**XỬ LÝ SỰ KIỆN VÀ BẤT ĐỒNG BỘ**

* Việc xử lý sự kiện giúp tách biệt các phần của hệ thống và giúp mở rộng, linh hoạt hơn trong việc xử lý các tác vụ khác nhau mà không phải thay đổi trực tiếp trong class gốc
* Các thành phần trong xử lý sự kiện:
  + Source: là đối tượng gây ra sự kiện
  + Event: là sự kiện - đối tượng chứa thông tin về sự kiện diễn ra
  + Publisher: là đối tượng phát đi sự kiện và thông báo cho các Listener biết về sự kiện đó
  + Listener: là đối tượng lắng nghe và xử lý sự kiện, có thể có nhiều Listener để xử lý sự kiện theo các cách khác nhau

Áp vào thực tế: Giả sử chúng ta có một cái chuông cửa, khi có người tới bấm cái chuông này. Chuông sẽ phát ra tiếng kêu. Ở trong nhà có chó, nó nghe thấy tiếng kêu, nó sẽ sủa lên thì người bấm chuông cửa là Source, sự kiện bấm chuông cửa là Event, chuông là Publisher phát đi âm thanh để thông báo và con chó là Listener lắng nghe và xử lý sự kiện bằng cách sủa lên

* Để lắng nghe sự kiện trong Spring Boot, chúng ta có thể sử dụng hai cách:
  + Implement interface ApplicationListener: Đây là cách truyền thống và linh hoạt, cho phép chúng ta tuỳ chỉnh logic xử lý sự kiện một cách tường minh

A picture containing text, screenshot, font, line

Description automatically generated

* + Sử dụng **@EventListener**: Spring Framework cung cấp **@EventListener** cho phép chúng ta đánh dấu phương thức để lắng nghe và xử lý sự kiện. Khi sự kiện xảy ra, các phương thức được đánh dấu bằng **@EventListener** sẽ được gọi. Đây là một cách tiện lợi và ngắn gọn để xử lý sự kiện, và nó không yêu cầu triển khai interface ApplicationListener

A picture containing text, screenshot, font, line

Description automatically generated

* Demo:
  + Một event (sự kiện) muốn được Spring Boot hỗ trợ thì sẽ phải kế thừa lớp ApplicationEvent. Nhưng kể từ Spring Framework 4.2, interface ApplicationEventPublisher cung cấp phương thức publishEvent(Object event) để chấp nhận bất kỳ đối tượng nào làm sự kiện. Do đó, chúng ta có thể sử dụng bất kỳ đối tượng nào làm event mà không cần extends lớp ApplicationEvent

A picture containing text, screenshot, font, line

Description automatically generated

* + Trong Spring Boot, để bắn ra một sự kiện, chúng ta sử dụng đối tượng ApplicationEventPublisher. Đây là một Bean có sẵn trong Context do Spring cung cấp, vì vậy chúng ta có thể lấy ra và sử dụng

A picture containing text, screenshot, font, line

Description automatically generated

* + Để lắng nghe các sự kiện do ApplicationEventPublisher bắn ra, chúng ta có thể sử dụng **@EventListener**. **@EventListener** được đánh dấu trên một method, với tham số đầu vào chính là sự kiện mà chúng ta muốn lắng nghe

A screen shot of a computer code

Description automatically generated with low confidence

* + Chạy thử và kết quả

A screenshot of a computer program

Description automatically generated with low confidence

A picture containing text, font, screenshot, algebra

Description automatically generated

Nhận thấy: Theo mặc định, các sự kiện trong Spring là đồng bộ (synchronous)

* Các sự kiện trong Spring Boot mặc định là đồng bộ (synchronous). Để thực hiện xử lý bất đồng bộ cho các sự kiện, chúng ta có thể sử dụng **@Async** để đánh dấu phương thức xử lý sự kiện và bật tính năng bất đồng bộ bằng cách thêm **@EnableAsync** vào lớp cấu hình

A picture containing text, font, white

Description automatically generated

A screen shot of a computer code

Description automatically generated with low confidence

* Nếu chúng ta không tạo bean TaskExecutor, Spring Boot sẽ sử dụng SimpleAsyncTaskExecutor làm TaskExecutor mặc định. Với SimpleAsyncTaskExecutor, các phương thức xử lý sự kiện sẽ được thực hiện bất đồng bộ trong các luồng riêng biệt. Tuy nhiên, nó không cung cấp sự quản lý tốt về tài nguyên và hiệu suất của các luồng đó. Để tận dụng tối đa hiệu suất và quản lý tài nguyên trong việc xử lý bất đồng bộ, chúng ta có thể tạo một bean TaskExecutor và cấu hình nó trong lớp cấu hình của ứng dụng. Điều này cho phép chúng ta tùy chỉnh và điều khiển cách thức thực hiện xử lý bất đồng bộ, bao gồm quản lý số luồng, thời gian chờ và các thuộc tính khác liên quan đến tài nguyên và hiệu suất

A picture containing text, screenshot, font, number

Description automatically generated

* Kết quả sau khi chạy bất đồng bộ:

A picture containing text, font, screenshot, line

Description automatically generated