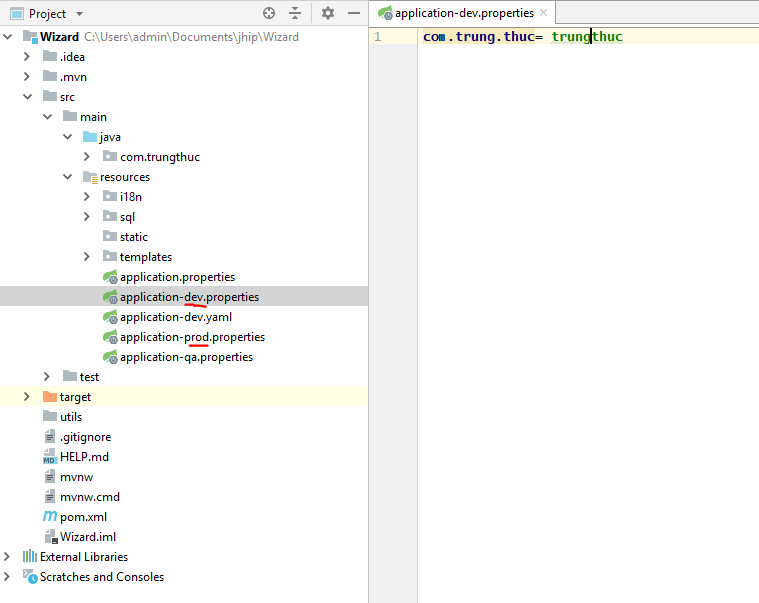
**application-qa.properties**– Phần này bao gồm các cài đặt dành cho QA và tester.

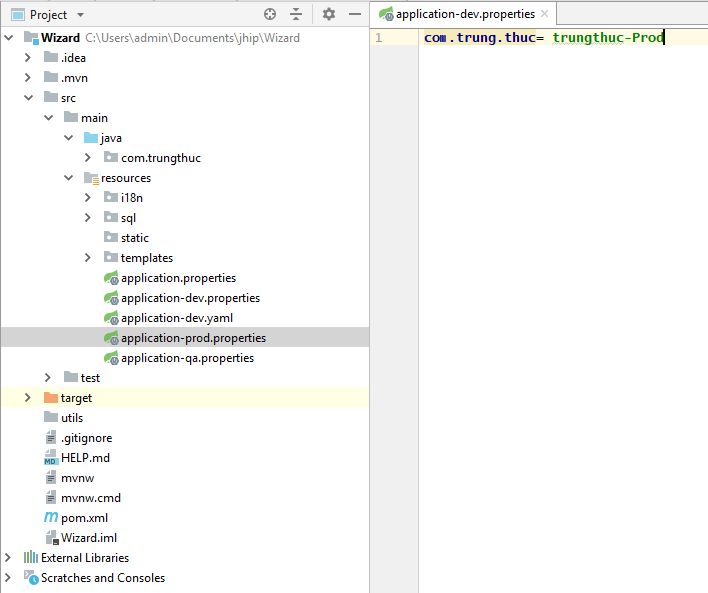
**application-prod.properties**– Phần này bao gồm các cài đặt cho ứng dụng khi dự ấn đã hoàn thành và vận hành dựa vào cài đạt trong file này.

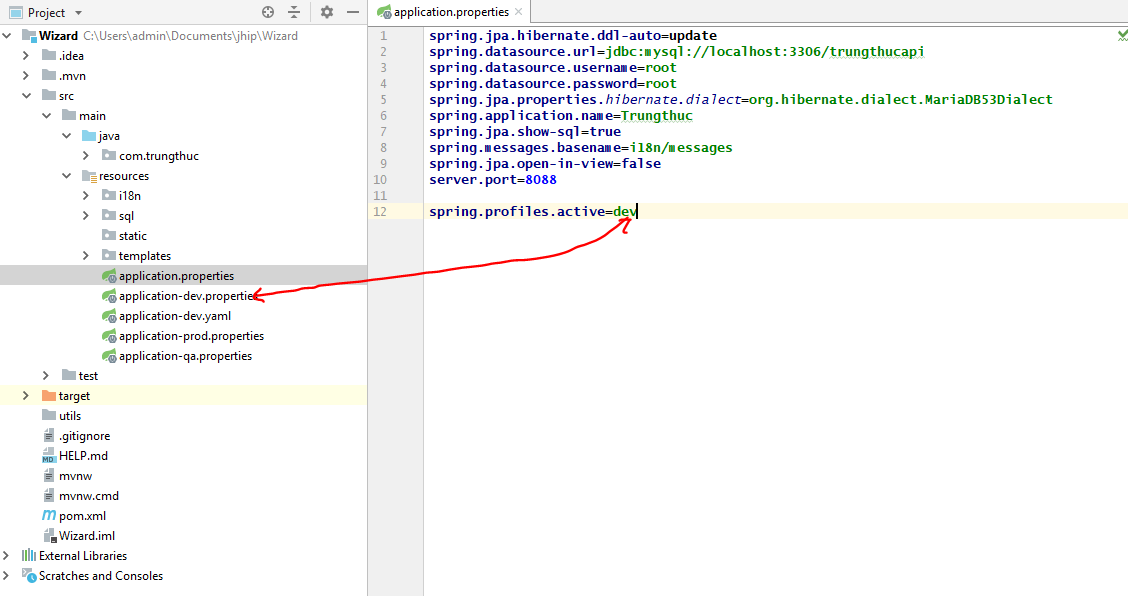
**Trong spring profile các file application-xxxx.properties  name được quy định lấy tên sau dấu “-” của tên file.**

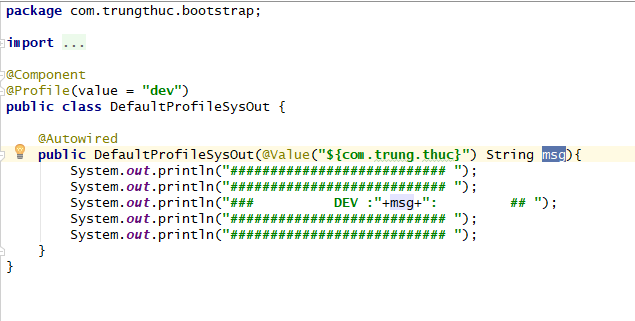
vd: **application-prod.properties profile sẽ có tên là prod.**

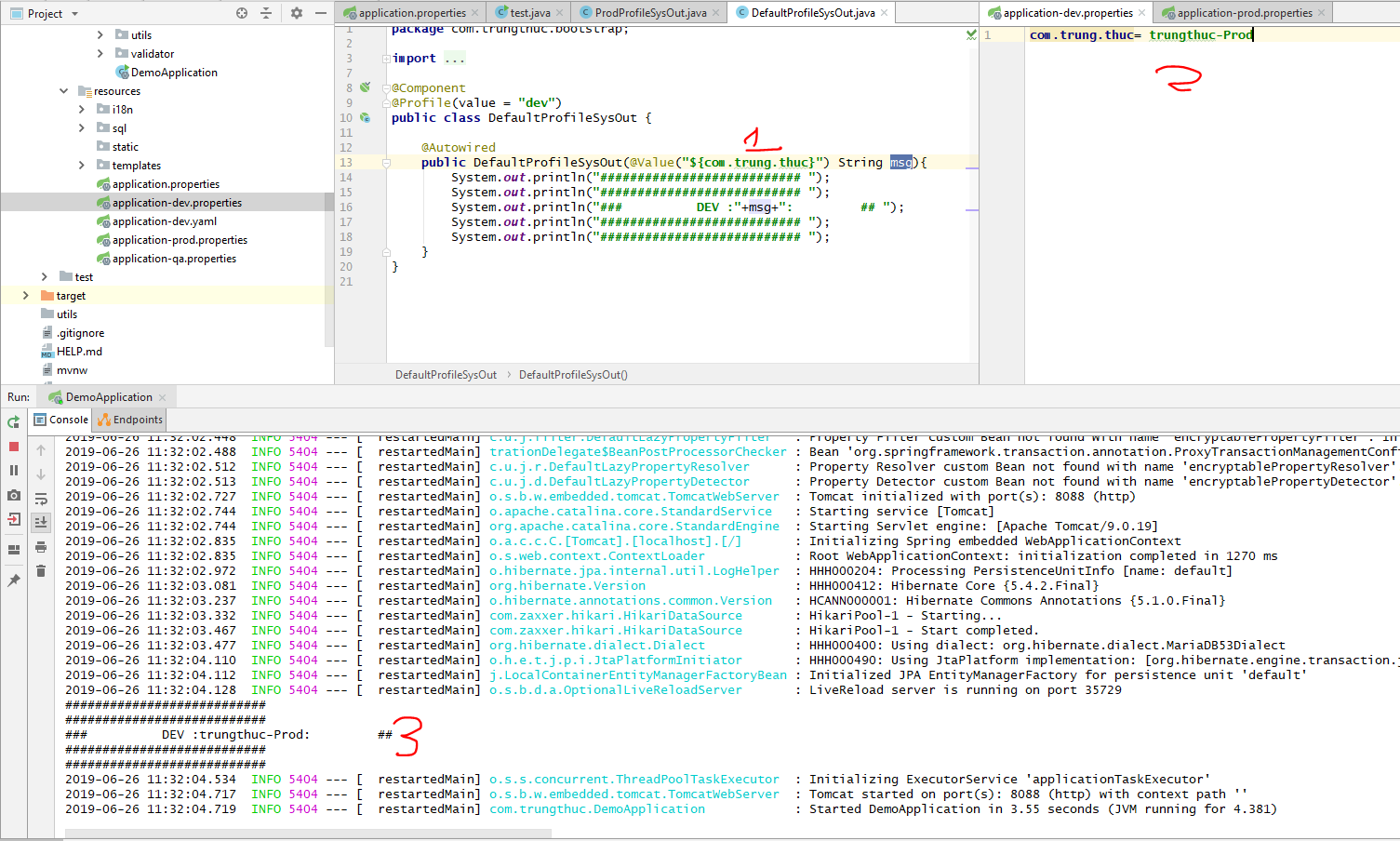
Để hiểu rõ hơn sau đây mình sẽ minh họa cho các bạn cách  sử dụng Spring profile. Trong minh họa này mĩnh sẽ sử dụng 2 profile là dev và prod.

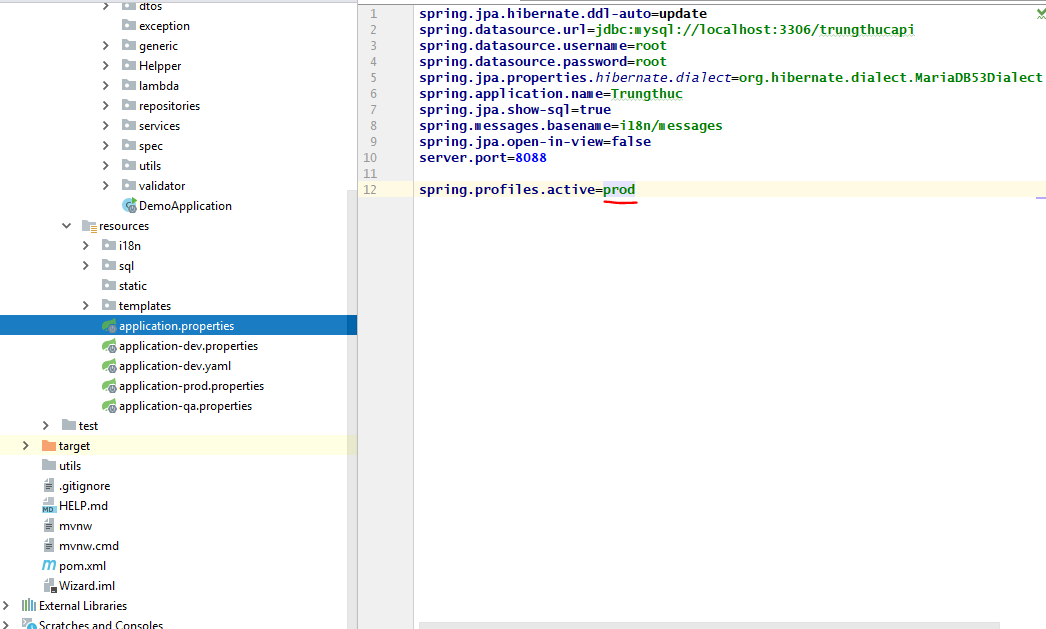
**application-dev.properties (profile = “dev” lấy sau dấu “-“)**

**application-dev.properties (profile = “prod” lấy sau dấu “-“)**

Config application.properties.Mục đích là để thông báo cho app là App sẽ sửu dụng profile nào. Ở đây mình sử dụng profile dev.

Test kết quả PROFILE devXây dựng class để lấy thông tin từ profile.

Chạy ứng dụng.

Test kết quả PROFILE prodBây h thử chuyển sang cài đặt của prod (thông thương dùng cho sản phẩm chính thức (bàn dao cho khách hàng)).

Kết quả PROFILE prod