

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO SEMINAR

Môn học: Lập trình Java – IS216.P23

Giảng viên: Tạ Việt Phương

Nhóm sinh viên thực hiện:

1. Bùi Tân Chính – MSSV :23520185
2. Nguyễn Trần Bảo Anh – MSSV: 22520066
3. Lê Nguyễn Minh Khang – MSSV: 23520690
4. Nguyễn Khánh Vân – MSSV: 23521772
5. Nguyễn Lê Thanh Hiền – MSSV: 22520590
6. Lê Thị Phương Thảo – MSSV: 23521468
7. Lê Thị Thùy Trang – MSSV: 23521627
8. Nguyễn Thanh Trí – MSSV: 23521645

Thành Phố Hồ Chí Minh, 2025

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Tạ Việt Phương đã hướng dẫn và tạo điều kiện cho chúng em thực hiện seminar về chủ đề “So sánh Spring Framework và Spring Boot”. Nhờ sự chỉ dẫn tận tình của thầy, chúng em đã có cơ hội tìm hiểu sâu hơn về hai công nghệ quan trọng trong hệ sinh thái Java, từ đó nâng cao kiến thức và kỹ năng lập trình của mình.

Chúng em cũng xin chân thành cảm ơn các anh chị và bạn bè trong lớp đã tham gia, lắng nghe và đóng góp những ý kiến quý báu. Những câu hỏi và phản hồi của mọi người không chỉ giúp chúng em nhận ra những điểm cần cải thiện mà còn mở rộng góc nhìn về cách áp dụng Spring Framework và Spring Boot trong thực tế.

Bên cạnh đó, chúng em cũng xin cảm ơn nhóm thực hiện seminar, những người đã cùng nhau nghiên cứu tài liệu, thảo luận và chuẩn bị kỹ lưỡng để buổi trình bày diễn ra suôn sẻ. Nhờ sự nỗ lực của từng thành viên, seminar đã hoàn thành đúng tiến độ và đạt được mục tiêu đề ra.

Cuối cùng, mặc dù đã cố gắng hết sức để chuẩn bị nội dung một cách đầy đủ và chính xác, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em mong nhận được sự góp ý từ thầy cô và các bạn để có thể hoàn thiện hơn trong những lần sau.

Nhận xét của giảng viên:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....., ngày.....tháng.....năm 2023

Người nhận xét

(Ký tên và ghi rõ họ tên)

Bảng Phân Công, Đánh giá Thành Viên

Mục Lục

I.	Mở đầu.....	6
II.	Spring FrameWork.....	6
1.	Spring FrameWork là gì?.....	6
2.	Nguồn gốc ra đời.....	6
3.	Cách thức hoạt động của Spring FrameWork.....	7
4.	Các tính năng của Spring FrameWork.....	8
a)	Core Container:.....	8
b)	Data Access / Intergration (Truy Cập và Tích Hợp Dữ Liệu):.....	9
c)	Web:.....	10
d)	Các tính năng khác:.....	10
5.	Ưu và Nhược điểm của Spring FrameWork:.....	11
a)	Ưu điểm:.....	11
b)	Nhược điểm:.....	11
6.	Khi nào thì nên sử dụng SpringFrameWork:.....	12
III.	Spring Boot.....	13
1.	Giới thiệu về Spring Boot.....	13
2.	Nguồn gốc ra đời.....	13
3.	Sự khác nhau giữa Spring Boot và Spring FrameWork.....	14
4.	Các tính năng chính của Spring Boot.....	16
5.	Lợi ích của Spring Boot.....	18
IV.	Tổng kết:.....	19
	NGUỒN TÀI LIỆU THAM KHẢO:.....	24

I. Mở đầu

Trong thế giới phát triển phần mềm hiện đại, Java vẫn luôn là một trong những ngôn ngữ lập trình phổ biến và mạnh mẽ. Đặc biệt, các công cụ hỗ trợ phát triển ứng dụng web với Java ngày càng trở nên hoàn thiện và dễ sử dụng hơn. Trong số đó, Spring Framework và Spring Boot là hai nền tảng mạnh mẽ giúp các lập trình viên xây dựng các ứng dụng web hiệu quả. Hãy tưởng tượng bạn đang phát triển một ứng dụng web phức tạp, nơi mỗi quyết định công nghệ sẽ tác động trực tiếp đến hiệu suất và tốc độ phát triển. Bạn sẽ chọn gì: Một công cụ mạnh mẽ và linh hoạt như Spring Framework, hay một giải pháp nhanh chóng, tự động hóa và dễ triển khai như Spring Boot? Những công nghệ này không chỉ là công cụ, mà còn là nền tảng giúp các lập trình viên đưa ứng dụng của mình từ ý tưởng đến thực tế. Chúng ta sẽ cùng phân tích sự khác biệt giữa chúng, các tính năng nổi bật và, quan trọng hơn, khi nào bạn nên chọn cái nào để tối ưu hóa quá trình phát triển và triển khai ứng dụng của mình.

II. Spring Framework

1. Spring Framework là gì?

- Framework là một bộ công cụ và thư viện phần mềm được thiết kế sẵn, giúp lập trình viên xây dựng và phát triển ứng dụng một cách nhanh chóng, dễ dàng và có cấu trúc. Nó cung cấp một kiến trúc chung mà lập trình viên có thể sử dụng để phát triển phần mềm mà không phải bắt đầu từ đầu. Hiểu đơn giản là đã có các đơn vị khác làm, chúng ta chỉ học cách sử dụng và sử dụng sao cho hiệu quả, góp ý nếu chỗ nào cần để phần mềm tốt hơn.
- Spring là một Java framework mạnh mẽ, là một framework ứng dụng và bộ chứa đảo ngược điều khiển cho nền tảng Java. Chức năng tính 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. Nguồn gốc ra đời

- Trước đây để lập trình web với java vào những năm 2000, chúng ta hay sử dụng các công nghệ của JEE (tức j2ee hoặc jakarta ee hiện nay) đó là jboss, JSF, EJB, servlet, jsp, jdbc ...
- Tuy nhiên những thứ này khi sử dụng nó có 1 số hạn chế và càng về sau này chúng rất ít được cập nhật phiên bản do đó có nhiều hạn chế, có thể do thiếu nguồn lực nên nó không còn được quan tâm nhiều, vì vậy nó không còn được update liên tục.
- Vì vậy, năm 2004 một nhóm lập trình viên bao gồm: Rod Johnson, Juergen Hoeller, Keith Donald and Colin Sampaleanu đã cùng nhau tạo ra 1 nền tảng ok hơn, khắc phục được 1 số nhược điểm của JEE thời điểm đó, và nó chính là "Spring framework".

3. Cách thức hoạt động của Spring Framework

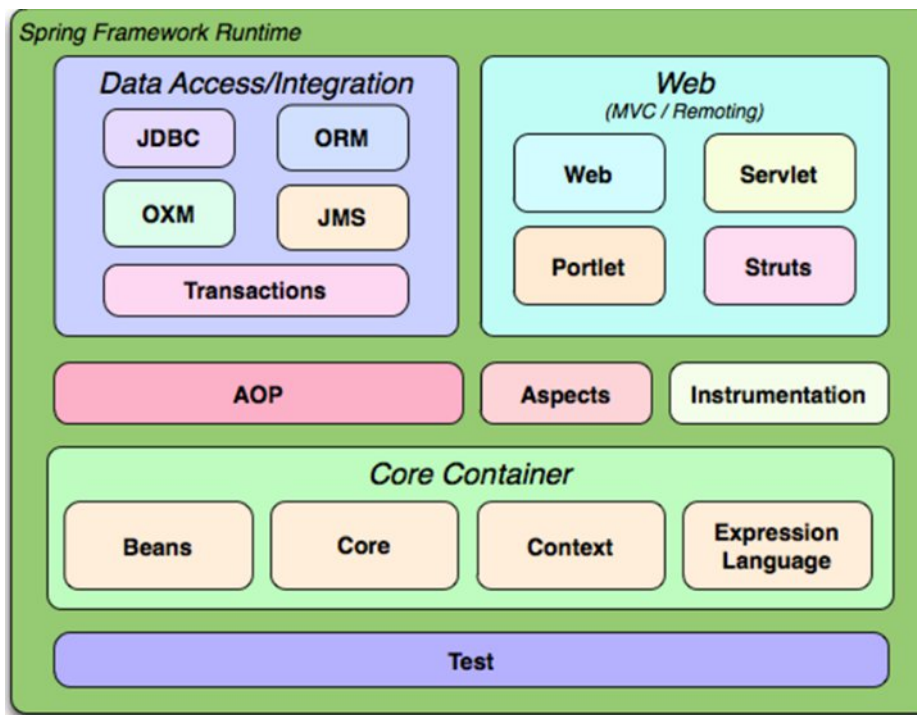
Spring Framework hoạt động dựa trên các nguyên lý và thành phần chủ yếu như Dependency Injection (DI), Aspect-Oriented Programming (AOP), và Inversion of Control (IoC) để hỗ trợ xây dựng các ứng dụng linh hoạt, dễ bảo trì và mở rộng:

- 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
- Dependency Injection – DI (Chèn khía cạnh) : là cốt lõi của Spring Framework 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.
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.

4. Các tính năng của Spring Framework

- Kiến trúc tổng thể của Spring Framework:



Hình 1: Kiến trúc tổng thể của SpringFramework

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.
- 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.

5. 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 dài từ đầu cho mỗi dự án. Điều này, đến lượt nó, 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 nỗ lực và cấu hình.

6. Khi nào thì nên sử dụng SpringFramework:

Spring Framework có khả năng cung cấp khả năng kiểm soát chi tiết đối với các cấu hình. Kiến trúc mô đun, khả năng kiểm soát chính xác đối với các cấu hình và khả năng tích hợp khiến nó trở thành lựa chọn tối ưu cho các dự án yêu cầu phương pháp tiếp cận tùy chỉnh. Các trường hợp có thể dùng:

- Ứng dụng web (Spring MVC và Spring WebFlux).
 - API RESTful (Spring MVC hoặc Spring WebFlux).
 - Phát triển các ứng dụng không có máy chủ và dựa trên Microservices (Spring Boot, Spring Cloud).
 - Xử lý dữ liệu (Spring Batch).
 - Tích hợp và nhắn tin (Spring Integration).
 - Xử lý và phân tích dữ liệu (Spring Data và Spring Cloud Data Flow).
 - Ứng dụng doanh nghiệp (Spring AOP, DI và container management).
 - Ứng dụng gốc trên nền tảng đám mây (Spring Boot and Spring Cloud tools).
 - Phát triển các ứng dụng có yêu cầu bảo mật cao ở phía máy chủ.
 - Phát triển phần mềm không đồng bộ.
 - Tạo ra một kiến trúc hướng sự kiện.
 - Phát triển hàng loạt cho quy trình tự động hóa.
- Cho dù bạn đang phát triển một ứng dụng web đơn giản, một giải pháp doanh nghiệp phức tạp hay khám phá các công nghệ mới như microservices và IoT, Spring Framework cung cấp các công cụ và mẫu cần thiết để hỗ trợ các loại ứng dụng khác nhau. Tuy nhiên, Spring Framework ngày càng trở nên phức tạp do phải xây dựng thêm chức năng. Điều này đòi hỏi một quá trình cấu hình dài trong một dự án mới. Vậy làm thế nào bạn có thể làm cho quá trình này dễ dàng hơn? Hãy cùng xem xét kỹ hơn về **Spring Boot**.

III. 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.



Hình 2: Spring Boot

2. Nguồn gốc ra đời

- Spring Boot được phát triển bởi Pivotal Software, với phiên bản đầu tiên (Spring Boot 1.0) được phát hành vào tháng 4 năm 2014. Mục tiêu ban đầu của Spring Boot là giải quyết những hạn chế và phức tạp trong việc cấu hình Spring Framework truyền thống. Spring Framework, mặc dù rất mạnh mẽ và linh hoạt, nhưng yêu cầu nhiều cấu hình thủ công và có thể trở nên cồng kềnh khi xây dựng các ứng dụng phức tạp.
- Trước khi Spring Boot ra đời, việc cấu hình một ứng dụng Spring thường đòi hỏi nhiều file XML hoặc các class Java cấu hình, gây khó khăn cho việc bảo

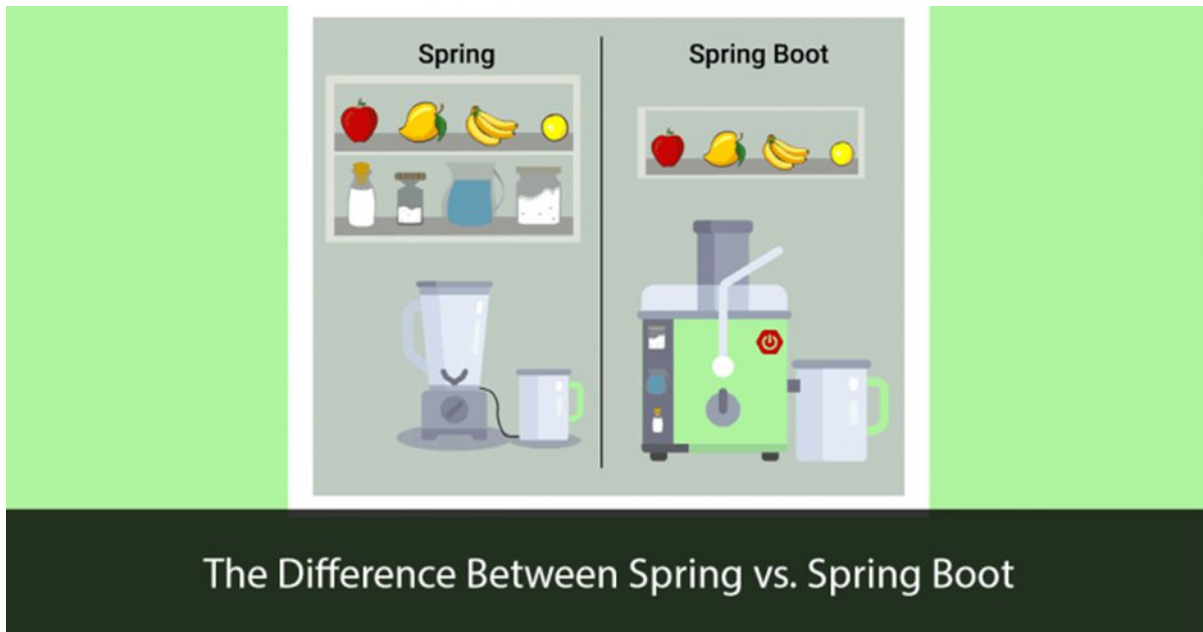
trì và phát triển. Spring Boot được giới thiệu như một cách tiếp cận đơn giản và hiệu quả hơn, với các cấu hình tự động và các công cụ hỗ trợ như Spring Initializr, giúp nhà phát triển bắt đầu dự án mới nhanh chóng chỉ với một vài cú nhấp chuột.

- Từ khi ra mắt, Spring Boot đã nhanh chóng trở thành một trong những framework phổ biến nhất trong cộng đồng Java, được sử dụng rộng rãi trong nhiều loại ứng dụng khác nhau, từ các ứng dụng web đơn giản đến các hệ thống microservices phức tạp. Sự phát triển và cải tiến liên tục của Spring Boot đã giúp nó trở thành một phần không thể thiếu trong hệ sinh thái Spring, giúp các nhà phát triển xây dựng các ứng dụng hiện đại một cách dễ dàng và nhanh chóng hơn.

3. Sự khác nhau giữa Spring Boot và Spring Framework

- Trong quá trình phát triển ứng dụng Java, việc thực hiện các thao tác với cơ sở dữ liệu thường yêu cầu nhiều công sức và có mức độ phức tạp cao. Đây là một thách thức lớn đối với các lập trình viên, đặc biệt khi phải quản lý kết nối, truy vấn và xử lý dữ liệu một cách hiệu quả. Tuy nhiên, sự ra đời của Spring Boot đã giúp đơn giản hóa đáng kể các tác vụ liên quan đến cơ sở dữ liệu, hỗ trợ lập trình viên phát triển ứng dụng nhanh chóng và hiệu quả hơn.
- Mặc dù Spring Framework mang lại nhiều lợi ích trong việc phát triển ứng dụng Java, song hệ sinh thái rộng lớn và tính phức tạp của nó có thể gây khó khăn cho lập trình viên, đặc biệt là những người mới bắt đầu. Việc nắm vững toàn bộ Spring Framework đòi hỏi nhiều thời gian và công sức. Trước thực tế đó, Spring Boot ra đời như một giải pháp tối ưu, giúp đơn giản hóa quá trình phát triển và triển khai ứng dụng.
- Spring Boot được thiết kế với mục tiêu giảm thiểu các cấu hình thủ công và cung cấp các thiết lập mặc định hợp lý, giúp lập trình viên tập trung vào logic nghiệp vụ thay vì xử lý cấu hình phức tạp. Thay vì phải định cấu hình chi tiết như trong Spring truyền thống, Spring Boot cho phép khởi tạo và chạy ứng dụng nhanh chóng chỉ với một số dòng lệnh đơn giản. Bên cạnh đó, Spring

Boot tích hợp sẵn các công cụ hữu ích và tự động hóa nhiều tác vụ quan trọng, giúp lập trình viên nâng cao năng suất và giảm thiểu công sức triển khai ứng dụng.



Hình 3: Sự khác nhau giữa Spring và Spring Boot

- Bên trái - Spring Framework (Dụng cụ pha chế thủ công):
 - Có nhiều nguyên liệu riêng lẻ (trái cây, sữa, đường, đá, v.v.), thể hiện việc lập trình viên cần tự cấu hình nhiều thành phần khi sử dụng Spring.
 - Sử dụng máy xay sinh tố, nhưng phải tự tay bỏ nguyên liệu vào, cắm điện và thực hiện nhiều thao tác thủ công, giống như việc lập trình viên phải cấu hình nhiều bước khi dùng Spring Framework.
- Bên phải - Spring Boot (Máy ép tự động):
 - Cũng có nguyên liệu nhưng ít hơn, thể hiện rằng Spring Boot có các thiết lập mặc định giúp giảm bớt cấu hình thủ công.
 - Sử dụng máy ép hiện đại, chỉ cần bấm nút là có thành phẩm, tượng trưng cho cách Spring Boot tự động hóa nhiều tác vụ như cấu hình máy chủ nhúng, kết nối cơ sở dữ liệu, và quản lý dependency.

4. Các tính năng chính của Spring Boot

- Spring Boot Starter: Spring Boot cung cấp các Starter – tập hợp các dependency được đóng gói sẵn, giúp tích hợp nhanh các tính năng mà không phải cấu hình thủ công. Ví dụ, spring-boot-starter-web hỗ trợ phát triển ứng dụng web, còn spring-boot-starter-data-jpa giúp làm việc với cơ sở dữ liệu. Nhờ đó, lập trình viên không cần tự chọn từng thư viện, tránh lỗi xung đột và tăng tốc phát triển.
- 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.
- Phát triển ứng dụng web: Spring Boot đặc biệt phù hợp để xây dựng các ứng dụng web nhờ tích hợp sẵn các module hỗ trợ. Framework này cho phép tạo máy chủ HTTP độc lập (self-contained HTTP server) bằng cách nhúng các máy chủ như Tomcat, Jetty hoặc Undertow. Với module spring-boot-starter-web, lập trình viên có thể nhanh chóng khởi tạo và vận hành ứng dụng mà không cần cấu hình phức tạp.
- 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.

```
public static void main(String[] args){  
    SpringApplication.run(ClassName.class, args);  
}
```

Hình 4: Minh họa Spring Application

- Sự kiện và Listener trong ứng dụng (Application Events and Listeners): Spring Boot sử dụng cơ chế sự kiện (events) để xử lý nhiều tác vụ khác nhau. Lập trình viên có thể tạo và đăng ký `ApplicationListener` để theo dõi và xử lý các sự kiện trong ứng dụng, giúp quản lý luồng hoạt động một cách linh hoạt.
- 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.

```
spring.application.admin.enabled=1
spring.boot.admin.client.username=admin
spring.boot.admin.client.password=admin
spring.boot.admin.routes.endpoints=env, metrics, trace, jolokia, info,
configprops
spring.boot.admin.notify.hipchat.auth-token=<generated_token>
spring.boot.admin.notify.hipchat.room-id=<room-id>
spring.boot.admin.notify.hipchat.url=https://yourcompany.hipchat.com/v2/
```

Hình 5: Minh họa tính năng hỗ trợ quản trị ứng dụng từ xa

- 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 tắt các tính năng quan trọng:
 - 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 ...







5. Lợi ích 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.

IV. Tổng kết:







- So sánh chi tiết giữa Spring Framework và Spring Boot:

Tiêu chí	Spring Framework	Spring Boot	Khi nào nên sử dụng
Khởi tạo và triển khai ứng dụng	 Cần triển khai dưới dạng WAR và cấu hình máy chủ ứng dụng	 Cho phép triển khai dễ dàng qua JAR, không cần máy chủ bên ngoài.	Spring Framework: Khi cần triển khai ứng dụng trong môi trường cần máy chủ ứng dụng cụ thể (ví dụ: server truyền thống). Spring Boot: Khi cần ứng dụng độc lập và triển khai nhanh chóng mà không cần máy chủ bên ngoài.
Cấu hình	 Cần cấu hình thủ công chi tiết, bao gồm XML hoặc Java Config.	 Tự động cấu hình nhiều thành phần và có các Starter giúp cấu hình dễ dàng.	Spring Framework: Khi cần kiểm soát chi tiết và tối ưu hóa cấu hình (ví dụ: hệ thống phức tạp, yêu cầu tùy chỉnh sâu). Spring Boot: Khi muốn giảm thiểu cấu hình thủ công và đơn giản hóa
Quản lý phụ thuộc	 Cần quản lý thủ công các thư viện phụ thuộc.	 Tự động quản lý phụ thuộc thông qua Spring Dependency Management.	Spring Framework: Khi bạn cần tự chọn các thư viện và kiểm soát phiên bản.

	Quản lý phụ thuộc có thể phức tạp khi có nhiều thư viện và cần phải tự cấu hình.	SpringBoot Starters giúp tự động quản lý và cấu hình phụ thuộc dễ dàng, tránh lỗi xung đột.	soát chặt chẽ việc tích hợp. Spring Boot: Khi muốn giảm thiểu vấn đề về phụ thuộc và sử dụng các thư viện đã được cấu hình sẵn.
Cấu hình tự động	 Cần cấu hình thủ công tất cả các thành phần.	 Tự động cấu hình dựa trên các thư viện có sẵn trong classpath	Spring Framework: Khi cần kiểm soát chi tiết về cấu hình. Spring Boot: Khi muốn ứng dụng hoạt động ngay lập tức mà không cần phải lo lắng về cấu hình thủ công
Khởi tạo ứng dụng	 Khởi tạo ứng dụng cần nhiều công đoạn, phải cấu hình các thành phần riêng biệt.	 Khởi tạo ứng dụng nhanh chóng và dễ dàng thông qua lớp SpringApplication.	Spring Framework: Khi cần triển khai các ứng dụng phức tạp với nhiều yêu cầu đặc biệt về cấu hình. Spring Boot: Khi cần một ứng dụng khởi tạo nhanh chóng mà không mất nhiều thời gian cấu hình.
Quản lý sự kiện và Listener	 Quản lý sự kiện yêu cầu cấu hình thủ công, không tích hợp sẵn.	 Spring Boot tích hợp cơ chế sự kiện và Listener giúp theo dõi các sự kiện trong ứng dụng	Spring Framework: Khi cần kiểm soát chi tiết về các sự kiện và Listener trong ứng dụng. Spring Boot: Khi muốn quản lý sự kiện dễ dàng hơn mà không cần cấu hình phức tạp

Quản trị ứng dụng từ xa	✗ Không tích hợp sẵn tính năng quản trị từ xa.	✓ Hỗ trợ tính năng quản trị từ xa dễ dàng, có thể cấu hình đơn giản.	Spring Framework: Khi ứng dụng yêu cầu kiểm soát phức tạp và tùy chỉnh về quản trị. Spring Boot: Khi cần tính năng giám sát và quản lý từ xa mà không cần cấu hình nhiều
Cấu hình linh hoạt (Externalized Configuration)	✗ Cấu hình cần phải thay đổi mã nguồn nếu cần thay đổi môi trường	✓ Cấu hình linh hoạt với tệp application.properties hoặc YAML, dễ dàng thay đổi cho các môi trường khác nhau.	Spring Framework: Khi cần cấu hình riêng biệt cho các môi trường mà không muốn làm ảnh hưởng đến mã nguồn. Spring Boot: Khi muốn thay đổi cấu hình một cách dễ dàng mà không cần thay đổi mã nguồn
Hỗ trợ YAML	✗ Không hỗ trợ YAML.	✓ Hỗ trợ cấu hình bằng YAML, giúp cấu trúc dữ liệu dễ đọc và dễ quản lý.	Spring Framework: Khi yêu cầu sử dụng cấu hình bằng XML hoặc Java Config. Spring Boot: Khi muốn cấu hình ứng dụng dễ dàng hơn bằng YAML
Bảo mật và quản lý quyền	✓ Tích hợp với Spring Security nhưng yêu cầu cấu hình chi tiết	✓ Tích hợp Spring Security mặc định với Basic Authentication cho tất cả các endpoint HTTP.	Spring Framework: Khi cần kiểm soát chi tiết các chính sách bảo mật và phân quyền. Spring Boot: Khi cần bảo mật ứng dụng ngay từ đầu mà không cần cấu hình

			phức tạp.
Quản lý log	✓ Cần cấu hình thủ công hệ thống log.	✓ Hệ thống logging mạnh mẽ đã được cài đặt sẵn và dễ dàng cấu hình.	Spring Framework: Khi cần cấu hình hệ thống log riêng biệt cho ứng dụng. Spring Boot: Khi cần quản lý log mà không cần phải cấu hình thêm.
Linh hoạt và tùy chỉnh	✓ Cung cấp nhiều khả năng tùy chỉnh sâu.	✗ Ít linh hoạt hơn trong việc cấu hình và tùy chỉnh so với Spring Framework.	Spring Framework: Khi cần khả năng tùy chỉnh chi tiết cho từng thành phần trong ứng dụng. Spring Boot: Khi không yêu cầu quá nhiều tùy chỉnh và muốn một giải pháp nhanh chóng, đơn giản.
Sử dụng trong ứng dụng doanh nghiệp	✓ Phù hợp cho các hệ thống doanh nghiệp phức tạp.	✗ Không hoàn toàn phù hợp cho các hệ thống doanh nghiệp yêu cầu cấu hình phức tạp.	Spring Framework: Khi phát triển các ứng dụng doanh nghiệp phức tạp và yêu cầu khả năng tùy chỉnh cao. Spring Boot: Khi ứng dụng không yêu cầu sự phức tạp trong cấu hình và triển khai.
Phát triển ứng dụng web	✓ Hỗ trợ đầy đủ các tính năng cho ứng dụng web với Spring MVC.	✓ Tích hợp dễ dàng các công cụ hỗ trợ phát triển web.	Spring Framework: Khi cần kiểm soát chi tiết các yêu cầu và cấu hình cho ứng dụng web. Spring Boot: Khi cần phát triển ứng dụng web nhanh chóng và

			đơn giản.
Tính mở rộng	 Rất mạnh mẽ, dễ dàng mở rộng và mở rộng tính năng.	 Mở rộng nhanh chóng, nhưng ít kiểm soát hơn về cấu hình.	Spring Framework: Khi ứng dụng cần tính năng mở rộng cao và khả năng điều chỉnh chi tiết. Spring Boot: Khi cần mở rộng ứng dụng nhanh chóng mà không cần quá nhiều cấu hình.
Tính dễ dàng triển khai và phát triển	 Cần nhiều cấu hình thủ công và xử lý môi trường phức tạp.	 Giảm thiểu công việc cấu hình và triển khai, giúp tiết kiệm thời gian phát triển.	Spring Framework: Khi yêu cầu tùy chỉnh môi trường và kiểm soát chi tiết về cấu hình. Spring Boot: Khi muốn phát triển nhanh và triển khai dễ dàng, tiết kiệm thời gian cho các dự án nhỏ và trung bình.
Quản lý giao dịch và cơ sở dữ liệu	 Hỗ trợ quản lý giao dịch mạnh mẽ và tích hợp với các công nghệ cơ sở dữ liệu.	 Hỗ trợ quản lý giao dịch và tích hợp dễ dàng, nhưng ít tùy chỉnh.	Spring Framework: Khi cần kiểm soát giao dịch và cơ sở dữ liệu ở mức độ chi tiết. Spring Boot: Khi cần triển khai nhanh các tính năng cơ bản của cơ sở dữ liệu và giao dịch mà không cần cấu hình phức tạp.

NGUỒN TÀI LIỆU THAM KHẢO:

ItViet : <https://itviec.com/blog/spring-framework-la-gi/>

Spring.io <https://docs.spring.io/spring-framework/reference/overview.html>

Wikipedia https://vi.wikipedia.org/wiki/Spring_Framework

TopDev <https://topdev.vn/blog/gioi-thieu-ve-spring-boot-spring-boot-la-gi/>

HocSpringBoot <https://hocspringboot.net/2020/08/29/spring-boot-la-gi/>

FptShop <https://hocspringboot.net/2020/08/29/spring-boot-la-gi/>