Chức năng của framework này có thể áp dụng cho bất kỳ ứng dụng Java nào, Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code.

2. **Cách thức hoạt động của Spring FrameWork**

- **Aspect-oriented Programming (Lập trình hướng chức năng):** Là một phương pháp lập trình trong Spring Framework, AOP tập trung vào chia nhỏ chức năng của ứng dụng thành các khía cạnh riêng lẻ và chèn chúng vào source code một cách linh hoạt. AOP được sử dụng trong Spring Framework để:

▪ Cung cấp dịch vụ doanh nghiệp khai báo. Dịch vụ quan trọng nhất là

quản lý giao dịch khai báo .

▪ Cho phép người dùng triển khai các khía cạnh tùy chỉnh, bổ sung việc sử dụng OOP của họ bằng AOP

Bức ảnh minh họa các khái niệm chính trong **Lập trình hướng khía cạnh (Aspect-Oriented Programming - AOP)**. Đây là một mô hình lập trình giúp tách biệt các mối quan tâm chung (cross-cutting concerns) như logging, bảo mật, quản lý giao dịch khỏi logic chính của ứng dụng.

**Giải thích các thành phần trong ảnh:**

1. **Join point**: Điểm trong luồng thực thi của chương trình mà tại đó một hành động có thể được chen vào (ví dụ: phương thức được gọi, một ngoại lệ được ném).
2. **Advice**: Mã được thực thi tại một join point. Advice có thể là Before (trước), After (sau), hoặc Around (xung quanh) một join point.
3. **Pointcut**: Một tập hợp các join points nơi advice sẽ được áp dụng. Đây là nơi quy định điểm nào trong mã nguồn sẽ bị can thiệp.
4. **Target Object**: Đối tượng có phương thức bị cắt ngang bởi AOP (đối tượng bị áp dụng aspect).
5. **Aspect**: Một mô-đun chứa logic của cross-cutting concerns, chẳng hạn như logging, bảo mật hoặc xử lý giao dịch.
6. **Proxy**: Một đối tượng ủy quyền giúp chèn các aspect vào target object mà không thay đổi mã nguồn gốc.
7. **Weaving**: Quá trình tích hợp aspect vào mã nguồn của chương trình. Có thể thực hiện trong giai đoạn biên dịch, load-time hoặc runtime.

AOP thường được sử dụng trong các framework như **Spring AOP** để giúp quản lý mã sạch hơn và dễ bảo trì hơn bằng cách tách biệt các concern ra khỏi logic chính của ứng dụng.

- **Dependency Injection – DI ( Chèn khía cạnh)** : *giúp quản lý và giảm sự phụ thuộc giữa các thành phần trong ứng dụng. Thay vì tạo đối tượng một cách trực tiếp, DI cho phép tạo các đối tượng phụ thuộc bên ngoài một class và cung cấp các đối tượng đó cho một class theo nhiều cách khác nhau*, giúp tăng tính linh hoạt, tái sử dụng và kiểm thử của source code. Chúng ta có thể định nghĩa khái niệm Spring bằng Inversion of Control (IoC – đảo ngược kiểm soát).

- **Inversion of Control :** *một mô hình* lập trình *được áp dụng* trong những ứng dụng phần mềm *với mục đích quản lý sự phụ thuộc giữa những thành phần trong code*. Đây là một phương thức tiếp cận rất khác biệt so với phong cách lập trình truyền thống. Trong đó các đối tượng sẽ tự quản lý chính bản thân mình đồng thời tương tác với nhau để có thể hoàn thành công việc.

Trong ảnh, ta thấy:

Con ngựa kéo chiếc xe — như thường lệ.

Nhưng bây giờ: con ngựa ngồi trên xe và “kéo” con người đi — một cách "đảo ngược vai trò".

Đây chính là một phép ẩn dụ hài hước cho “Inversion of Control” — quyền kiểm soát bị đảo ngược.

Ý nghĩa trong lập trình:

Trước đây: code của bạn (developer) tự điều khiển mọi thứ — bạn gọi và kiểm soát các đối tượng.

Sau này với IoC (ví dụ như Spring): framework điều khiển, còn bạn chỉ viết các đoạn logic cần thiết. Framework sẽ tự tạo, quản lý và cung cấp đối tượng khi cần.

Ví dụ: Class X phụ thuộc vào class Y. Sự phụ thuộc này có thể tạo ra nhiều vấn đề trên thực tế, bao gồm cả lỗi hệ thống. Do đó cần phải tránh sự phụ thuộc như vậy. Spring IOC giải quyết các vấn đề phụ thuộc như vậy bằng Dependency Injection. Ở đây, nó chỉ ra rằng class Y sẽ được IoC đưa vào class X.

**3. Các tính năng của Spring FrameWork**

• Kiến trúc tổng thể của Spring FrameWork:

**a) Core Container:**

- Spring Core: Module này là thành phần cơ bản của Spring Framework, cung cấp các tính năng cốt lõi như Inversion of Control (IoC) và Dependency Injection (DI). IoC container là trái tim của Spring, có nhiệm vụ tạo và quản lý các đối tượng JavaBeans.

- Spring Beans: Cung cấp BeanFactory – một thành phần cơ bản của IoC container và BeanWrapper, chịu trách nhiệm quản lý vòng đời của beans. BeanFactory là giao diện chính giúp truy xuất các bean từ container, hỗ trợ nhiều phương thức để lấy bean theo yêu cầu.

- Spring Context: Module này mở rộng BeanFactory và cung cấp các tính năng nâng cao như quốc tế hóa, quản lý tài nguyên, cũng như các chức năng liên quan đến xuất bản và xử lý sự kiện.

- Spring Expression Language (SpEL): Đây là ngôn ngữ biểu thức mạnh mẽ cho phép truy vấn và thao tác đối tượng trong thời gian chạy. SpEL hỗ trợ các tính năng như truy cập thuộc tính, gọi phương thức, điều kiện, vòng lặp, và chuyển đổi kiểu dữ liệu.

b) **Data Access / Intergration (Truy Cập và Tích Hợp Dữ Liệu):**

Phần này cung cấp khả năng tích hợp với cơ sở dữ liệu và các nguồn dữ liệu khác:

- Spring JDBC: Module này giúp đơn giản hóa việc làm việc với JDBC bằng cách giảm thiểu mã cần viết. Spring JDBC hỗ trợ quản lý giao dịch, giúp nhà phát triển quản lý giao dịch cơ sở dữ liệu một cách dễ dàng.

- Spring ORM: Cung cấp hỗ trợ cho các framework ORM như Hibernate và JPA. Spring ORM giúp giảm bớt mã boilerplate và tích hợp các công nghệ ORM với các tính năng khác của Spring, như quản lý giao dịch và caching.

- Spring Data: Module này mang đến một mô hình lập trình dễ sử dụng để làm việc với các công nghệ truy cập dữ liệu, bao gồm cơ sở dữ liệu, NoSQL và các dịch vụ dữ liệu đám mây. Các tính năng như CRUD tự động và hỗ trợ phân trang được tích hợp.

- Spring Transaction: Hỗ trợ quản lý giao dịch trong ứng dụng Spring, bao gồm các cấp độ giao dịch và chiến lược quản lý giao dịch khác nhau, như JTA hoặc JDBC.

**c) Web:**

Spring hỗ trợ xây dựng ứng dụng web với các module sau:

- Spring MVC: Module cung cấp framework Model-View-Controller (MVC), hỗ trợ các tính năng như xử lý yêu cầu và phản hồi HTTP, xử lý biểu mẫu và ràng buộc dữ liệu. Spring MVC tương thích với nhiều công nghệ hiển thị như JSP, Thymeleaf và Velocity.

- Spring WebFlux: Đây là một framework lập trình phản ứng dành cho các ứng dụng web yêu cầu khả năng mở rộng và tính tương tác cao. Spring WebFlux hỗ trợ nhiều công nghệ, như Netty, Undertow và các container Servlet 3.1+.

- Spring Web Services: Cung cấp hỗ trợ xây dựng các dịch vụ web SOAP và RESTful. Module này giúp tạo WSDL từ các class Java và ngược lại, giúp xác định hợp đồng dịch vụ web dễ dàng.

**d) Các tính năng khác:**

Các module bổ sung giúp mở rộng tính năng của Spring:

- Spring Security: Cung cấp các tính năng xác thực và phân quyền cho ứng dụng, hỗ trợ kiểm soát quyền dựa trên vai trò hoặc biểu thức, giúp bảo mật các phần của ứng dụng thông qua các chính sách bảo mật linh hoạt.

- Spring Integration: Hỗ trợ xây dựng kiến trúc tích hợp động và sự kiện, hỗ trợ nhiều mô hình như truyền tin và định tuyến, cũng như tích hợp với các hệ thống truyền tin như JMS và Apache Kafka.

- Spring Batch: Cung cấp các công cụ xử lý hàng loạt, bao gồm hỗ trợ gỡ lỗi, kiểm thử công việc và giám sát. Spring Batch cũng tích hợp với các module khác như Spring Data và Spring Integration.

- Spring Cloud: Hỗ trợ xây dựng ứng dụng native đám mây, cung cấp các tính năng như phát hiện dịch vụ, quản lý cấu hình và cân bằng tải. Nó tích hợp với nhiều nền tảng đám mây như AWS và GCP, cùng với công nghệ container và serverless.

➢ Như vậy, Spring Framework đã mở đường cho việc phát triển một hệ sinh thái mạnh mẽ. Điều này cho phép các nhà phát triển tận dụng một bộ công cụ và thư viện toàn diện để đáp ứng các nhu cầu ứng dụng khác nhau.

**4. Ưu và Nhược điểm của Spring FrameWork:**

**a) Ưu điểm:**

- Tính linh hoạt và tính mô-đun.

- Lập trình hướng khía cạnh (AOP).

- Dễ dàng tích hợp.

- Một cách nhất quán để truy cập các nguồn dữ liệu khác nhau.

- Bảo mật và xác thực.

- Nhẹ và linh hoạt về mặt kiến trúc.

- Cộng đồng hỗ trợ mạnh mẽ.

- Được sử dụng trong nhiều lĩnh vực khác nhau – từ hệ thống doanh nghiệp đến các ứng dụng web và dịch vụ siêu nhỏ.

- Giảm mã lặp lại.

- Kiểm tra đơn giản.

**b) Nhược điểm:**

- Cấu hình tốn thời gian. Làm việc với Spring phức tạp và kéo dài hơn Spring Boot vì Spring yêu cầu cấu hình từ đầu cho mỗi dự án. Điều này, đòi hỏi nhiều thời gian phát triển hơn. Cấu hình các ứng dụng Spring, đặc biệt là trong cấu hình dựa trên XML, thường dẫn đến các tệp cấu hình phức tạp và rộng lớn.

- Việc quản lý các phụ thuộc có thể trở nên khó khăn, đặc biệt là khi có số lượng lớn thư viện.

- Thách thức về tích hợp: Việc tích hợp Spring với các công nghệ hoặc thư viện của bên thứ ba có thể đòi hỏi thêm về cấu hình.

**II. Spring boot**

**1. Giới thiệu về Spring Boot**

- Spring Boot là một trong số các module của Spring framework chuyên cung cấp các tính năng RAD (Rapid Application Development) cho phép tạo ra và phát triển các ứng dụng độc lập dựa trên Spring một cách nhanh chóng.

- Spring Boot ra đời với mục đích loại bỏ những cấu hình phức tạp của Spring, nó không yêu cầu cấu hình XML và nâng cao năng suất cho các nhà phát triển. Với sự góp mặt của Spring Boot, hệ sinh thái Spring đã trở nên mạnh mẽ, phổ biến và hiệu quả hơn bao giờ hết.

**2. Các tính năng chính của Spring Boot**

- Auto-configuration (Tự động cấu hình): Spring Boot tự động phát hiện thư viện trong classpath và thiết lập các bean cần thiết mà không yêu cầu cấu hình thủ công. Ví dụ, khi sử dụng spring-boot-starter-web, Spring Boot tự động cấu hình Tomcat làm server mặc định. Điều này giúp giảm thiểu cấu hình phức tạp, cho phép ứng dụng chạy ngay với thiết lập hợp lý, đồng thời vẫn có thể tùy chỉnh khi cần.

- SpringApplication – Khởi chạy ứng dụng dễ dàng: Spring Boot cung cấp lớp SpringApplication, giúp khởi động ứng dụng một cách đơn giản và trực quan từ phương thức main trong Java. Điều này giúp quản lý vòng đời ứng dụng hiệu quả hơn, giảm bớt công đoạn cấu hình thủ công.

- Hỗ trợ quản trị ứng dụng từ xa (Admin Support): Spring Boot cung cấp tính năng hỗ trợ quản trị, cho phép giám sát và quản lý ứng dụng từ xa. Tính năng này có thể được kích hoạt dễ dàng bằng cách thiết lập thuộc tính.

- Cấu hình linh hoạt (Externalized Configuration): Spring Boot cho phép cấu hình ứng dụng theo từng môi trường cụ thể mà không cần thay đổi mã nguồn. Các tệp YAML hoặc properties được sử dụng để tách biệt cấu hình khỏi mã ứng dụng, giúp dễ dàng kiểm soát và thay đổi khi cần thiết.

- Hỗ trợ YAML: Spring Boot hỗ trợ YAML như một phương pháp cấu hình thay thế cho tệp properties truyền thống. YAML giúp tổ chức cấu hình theo cấu trúc phân cấp, dễ đọc và quản lý hơn.

- Cấu hình kiểu an toàn (Type-safe Configuration): Spring Boot cung cấp khả năng cấu hình kiểu an toàn, giúp đảm bảo dữ liệu cấu hình luôn chính xác, giảm thiểu lỗi trong quá trình chạy ứng dụng. Annotation như @ConfigurationProperties giúp lập trình viên dễ dàng quản lý và kiểm soát các thông số cấu hình.

- Hệ thống Logging mạnh mẽ: Spring Boot sử dụng Common Logging để quản lý log mặc định. Các phụ thuộc về logging được cài đặt sẵn, giúp quá trình ghi log trong ứng dụng diễn ra ổn định mà không cần cấu hình thêm.

- Bảo mật với Spring Security: Spring Boot tích hợp Spring Security, giúp bảo mật ứng dụng ngay từ đầu. Mặc định, Spring Boot áp dụng Basic Authentication cho tất cả các endpoint HTTP, giúp bảo vệ hệ thống mà không cần cấu hình phức tạp.

➢ Tóm lại:

- 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 …

**Giải thích ảnh**

**Bức ảnh trình bày một số tính năng của Spring Boot, đặc biệt là:**

**SpringApplication**

Đây là cách khởi chạy một ứng dụng Spring Boot.

Lệnh SpringApplication.run(ClassName.class, args); sẽ giúp Spring Boot tự động cấu hình và khởi động ứng dụng mà không cần nhiều cấu hình phức tạp.

Nó thay thế cách khởi động ứng dụng Java truyền thống bằng cách kết hợp Spring Framework với cơ chế nhúng sẵn.

**Admin Support (Hỗ trợ quản trị)**

Spring Boot hỗ trợ Spring Boot Admin, một công cụ giúp giám sát và quản lý ứng dụng.

Cấu hình trong ảnh kích hoạt chế độ admin cho ứng dụng Spring Boot với các thông số:

spring.application.admin.enabled=1: Bật tính năng admin.

spring.boot.admin.client.username=admin & spring.boot.admin.client.password=admin: Cấu hình tài khoản quản trị viên.

spring.boot.admin.routes.endpoints=env, metrics, trace, jolokia, info, configprops: Chỉ định các endpoint giám sát hệ thống.

spring.boot.admin.notify.hipchat.auth-token: Tích hợp với HipChat để gửi thông báo.

Tóm lại

SpringApplication giúp khởi chạy ứng dụng Spring Boot dễ dàng.

Admin Support giúp theo dõi và quản lý ứng dụng với các tính năng giám sát, cấu hình và thông báo.

**3. Ưu điểm của spring boot**

- Tích hợp các tính năng của Spring Framework, giúp phát triển ứng dụng một cách linh hoạt và mạnh mẽ.

- Đơn giản hóa cấu hình: Spring Boot cung cấp cấu hình mặc định hợp lý, giúp lập trình viên không cần mất nhiều thời gian thiết lập các tệp cấu hình như trong Spring Framework truyền thống. Nhờ Spring Boot Starter và Auto- configuration, ứng dụng có thể chạy nhanh chóng mà không cần nhiều thao tác thủ công.

- Dễ dàng triển khai do ứng dụng server được nhúng trực tiếp, giúp tránh những khó khăn khi triển khai lên môi trường production mà không cần phải tải file WAR.

- Tăng tốc quá trình phát triển:

▪ Spring Boot Starter giúp quản lý thư viện phụ thuộc dễ dàng.

▪ Spring Initializr hỗ trợ khởi tạo dự án nhanh chỉ trong vài cú click.

▪ Hỗ trợ hot-reloading, giúp lập trình viên thấy ngay thay đổi trong ứng dụng mà không cần khởi động lại toàn bộ hệ thống.

▪ Tích hợp sẵn các công cụ hiện đại(Spring Security, Spring Web…)

- Cung cấp nhiều plugin, số liệu và hỗ trợ cấu hình ứng dụng từ các nguồn bên ngoài, giúp quản lý và theo dõi ứng dụng dễ dàng hơn.

- Cộng đồng lớn và tài liệu phong phú: Với sự hỗ trợ mạnh mẽ từ cộng đồng lập trình viên Java, Spring Boot có tài liệu phong phú, giúp dễ dàng tìm kiếm giải pháp khi gặp vấn đề trong quá trình phát triển.  
  
**4.Nhược điểm của Spring boot**

quá trình phức tạp và tốn thời gian để chuyển đối.

thiếu kiểm soát.

không phù hợp với dự án quy mô lớn

**III. So sánh spring framework và spring boot**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Spring FrameWork | Spring Boot |
| Cách triển khai | Dạng WAR, cần server | Dạng JAR, chạy độc lập |
| Cấu hình | Thủ công (XML/Java) | Tự động, đơn giản |
| Quản lí thành phần | Tự quản lý, dễ xung đột | Có Starter, dễ tích hợp |
| Khởi tạo | Nhiều bước cấu hình | Khởi tạo nhanh chóng |
| Quản trị & log | Cấu hình thủ công | Có sẵn Actuator & log |
| Bảo mật | Cấu hình chi tiết | Có sẵn Spring Security |
| Tùy chỉnh | Linh hoạt, kiểm soát cao | Tối ưu đơn giản, ít cấu hình |
| Ứng dụng phù hợp | Doanh nghiệp lớn  phức tạp | Dự án vừa, nhỏ, phát triển nhanh |

**IV. Khi nào nên sử dụng**

Trường hợp

Dự án quy mô lớn, nghiệp vụ phức tạp.

Dự án vừa và nhỏ, cần phát triển nhanh.

Cần kiểm soát chi tiết từng thành phần.

Đội ngũ nhiều kinh nghiệm, có khả năng thiết lập cấu hình.

Yêu cầu triển khai theo chuẩn doanh nghiệp (WAR, server riêng).  
Ưu tiên tốc độ phát triển, thử nghiệm, demo sản phẩm.

V. Tổng kết

spring framework Là Java Framework toàn diện để xây dựng ứng dụng doanh nghiệp.

Ưu điểm: Linh hoạt, module hóa, dễ kiểm thử.

Nhược điểm: Phức tạp, nhiều cấu hình.

Cốt lõi: Dependency Injection (DI), Aspect-Oriented Programming (AOP), Invertion of Control (IoC)

Spring boot

* Xây dựng trên Spring Framework, giúp tạo ứng dụng nhanh hơn.
* Ưu điểm: Đơn giản, tự động cấu hình, dễ triển khai.
* Nhược điểm: "Magic" (khó hiểu), có thể overhead.
* Cốt lõi: Auto-configuration, Embedded Servers, Starters.

Spring Framework: Nền tảng mạnh mẽ, cấu hình nhiều.

Spring Boot: Xây dựng nhanh, cấu hình ít.

=> Chọn Spring Boot cho dự án mới, Spring Framework cho ứng dụng cũ cần kiểm soát chi tiết.