Spring Boot là một dự án phát triển bởi JAVA (ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.

Để phát triển một ứng dụng web cơ bản HelloWorld sử dụng Spring framework bạn sẽ cần ít nhất 5 công đoạn sau;

Tạo một project sử dụng Maven với các dependency cần thiết của Spring MVC và Servlet API.

Một tập tin web.xml để khai báo DispatcherServlet của Spring MVC.

Một tập tin cấu hình của Spring MVC.

Một class Controller trả về một trang “Hello World” khi có request đến.

Cuối cùng là phải có một web server dùng để triển khai ứng dụng lên chạy.

Trong các công đoạn này, chỉ có công đoạn tạo một class Controller thì có thể khác cho các ứng dụng khác nhau vì mỗi ứng dụng có một yêu cầu khác nhau. Còn các công đoạn khác thì như nhau.

Giờ đây với Spring Boot, chúng ta có thể tạo dự án Spring một cách nhanh chóng và cấu hình cũng đơn giản dùng Sublime Text để phát triển luôn khỏi cần cài đặt eclipse hay netbean nặng bỏ bà.

Dưới đây là một số tính năng nổi bật của Spring Boot:

Tạo các ứng dụng Spring độc lập

Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần phải deploy ra file WAR)

Các starter dependency giúp việc cấu hình Maven đơn giản hơn

Tự động cấu hình Spring khi cần thiết

Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML …

![image](https://github.com/vinhbui212/vccorp\_task2/assets/100837872/d5db9641-972b-49e6-825a-733b95a20f52)

Các thành phần của Spring

- Lớp Controller:

Lớp này nằm trên cùng của kiến trúc Spring Boot. Nó chịu trách nhiệm:

Thực hiện authentication (xác thực).

Chuyển đổi dữ liệu JSON thành đối tượng (và ngược lại).

Xử lý các HTTP request

Truyền authentication tới lớp bussiness

Class Controller xử lý tất cả các REST API request (GET, POST, PUT, DELETE, PATCH) đến từ client

- Lớp Service:

Lớp này chịu trách nhiệm:

Thực hiện validation.

Thực hiện authorization (uỷ quyền).

Xử lý các logic và quy tắc nghiệp vụ.

là nơi xử lý logic nghiệp vụ. Ngắn gọn thì logic nghiệp vụ trong kỹ nghệ phần mềm là những gì mà chúng ta xác định phần mềm cần phải làm.

Một ví dụ là validation. Nếu bạn muốn validate gì đó, nó phải được thực hiện trong class Service.

- Interface repository

Lớp này chịu trách nhiệm:

Chứa các logic lưu trữ

Lấy các đối tượng và chuyển đổi thành các hàng trong database (và ngược lại).

Chúng ta viết các truy vấn tới database trong interface này.Giao tiếp với lớp Service và database

- Lớp model

Tượng trưng cho 1 bảng trong database thường dùng hibernate để dễ dàng tạo bảng trong database.

\* Ưu điểm

- Tự động cấu hình một cách mạnh mẽ được nhúng sẵn server khác với java web cũ phải tải và cài đặt cấu hình rất nhiều

- Ghi đè các bean cũ để sử dụng lại nếu cần

\*Nhược điểm

-Thiếu kiểm soát. Spring Boot tạo ra rất nhiều dependency không được sử dụng, dẫn đến file triển khai có dung lượng lớn.

-Quá trình phức tạp và tốn thời gian để chuyển đổi một dự án Spring cũ hoặc hiện có thành các ứng dụng Spring Boot.

- Các thành phần trong spring core:

1. IOC: Đảo ngược điều khiển, nó giúp làm thay đổi luồng điều khiển của chương trình một cách linh hoạt.

Thường dùng với Denpendency Injection: là tiêm sự phụ thuộc class vào class khác

Constructor Injection: Các dependency sẽ được truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất. (ví dụ trên mình dùng theo cách này)

Setter Injection: Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter/Getter

2. Spring IOC container

Trong Spring, Spring Container (IoC Container) sẽ tạo các đối tượng, lắp rắp chúng lại với nhau, cấu hình các đối tượng và quản lý vòng đời của chúng từ lúc tạo ra cho đến lúc bị hủy.

Spring container sử dụng DI để quản lý các thành phần, đối tượng để tạo nên 1 ứng dụng. Các thành phần, đối tượng này gọi là Spring Bean

IoC Container trong Spring có 2 kiểu là:

BeanFactory

ApplicationContext

Sự khác nhau giữa BeanFactory và ApplicationContext:

BeanFactory và ApplicationContext đều là các interface thực hiện IoC Container. ApplicationContext được xây dựng BeanFactory nhưng nó có thêm một số chức năng mở rộng như tích hợp với Spring AOP, xử lý message, context cho web application.

3. Spring Bean

Spring Bean là các object trong Spring Framework, được khởi tạo thông qua Spring Container. Bất kỳ class Java POJO nào cũng có thể là Spring Bean nếu nó được cấu hình và khởi tạo thông qua container bằng việc cung cấp các thông tin cấu hình (các file config .xml, .properties..)

Có 5 scope được định nghĩa cho Spring Bean:

Singleton: Chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container. Đây là scope mặc định cho spring bean. Khi sử dụng scope này cần chắc chắn rằng các bean không có các biến/thuộc tính được share.

Prototype: Một thể hiện của bean sẽ được tạo cho mỗi lần được yêu cầu(request)

Request: giống với prototype scope, tuy nhiên nó dùng cho ứng dụng web, một thể hiện của bean sẽ được tạo cho mỗi HTTP request.

Session: Mỗi thể hiện của bean sẽ được tạo cho mỗi HTTP Session

Global-Session: Được sử dụng để tạo global sesion bean cho các ứng dụng Portlet.

Ví dụ về cách config bean :

<bean id="address" class="stackjava.com.springdiobject.demo.Address">

<property name="country" value="Viet Nam"></property>

<property name="province" value="Ha Noi"></property>

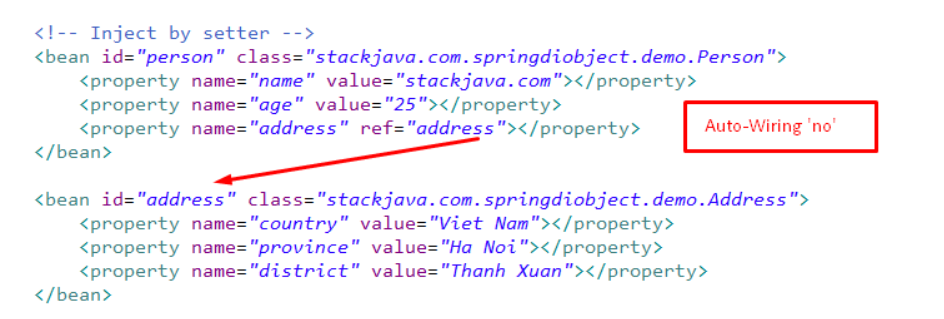
<property name="district" value="Thanh Xuan"></property>

</bean>

4. Spring autowiring

- Việc  Spring Container tự động tìm một bean khác để inject nó vào.

- Cách inject bằng thuộc tính ref



Cách này thường config lâu và cầu kì hơn bây giờ thường dùng annotation @Autowired

@Autowired(required = false)

private Address address;

*//hoặc*

@Autowired(required = false)

public void setAddress(Address address) {

this.address = address;

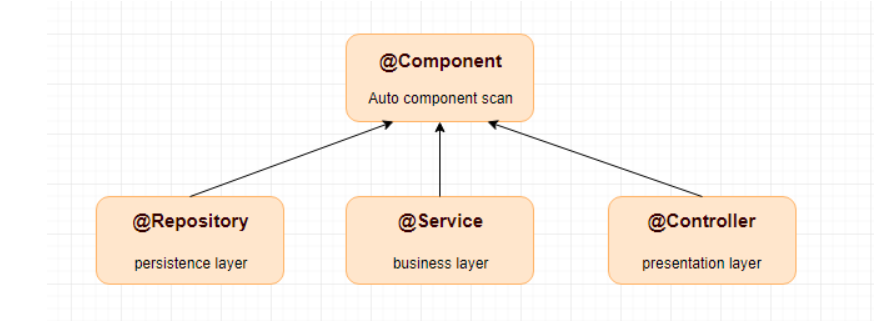
}

Cho spring container tự động quét để tìm bean thích hợp

5. @Component

Các Annotation:

* @Component – biểu thị đây là một component được tự động scan.
* @Repository – biểu thị đây là  một DAO component trong tầng persistence.
* @Service – biểu thị đây là một  Service component trong tầng business.
* @Controller – biểu thị đây là một Controller component trong tầng presentation



Vậy khi nào dùng @Repository,@Service hay @Controller?

Thực ra cả 4 annotation này chỉ dùng với mục đích đánh đấu là auto component scan, bạn có thể dùng chúng lẫn lộn, hoặc nếu không rõ class đang ở tầng persistence, business hay presentation thì cứ dùng @Component nó vẫn hoạt động.

Tuy nhiên bạn nên dùng các annotation @Repository,@Service hay @Controller một cách phân biêt, phù hợp với các tầng chức năng của nó, như thế code của bạn sẽ dễ hiểu, dễ đọc hơn.

Bean:

Bean được cấu hình và quản lý bởi Spring container. Container này chịu trách nhiệm về vòng đời của Bean, từ tạo lập, tiêm phụ thuộc, đến hủy bỏ Bean.

Các Bean có thể được cấu hình bằng XML, annotation, hoặc Java configuration.

Component:

Component là một phần của quá trình tự động phát hiện Bean trong Spring. Khi bạn đánh dấu một lớp với @Component, Spring container sẽ tự động quét lớp này và tạo Bean cho nó.

Các Component có thể có thêm các chú thích như @Autowired để tiêm phụ thuộc tự động.

\*\*

Giống nhau:

Cả Bean và Component đều là các đối tượng Java được quản lý bởi một container (như Spring).

Đều có thể được tiêm các phụ thuộc (Dependency Injection).

Khác nhau:

Bean là một khái niệm tổng quát hơn trong Spring, nó có thể là bất kỳ đối tượng nào được Spring container quản lý.

Component là một chú thích (annotation) cụ thể được sử dụng để tự động phát hiện và đăng ký các Bean với Spring container.

Sử dụng @Component

@Component được sử dụng để đánh dấu một lớp Java là một Spring bean. Đây là cách sử dụng đúng:

import org.springframework.stereotype.Component;

@Component

public class SimpleComponent {

public void printMessage() {

System.out.println("Hello from SimpleComponent");

}

}

Sử dụng @Bean

@Bean thường được sử dụng trong một lớp cấu hình (@Configuration) để khai báo một phương thức tạo ra và trả về một bean.

Ví dụ, bạn có thể tạo một lớp cấu hình và sử dụng @Bean như sau:

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public SimpleComponent simpleComponent() {

return new SimpleComponent();

}

}