SpringBoot Notes

Dependency Injection, API 3 Layer Architecture

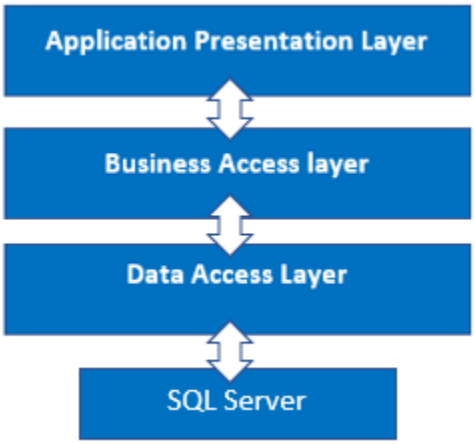
Hai trong số những khái niệm đầu tiên được tiếp cận khi tìm hiểu về: cách viết API với Spring Boot chính là DI-triết lý đằng sau thư viện Spring và cấu trúc API 3 lớp cơ bản để viết mã nguồn bất kỳ API nào.

Dependency Injection

Dependency Injecttion có thể hiểu 1 cách đơn giản là, trong quá trình bắt đầu App, khởi tạo **tất cả các lớp thành phần được khai báo,** sau đó nhồi nhét các Object cấu thành nên một Object khác vào Object cha đó. Điều này sẽ không làm thay đổi cấu trúc phân tầng lớp theo thiết kế UML ban đầu của App, nhưng nó lại đơn giản hóa việc viết code, giờ đây, thay vì fix cứng Object nào cấu thành nên Object nào và phải viết construction code cho Object thành viên (dependency, cùng với config của chúng) trong từng Object, ta sẽ bốc dependency từ Container và auto-inject nó vào trong Object cần nó khi đang chạy App.

3-Layer API architecture

Spring Boot là thư viện được cài đặt trên nền tảng của framework Spring, với mục tiêu đơn giản hóa việc viết code và viết config của Spring. Spring Boot được sử dụng nhiều để viết backend webapp, API. Các RESTAPI được viết bằng SpringBoot sẽ đi theo phân tầng hệ thống 3 lớp:



Mỗi tầng sẽ được biểu diễn bằng 1 lớp, và thành phần của lớp đó sẽ chứa object tầng dưới nó. Những tầng này còn được biết đến với tên phổ biến hơn: API layer, Service Layer và DAO(Data-Access-Objects) Layer, riêng DAO, vì độ tái sử dụng cao nên thường không cần phải code mà thay vào đó người dùng sẽ sử dụng những Interface có sẵn tên là Repository. Tuy nhiên, đôi khi Repository sẽ không cần thiết (VD: API đơn giản, code đơn giản không dùng hết tính năng của Repo, các tính năng của Repo không phù hợp với yêu cầu của các Service và cần 1 kết nối "sâu" hơn với DB), thay vào đó, một DB Connection Config sẽ được viết để thiết lập kết nối với DB, rồi

từ đó khởi tạo một template để sử dụng giao tiếp trực tiếp với DB. Ở cách viết này, DAO sẽ chính là Template mang Config chứ không phải Repository.

Spring Boot nâng cấp Spring với khả năng tự động config thay vì config inject object bằng tay như trong SpringMVC thuần túy. SB hiện thực hóa điều này bằng cách đi theo cấu trúc Singleton, và giới thiệu rất nhiều Java Annotations để giúp Spring Container auto-inject những object cần thiết vào object cần khởi tạo trong quá trình khởi động API.

Báo cáo này sẽ không giới thiệu Spring mà tập trung vào những Spring Boot Annotation cơ bản và hữu ích, xoay quanh việc lập trình một API.

Spring Boot basic Annotations

Các Annotation là các "tag" được gắn vào Java Object để bổ sung metadata cho chúng. Đối với Spring container, đây là một trong những thông tin hữu ích để nó làm nhiều việc như Auto-Config, Auto-Inject, thậm chí là binary-codegen (thư viện Lombok),…

Dưới đây sẽ là giới thiệu ngắn về một số Annotation được sử dụng nhiều trong Project API vừa rồi và cũng là bộ Annotation cơ bản trong SpringWeb dùng để viết các API.

Bean, Component, Autowired

Annotation cơ bản nhất của SpringWeb sẽ là bộ ba Bean, Component và Autowired, chúng có liên hệ mật thiết với ý tưởng DI.

@Bean là Annotation cơ bản nhất, đánh vào Class và method, anno này thể hiện Java Object này sẽ được khởi tạo trong quá trình khởi động SpringApp và đưa vào Container để quản lý. Điều này đồng nghĩa với việc tất cả các Spring Bean đã sẵn sàng để được inject. thường được dùng với @AllArgsConstructor để có luôn 1 hàm khởi tạo với đầu vào là tất cả các biến.

@Component Cũng như Bean, nhưng chỉ được đánh vào Class.

@Autowired đánh vào class attribute, thể hiện rằng attr này sẽ được tìm ở trong đống Components được quản lý bởi Container trong khi bắt đầu App, và auto-inject vào Object.

PostConstruct & PreDestroy

Như tất cả các ngôn ngữ lập trình, ta sẽ muốn làm gì đó ngay sau khi khởi tạo Object, có thể là check connection đến Database, hoặc đơn giản là log vào file là Object đã khởi tạo với các biến này này. Bên cạnh đó là khả năng làm gì đó trước khi phá huỷ Object, VD như là dump dữ liệu vào file log, gửi 1 alert đến dashboard nào đó,…

(Nếu đầy đủ, phần này phải đề cập tới *Spring Bean life cycle* nhưng mà nó sẽ out-of-scope với báo cáo này. Người đọc có thể tìm hiểu thêm qua docs chính thức của Spring.)

Hai hàm đó sẽ được annotate với PostConstruct và PreDestroy.

@PostConstruct: dành cho class method của Bean, làm gì đó ngay sau khi khởi tạo Bean

@PreDestroy: dành cho class method của Bean, làm gì đó ngay trước khi xóa Bean khỏi RAM

Service, Repository, Configuration

Như đã đề cập ở trên, sau đây là 3 annotation dùng để thông báo cho Container biết 3 lớp của API. Về tính năng thì chúng không có gì đặc biệt, với usecase đơn giản thì các anno này chỉ dùng để đánh dấu Class để phân tầng cho chúng.

@Service dùng để định nghĩa đây là 1 object thuộc tầng Service/BusinessLogic

@Config dùng để định nghĩa đây là 1 object thuộc tầng DAO, cụ thể là config để kết nối đến DB. cái này sẽ được dùng với anno kế

@Repository gần như chỉ được dùng để annotate Interface. Đây là anno thể hiện rằng Class này là một DAO. nó sẽ dùng config ở trên đã đề cập để kết nối với 1 DB nào đó. Thường thì Interface Repo sẽ extend CrudRepository và thừa kế toàn bộ các tính năng CRUD cơ bản, Còn CRUD đến DB nào thì Config đã đề cập và auto-inject rồi, người lập trình không cần để ý nữa.

Value, RestController, SpringBootApplication, AutoConfiguration

Đây là bộ 4 anno cơ bản thuộc tầng cao nhất trong sơ đồ cấu trúc 3 lớp ở chương I. Dùng để Annotate tầng cao nhất và đóng gói toàn bộ App của chúng ta.

@Value dùng cho class attribute, thông báo với Container là attr này được lấy trong resources/application.properties hoặc bất kỳ file properties nào trong dir này. VD:

@Value(“redis.host”) String redisHost;

@RestController chính là tầng API, các method trong Class mang anno này sẽ được annotate với RequestMapping, GetMapping, PostMapping,… cùng với đường dẫn tương đối để map request route trong API đến business logic của nó. Các class RestController mang 1 class Service để thực hiện request mapping. Sẽ có báo cáo riêng cho RequestMapping và API routing, hiện tại báo cáo này không đề cập tới phần này.

@SpringBootApplication dành cho Class, đây là Anno sẽ gói tất cả mọi thứ vào, không có tính năng gì dặc biệt, **có duy nhất 1 hàm main để bắt đầu app**. Có thể lập trình thêm các logic cần thiết, nhưng trong thực tế thì đây là boilerplate code.