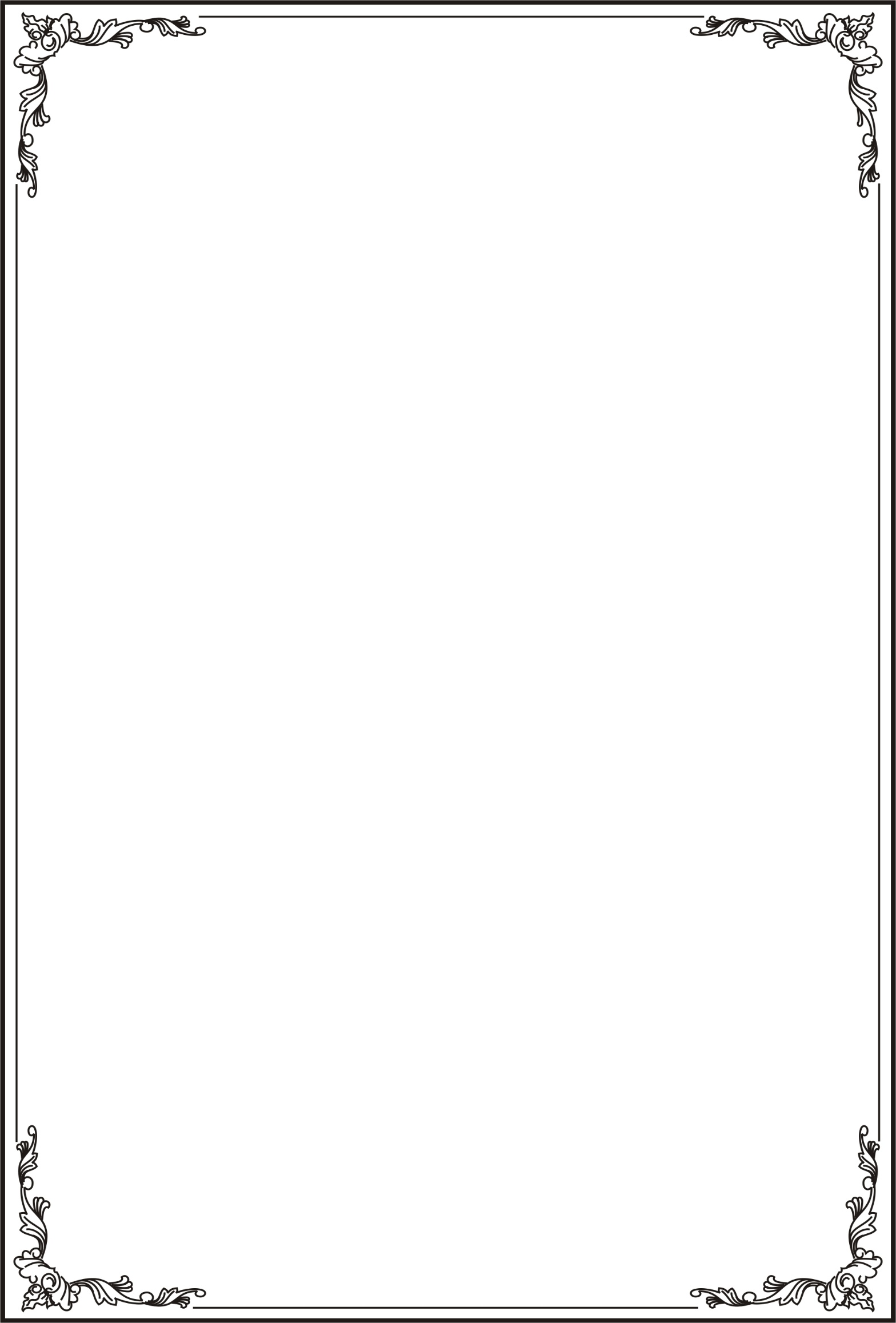
**TRƯỜNG ĐẠI HỌC QUY NHƠN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO**

**SPRING FRAMEWORK**

**Môn học :** Công nghệ Java

**Giảng viên :** Thầy Bình

ádasdas

* **Nhóm Spring** 
  + Nguyễn Hữu Hòa
  + Nguyễn Thái Dương

Quy Nhơn, 10/2019

Nội dung

[**I.** **Tổng quan về Framework** 3](#_Toc23285406)

[**1.** **Khái niệm framework** 3](#_Toc23285407)

[**2.** **Vai trò framework** 3](#_Toc23285408)

[**3.** **Một số framework tiêu biểu** 4](#_Toc23285409)

[**II.** **Giới thiệu Spring Framework** 4](#_Toc23285410)

[**1.** **Lịch sử sơ bộ Framework Spring** 4](#_Toc23285411)

[**a.** **Hoàn cảnh** 4](#_Toc23285412)

[**b.** **Sự ra đời Framework Spring** 5](#_Toc23285413)

[**c.** **Lịch sử phát triển** 5](#_Toc23285414)

[**2.** **Khái niệm Framework Spring** 6](#_Toc23285415)

[**3.** **Các phiên bản Framework Spring** 7](#_Toc23285416)

[**4.** **Nguyên tắc hoạt động** 7](#_Toc23285417)

[**a.** **Dependency Injection (DI)** 7](#_Toc23285418)

[**b.** **Aspect Oriented Programming (AOP)** 13](#_Toc23285419)

[**5.** **Cấu trúc Framework Spring** 14](#_Toc23285420)

[**6.** **Các dự án tiêu biểu trong Spring Framework** 16](#_Toc23285421)

[**7.** **Spring MVC** 16](#_Toc23285422)

[**8.** **Lợi ích của Spring** 18](#_Toc23285423)

[9. Khuyết điểm 19](#_Toc23285424)

[**III.** **Demo một ứng dụng Framework Spring** 19](#_Toc23285425)

[**IV.** **Các nguồn tham khảo** 20](#_Toc23285426)

1. **Tổng quan về Framework**
   1. **Khái niệm framework**

Framework là một khung hoặc giàn giáo cung cấp các chức năng, giải pháp được cài đặt sẵn giúp tiết kiệm thời gian trong quá trình phát triển ứng dụng.

Nói dễ hiểu, framework giống như khung một ngôi nhà , nó cung cấp nền tảng vững chắc và cơ bản để bạn có thể xây dựng được đa số kiểu nhà được thiết kế trong bản vẽ.

Thực tế bạn có thể xây dựng hoàn chỉnh 1 ngôi nhà ngay từ đầu, nhưng nếu bạn bắt đầu với một dàn khung sẽ giúp tiết kiệm được tối đa công sức cho việc bắt đầu.

Framework có 2 thành phần chính là Common Language Runtime (CLR) và NET Framework class library. CLR là một chương trình viết bằng .NET, không được biên dịch ra mã máy mà nó được dịch ra một ngôn ngữ trung gian Microsoft Intermediate Language (MSIL). Khi chạy chương trình, CLR sẽ dịch MSIL ra mã máy để thực thi các tính năng. Còn NET Framework class library cung cấp thư viện lập trình như cho ứng dụng, cơ sở dữ liệu, dịch vụ web...

Framework trong lập trình ứng dụng cơ bản cũng có cấu trúc như vậy, nó cung cấp đa số các kiểu mẫu thiết kế phù hợp với ứng dụng bạn sắp thực hiện, các thư viện, API, trình biên dịch và hơn thế nữa để hợp lý hóa việc tạo ra phần mềm mới.

Biết được framework nào để sử dụng cho dự án là một kỹ năng quan trọng đối với bất kỳ nhà phát triển nào

* 1. **Vai trò framework**

Trong Framework bất kỳ bạn có thể

* **Đảo ngược kiểm soát ( Inversion of control ) :** tức là trong 1 framework các *luồng thông tin* và cách thức *xử lý dữ liệu* được chính Framework đó quy định mà không phải do người lập trình viên quy định. Lập trình viên chỉ có thể thêm thắt các xử lý logic trên các mô hình Framework cung cấp
* **Khả năng mở rộng ( Extensibility )**: Người dùng framework có thể mở rộng bằng cách kế thừa những gì mà framework đang cung cấp và mở rộng nó bằng việc ghi đè có chọn lọc lên các lớp, các phương thức. Lập trình viên có quyền viết thêm chức năng mới cho nền tảng này, miễn là nó hợp lệ và đúng chuẩn.
* **Mã nguồn framework hạn chế sủa đổi (Non – modifiable framework code):**  Hầu như việc trực tiếp sửa đổi lên mã nguồn của framework là 1 việc hoàn toàn không nên thực hiện. Thay vì sửa đổi, có thể thêm các extension mở rộng, các chức năng của framework có thể cung cấp được hầu hết các yêu cầu được đặt ra.
  1. **Một số framework tiêu biểu**
* Spring Framework
* Hibermale Framework
* Struct 2 Framework
* JSF Framework
* EJB Framework
* Maven Framework

1. **Giới thiệu Spring Framework**
   1. **Lịch sử sơ bộ Framework Spring** 
      1. **Hoàn cảnh**

Trước khi nói về Framework Spring , không thể nào không nhắc đến EJB. Có thể nói rằng Spring Framework ra đời là nhờ sự thành công của EJB (Enterprise JavaBeans) .

* EJB được tạo bởi 2 gã khổng lồ công nghệ thời điểm bấy giờ là IBM và Sun vào những năm cuối thập niên 90.
* EJB là một kiến trúc phát triển dành cho các ứng dụng doanh nghiệp đòi hỏi tính mạnh mẽ va khả năng mở rộng cao.
* EJB là một thành phần quan trọng của nền tảng java EE.

Khởi đầu, EJB được ca ngợi là một giải pháp tốt xây dựng các ứng dụng phân tán dành cho các doanh nghiệp. Nhưng so với sự nổi tiếng đó thì những chỉ trích về mặt kỹ thuật của EJB.

EJB là một framework lớn với độ phức tạp cao

EJB chỉ tuyệt vời trên bàn giấy, nhưng thực tế đó là một việc khác

Hiện tại, EJB vẫn được phát triển nhưng nỗi thất vọng trong cộng đồng JAVA ngày càng tăng lên. Chính vì vậy, công đồng JAVA lúc bấy giờ vẫn khao khát có 1 nền tảng khác thay thế.

* + 1. **Sự ra đời Framework Spring**

Dưới bối cảnh đó, năm 2002, Spring Framework đã được ra đời bởi *Rod Johnson.* Phiên bản đầu tiên được viết và được xuất bản cùng với quyển sách đầu tay Expert One-on-One JavaEE Design and Development ( Nhà xuất bản Wrox Press, tháng 10 năm 2002 )

Kiến trúc của Spring Framework được ra mắt công chúng lần đầu tiên hồi tháng 6 năm 2003 dưới giấy phép Apchae – phiên bản 2.0. Phiên bản 1.0 đánh dấu mốc thành đạt đầu tiên được xuất bản vào tháng 3 năm 2004 và tiếp đó vào tháng 9 năm 2004, tháng 3 năm 2005.

Việc Spring Frameword ra đời đã khiến cho việc xây dựng ứng dụng doanh nghiệp trở nên đơn giản và dễ dàng hơn, gần như thay thế EJB. Theo đà phát triển đó Spring Framework dần trở thành framwork mã nguồn phổ biến nhất.

Hiện nay, trên 50% các ứng dụng web java hiện sử dụng là Spring

* + 1. **Lịch sử phát triển**

Phần đầu tiên của Spring framwork ban đầu nguyên được Rod Johnson viết vào năm 2000. Vào năm 2001, những mô hình lập trình cho các ứng dụng web được đại đa số sử dụng đều do Java Servlet API và Enterprise Java Bean cung cấp.

Năm 2003, thành lập lập dư án phát tại Sourceforge để phát triển Spring. Sau khi phát triển trên nền tảng ban đầu hơn một năm họ đã phát hành phiên bản đầu tiên (1.0) vào tháng 3 năm 2004

Spring framework đã làm cho những kỹ năng vốn không được biết đến mấy trở nên những kỹ thuật được mọi người ưu chuộng trong một thời gian ngắn ngủi. Một số những kỹ năng nổi tiếng là kỹ năng “ đảo ngược quyền điều khiên” ( Inversion of Control)

Năm 2005, cho thấy mức độ hưởng ứng nâng cao hơn những năm trước nguyên do cũng vì những phiên bản mới được giaoiws thiệu với những cột mốc đáng kể và những tính năng mới được thêm vào. Diễn dàn Spring (Spring Forum) ra đời cuối năm 2004 cũng góp phần không nhỏ trong việc nâng cao tính phổ biến của bộ framework và từ đó đến nay đã trở thành một nguồn thông tin quan trọng, giúp đỡ cho những người sử dụng

Vào tháng 12 năm 2005, hội thảo đầu tiên về Spring Framework đã được tổ chức tại Miami, Florida thu hút 300 nhà phát triển trong 3 ngày và tiếp theo đó là cuộc hội thảo ở Antwerp vào tháng 6 năm 2006, thu hút hơn 400 người.

* 1. **Khái niệm Framework Spring**

Là một một framework phát triển các ứng dụng java, giúp tạo ra các ứng dụng có hiệu năng cao, dễ kiểm thử, tái sử dụng code,..

Chức năng chính của bộ khung này có thể áp dụng cho bất kỳ ưng dụng java nào, tuy vậy, nhiều bản mở rộng dành cho việc xây dựng nền web cũng được phát triển lên nền tảng Java EE. Tuy bộ khung không ấn định lên một cụ thể nào, Spring Framework đã trở nên phổ biến trong cộng đồng java như một phương án song hành, thay thế hoặc thậm chí bổ sung cho mô hình Enterpise JavaBeans (EJB).

Tóm lại, Framework Spring một mã nguồn mở, được phát triển, chia sẽ và cộng đồng người dùng lớn

***Quan điểm của Spring Framework***

* + Đơn giản hóa công việc phát triển thông qua việc sử dụng các POJO ( Plain Old Java Objetc)
  + Nới lỏng ràng buộc giữa các thành phần thông qua việc sử dụng Dependency Injection
  + Giảm thiểu các mã boilerplate thông qua việc sử dụng template và aspect…
  1. **Các phiên bản Framework Spring**

Hiện tại có 5 version :

2002 : version 0.9

2003 : version 1.0

2006 : version 2.0

2009 : version 3.0

2013 : version 4.0

2017 : version 5.0

* 1. **Nguyên tắc hoạt động**

Dựa theo 2 nguyên tắc làDependency Injection và Aspect Oriented Program

* + 1. **Dependency Injection (DI)**

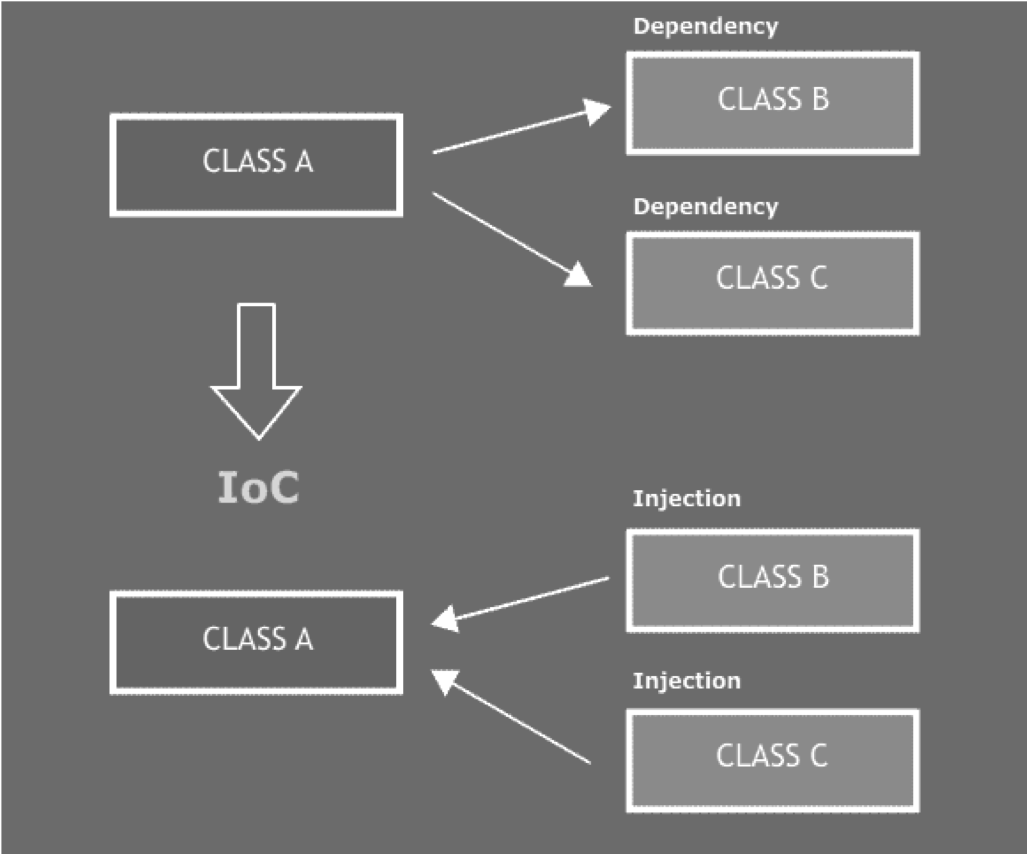
Là thành phần phụ thuộc, là sức mạnh nổi bật của Spring Framework

Là một mẫu thiết kế phần mềm mà các đối tượng phụ thuộc sẽ được inject vào một lớp nào đó

Là một implemetation cụ thể của khái niệm Inversion of Control *( đảo ngược điều khiển )*

*Hay nói cách khác Dependency injection (DI) là một mô hình triển khai từ nguyên lý IoC, là một kỹ thuật trong lập trình trong đó một đối tượng cung cấp những phụ thuộc (dependency - là đối tượng, dịch vụ, chức năng) của đối tượng khác. Injection - Bơm vào (tiêm vào) ám chỉ một phụ thuộc (đối tượng, dịch vụ) đưa vào đối tượng để đối tượng sử dụng nó.*

Hình ảnh mô tả sự hoạt động giữa các class có và không có DI



Ở mô hình không IoC, Class A khi cần chủ động tạo ra đối tượng lớp Class B và Class C (nó nắm quyền khởi tạo, điều khiển)

Với mô hình **IoC** thì class A không tự khởi tạo cũng không chịu trách nhiệm quản lý Class B, Class C. Nó nhận được hai dependency này từ bên ngoài thông qua một cơ chế nào đó (như bằng setter, bằng tham số hàm tạo lớp A, bằng gán thuộc tính ...)

**Nhiệm vụ của dependency injection:**

* Tạo các đối tượng.
* Quản lý sự phụ thuộc (dependencies) giữa các đối tượng.
* Cung cấp (inject) các phụ thuộc được yêu cầu cho đối tượng (được truyền từ bên ngoài đối tượng).

**Nguyên tắc hoạt động của DI:**

* Các module không giao tiếp trực tiếp với nhau, mà thông qua interface. Module cấp thấp sẽ implement interface, module cấp cao sẽ gọi module cấp thấp thông qua interface.
* Việc khởi tạo các module cấp thấp sẽ do DI Container/ IoC Container thực hiện.
* Việc Module nào gắn với interface nào sẽ được config trong file properties, trong file XML hoặc thông qua Annotation. Annotation là một cách thường được sử dụng trong các Framework, chẳng hạn như @Inject với [CDI](http://www.cdi-spec.org/), @Autowired với [Spring](https://spring.io/) hay @ManagedProperty với [JSF](https://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html).

Có 5 dạng **Dependency Injection**

* **Constructor Injection**: Các dependency sẽ được container truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất.
* **Setter Injection**: Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter.
* **Fields/ properties**: Các dependency sẽ được truyền vào 1 class một cách trực tiếp vào các field.
* **Interface Injection**: Class cần inject sẽ implement 1 interface. Interface này chứa 1 hàm tên Inject. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm Inject của interface đó. Đây là cách rườm rà và cũng ít được sử dụng.
* [**Service Locator**](https://gpcoder.com/4946-huong-dan-java-design-pattern-service-locator/): nó hoạt động như một mapper, cho phép thay đổi code tại thời điểm run-time mà không cần biên dịch lại ứng dụng hoặc phải khởi động lại.

Để hiểu rõ hơn về DI, ta cần biết Inversion of Control là gì? Hay nói cách khác chúng ta sẽ tìm hiểu về IoC Contrainer trong Spring

***IoC Contrainer là gì?***

**Ioc** Contrainer trong Spring được xây dựng dựa trên nguyên lý Inversion of Control ( đảo ngược điều khiển ) đã xuất hiện khá lâu trong các mẫu hình thiết kế ( design pattern)

*Control( điều khiển ) trong chương trình là gì?*

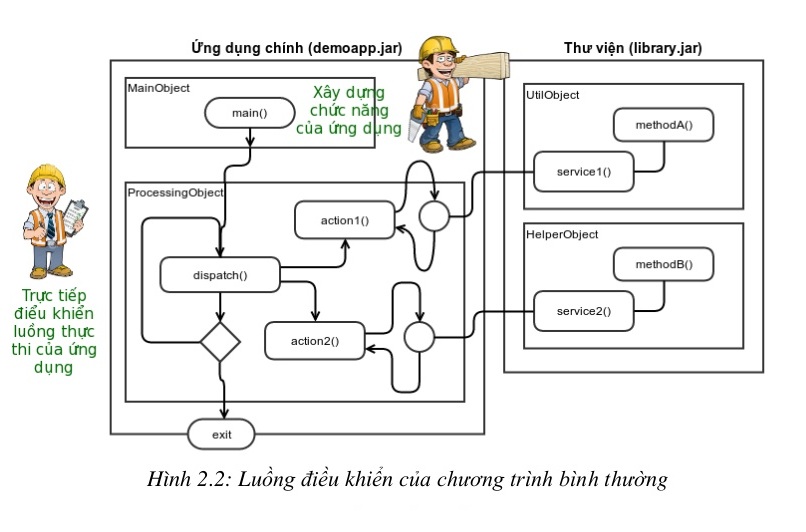
*Inversion ( sự đảo ngược ) đối với điều khiển là gì?*

Khái niệm Control Flow ( tạm dịch là luồng thực thi ) được sử dụng sao cho trình tự thực hiện các câu lệnh, chỉ thị hoặc lời gọi hàm trong 1 chương trình , khi chương trình này thực thi

Các lập trình viên khi xây dựng ứng dụng từ đầu, đã thực hiện 2 nhiệm vụ : trực tiếp điều khiển luồng thực thi chương trình và xây dựng các chức năng để đáp ứng nghiệp vụ của ứng dụng

Việc trực tiếp tạo dựng và kiểm soát thực thi của các chương trình lặp đi lặp lại khi xây dựng nhiều ứng dụng sẽ nhất nhiều công sức, chi phí, tạo ra sự nhàm chát và dễ phát

Để hiểu rõ hơn, xem hình ảnh mô tả về luồng thực thi của 1 chương trình bình thường sau đây và 2 ví dụ ở dưới :

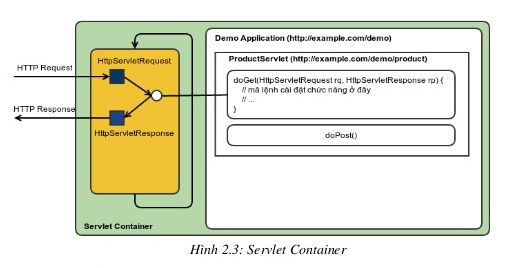
****

Ví dụ 1: Ứng dụng web trong java với JavaSerlet

Sau khi lập trình ứng dụng web JavaSerlet cần thực hiện các bước lập tình như sau:

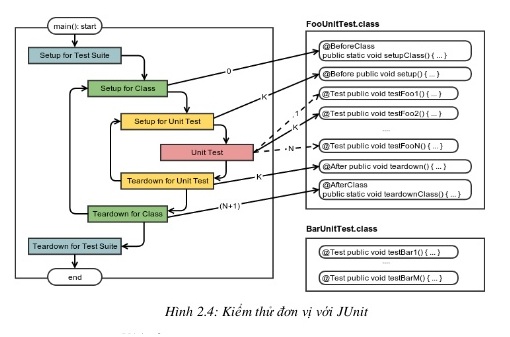
* Tạo lớp đốt tượng kế thừa từ HttpServlet, override các phương thức doGet(), doPost()…
* Đăng ký trong file cấu hình Deployment Descriptor tương ứng Servlet này với đường dẫn xác định
* Lớp đối tượng Servlet chúng ta ạo ra sẽ được gọi đến khi một truy vấn HTTP có đường dẫn đúng với đường dẫn khai báo trong Deployment Descriptor.

Vậy câu hỏi là ? Ai là kích hoạt ứng dụng chúng ta viết để đáp ứng mỗi khi có HTTP Request gửi đến? Ai chịu trách nhiệm chuyển đổi các thông điệp HTTP ( HTTP Request và HTTP Response) thành các đối tượng Java ( HttpServletRequest và HttpServletResponse) để truyền cho các hàm doGet(), doPost() ? Đó chính là Servlet Container.

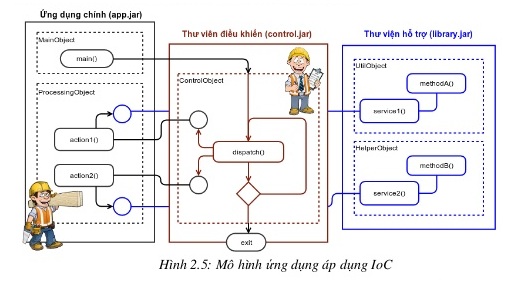


Ví dụ 2: Trong quá rình phát triển các thành phần chức năng của ứng dụng, chúng ta thường áp dụng kiểm thử đơn vị để đảm bảo chức năng đó vẫn chạy đúng trong suốt quá trình ứng dựng được mở rộng và phát triển thêm. Để tạo một bộ unit test, chúng ta chỉ cần tạo một đối tượng, định nghĩa các phương thức khởi tạo, phương thức kết thức và các phương thức test. Sau đó chỉ cần chạy bộ test kiểm thử.

Việc điều khiển trình tự thực thi các phương thức được giao cho thư viện bên ngoài đảm nhiệm chẳng hạn như TestNG hoặc Junit.



Với 2 ví dụ trên, chúng ta nhận thấy trong các ứng dụng đã có sự thay đổi vai trò. Ứng dụng không còn vừa trực tiếp tạo dựng và kiểm soát luồng thực thi, vừa xây dựng chức năng nghiệp vụ. Việc kiểm soát luồng thực thi được tách khổi chức năng nghiệp vụ và bị đẩy ra bên ngoài. Người lập trình đã ủy thắc việc kiểm soát luồng thực thi ứng dụng cho một thành phần ( thường là thư viện dựng sẵn ) bên ngoài đảm nhiệm, chỉ còn tập trung vào chức năng chính của ứng dụng.

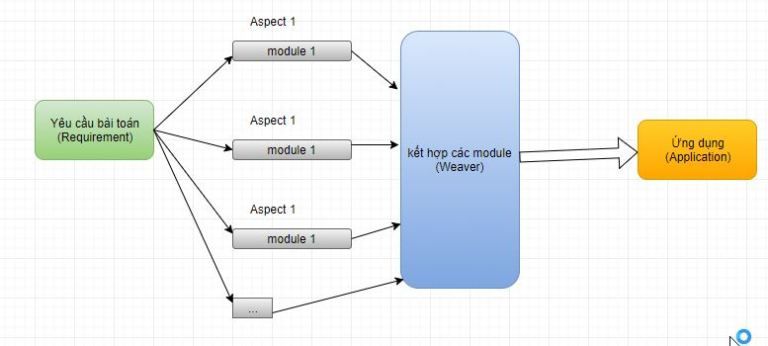


Như vậy, khai niệm “đảo ngược” ở đây chính là chuyển nhiệm vụ kiểm soát lưu trình thực thi từ ứng dụng cho một thành viên chuyên trách (thường là một thư viện phần mềm khung – framework – dựng sẵn bên ngoài). Ứng dụng chính là chúng ta quan tâm phát triển không kiểm soát việc điều khiển luồng thực thi nữa, mà chỉ tập trung vào định nghĩa chức năng. Thư viện ohaanf mềm khung chuyên trách kiểm điều khiển dựa trên mô tả trong cầu hình của ứng dụng để thay mặt ứng dụng điều phối luồng thực thi trong chương trình.

Dễ dàng nhận thấy thư viện phần mềm khung này khác thư viện thông thường ở chỗ: thư viện thông thường cung cấp các chức năng và chờ được ứng dụng gọi đến, còn thư viện phần mềm khung tạo dựng luồng thực thi và gọi đến các chức năng của ứng dụng

Nói một cách ngắn gọn hơn, IoC là một design pattern và tập hợp các kỹ thuật lập trình lên quan, trong đó luồng thực thi của một hệ thống bị đảo ngược so với cách tương tác truyền thống. IoC trong Spring cũng hoàn toàn mang ý nghĩa như trên. Trong Spring các đối tượng chính để xây dựng ứng dụng thì quản lý bởi Spring IoC container. Và IoC container gọi các đối tượng đó là các bean. Một Bean chỉ đơn giản là một đối tượng được khởi tạo và quản lý bởi Spring IoC contrainer

* + 1. **Aspect Oriented Programming (AOP)**

****

**Ví dụ trực quan :**

Một công ty được chia thành nhiều phòng ban, phòng kỹ thuật, phòng kế toán, phòng kinh doanh (mỗi phòng giống như một module trong chương trình)…

Mỗi phòng thực hiện một nhiệm vụ riêng, nhưng cần kết hợp lại để vận hành một công ty.

*Trong AOP có khái niệm “lát cắt”, “điểm cắt”, tạm hiểu là module A sẽ xen vào module B để thực hiện 1 chức năng nào đó mà không làm ảnh hưởng module B, “điểm cắt” là vị trí mà module A xen vào module B*

Ví dụ:

Trong công ty, tất cả các nhân viên đều cần phải quyết toán thuế, nếu từng người 1 thực hiện sẽ tốn rất nhiều thời gian và ảnh hưởng tới công ty. Phòng kế toán sẽ gộp lại và thực hiện quyết toán cho tất cả nhân viên. Bây giờ luật thuế thay đổi, thì cũng chỉ cần mỗi phòng kế toán chỉnh sửa lại cho phù hợp, không ảnh hưởng gì tới các phòng ban khác. (Module phòng kế toán thực hiện lát cắt với các phòng còn lại ở nhiệm vụ quyết toán thuế)

Điểm cắt ở đây tùy vào xếp muốn, luật yêu cầu, ví dụ luật yêu cầu là phải quyết toán thuế cuối năm thì phòng kế toán sẽ thực hiện lát cắt quyết toán thuế cho các phòng ban còn lại vào cuối năm.

* 1. **Cấu trúc Framework Spring**

**Kích thước:** Nhẹ, trong suốt : sersion cơ bản chỉ có 2MB

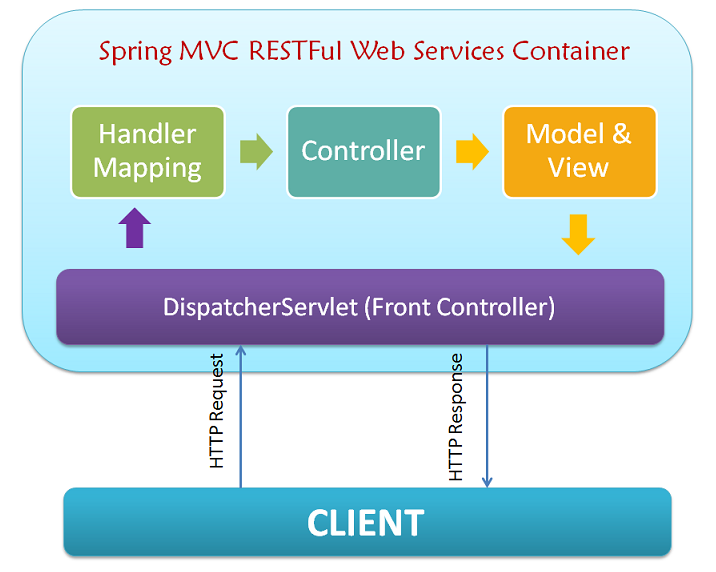
**Mục tiêu :** Giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên *mô hình POJO*

***Kiến trúc tổng quát:***

* **Tầng Test:** Cung cấp khả năng hỗ trợ kiểm thử với *Junit và TestNG*
* **Tầng Core Container:** Là thành phần cốt lỗi của Spring Frameword. Là nền tảng xây dựng nên các thành phần khác trong hệ sinh thái Spring Framework Có các module spring core, beans, context và expression languate.
  + **Spring beans**: là trung tâm của Spring Core. Là trái tim của một ứng dụng Spring. *Tương đồng với nguyên lý thiết kế “ Đơn trách nhiệm “ (single resoinsibility)* của Robert C Martin trong lập trình hướng đối tượng. Cung cấp tính năng IOC và Dependency Injection
  + **Spring Context :** Hỗ trợ đa ngôn ngữ, các tính năng Java như EJB, JMX. Mang mọi thứ lại với nhau.Kết thừa các tính năng của Spring Bean và bổ sung các hỗ trọ cho internationalization ( ví dụ như các resource bundle). Event propagation, resource loading,….Hỗ trợ các tính năng của Java EE như EJB, JMX, RMI.Interface ApplicationContext chính là trọng tâm của Spring Context
  + **Expression Language** được mở rộng từ Expresion Languge trong JSP, cung cấp hỗ trợ việc setting/getting giá trị, các method cải tiến cho phép truy cập collections, index, các toán tử logic,.. Là một ngôn ngữ ngắn gọn giúp cho việc cấu hình Spring Framwork trở nên linh hoạt hơn
* **Tầng AOP, Aspects and Instrumentation:** Những moduel này hỗ trợ cài đặt lập trình hướng khía cạnh ( Aspect Oriented Programing), hỗ trợ tích hợp với Aspectj. Trong các ứng dụng lớn, có một số tính năng chung có thể được áp dụng cho hầu hết các module trong hệ thống như logging, transaction, data validation, authentication, ... Trong Object Oriented Programming, các module của ứng dụng được thực hiện bởi các class, các module này tương tác với nhau thông qua các hành vi của các object. Tuy nhiên, các hành vi đan xen lẫn nhau của các modile trên thì OOP lại không thể hỗ trợ được, dẫn đến tính nhất quán của ứng dụng bị thay đổi, khó kiểm soát. Để khắc phục được nhược điểm này, AOP đã được phát triển. AOP là cho phép thực hiện các vấn đề riêng biệt một cách linh hoạt và kết hợp chúng lại để tạo nên hệ thống. AOP bổ sung cho OOP bằng việc hỗ trợ một dạng module khác, cho phép "kéo" thể hiện chung của vấn đề đan nhau vào một khối. Khối này được gọi là ‘aspect’ (tạm dịch là ‘lát’ – hàm ý lát cắt đi qua nhiều lớp object, từ chữ ‘aspect’ này chúng ta có tên của phương pháp phát triển phần mềm mới: aspect-oriented programming  - AOP. Nhờ source code được tách riêng, vấn đề đan nhau trở nên dễ kiểm soát hơn. Các aspect của hệ thống có thể thay đổi, thêm hoặc xóa lúc build và có thể tái sử dụng. Một dạng buid đặc biệt có tên là Aspect Weaver thực hiện kết hợp các thành phần riêng lẻ lại thành hệ thống hợp nhất.
* **Tâng Data Access/ Integration:** Nhóm này bao gồm JDBC, ORM, OXM, JMS và moduel Transaction. Cung cấp khả năng giao tiếp với database
* **Tầng Web:** Hỗ trợ việc tạo ứng dụng web
  1. **Các dự án tiêu biểu trong Spring Framework**
* **Spring MVC:** dành cho việc xây dựng các ứng dụng nền tả web
* **Spring Data:** Cung cấp 1 cách tiếp cận đúng đắn để truy cập dữ liệu từ cơ sở dữ liệu quan hệ, phi quan hệ, map-reduce và thậm chí còn hơn thế nữa
* **Spring Security**: Cung cấp cơ chế xác thực ( authentication ) và phân quyền (authorization) cho ứng dụng
* Spring boot: là một framwork giúp chúng ta phát triển cũng như chạy ứng dụng 1 cách nhanh chóng
* **Spring Batch:** Giúp dễ dàng tạo các lịch trình (scheduling) và tiến trình (processing) cho các công việc xử lý theo mẻ (batch job)
* **Spring integration:** Là một implemetation của Enterprise Integration Patterns ( EIP). Thiết kế một kiến trúc hướng thông điệp hỗ trợ việc tích hợp các hệ thống bên ngoài
* **Spring Socail:** Giúp kết nối ứng dụng của bạn với API bên thứ ba của facebook, twitter, Linkedin,,,
  1. **Spring MVC**

Spring MVC là một framework hiện thực [mô hình MVC](https://huongdanjava.com/vi/mo-hinh-mvc.html) trong ứng dụng web

**Cơ chế hoạt động Request và Response**

****

**- Model:** đóng gói dữ liệu ứng dụng và bao gồm các POJO.

**- View:** Chịu trách nhiệm nhận giá trị của model và vẽ ra trang HTML mà trình duyệt có thể hiển thị được.

**- Controller:** Chịu trách nhiệm nhận và xử lý các yêu cầu từ người dùng và tạo các model phù hợp và trả về cho view.

**- DispathcherServlet** là thành phần tiếp nhận tất cả các request từ browser, điều khiển luồng xử lý và trung chuyển giữa các thành phần (components) trong MVC.

**- Handler Mapping:** Nơi lưu các ánh xạ đường dẫn và chỉ dẫn tùy theo các Request

**Cách hoạt động :** Từ Client , người dùng gửi yêu cầu Request nào đó, khi đó DispatcherServlet sẽ tiếp nhận. Sau đó nó sẽ hỏi tới Handler Mapping , đạikhái là : **“**Người dùng gửi yêu cầu như thế thì mình làm gì Handler nhỉ” – Handler said.” Lúc đó thằng Handler sẽ lấy thông tin từ Request và ánh xạ đúng đường dẫn đến Controller tương ứng. Khi đó Controller ( nơi xử lý các yêu cầu Logic bussion này liên lạc với Model ( lớp này thường là truy cập vào cơ sở dữ liệu ) kéo dữ liệu ra và trả về dưới dạng ModelAndView object cho DispatcherServlet.

DispatcherServlet gửi gói ModelAndView cho ViewResolver để tìm xem trang web (JSP) nào sẻ được load lên.

DispatcherServlet trả về response cho người dùng và hiển thị lên View.

*Các bạn có để ý rằng : Mọi hoạt động đều phải thông qua DispatcherServlet*

* 1. **Lợi ích của Spring**

Spring cho phép lập trình viên sử dụng POJOs (**plain old Java objects :** POCO ko nên có bất kì nhiệm vụ nào khác ngoài việc lưu trữ và xử lý dữ liệu (trên bộ nhớ).Điều này đơn giản là tuân theo nguyên lý Single responsibility trong việc thiết kế hướng đối tượng (Mỗi lớp chỉ nên đảm nhận duy nhất và trọn vẹn một trách nhiệm/công việc ).). . Việc sử dụng POJOs giúp bạn không phải làm việc với EJB, ứng dụng, các luồng chạy, cấu hình ….. đơn giản đi rất nhiều

* + Spring được tổ chứ theo kiểu moodun. Số lượng các gói và lớp khá nhiều, nhưng bạn chỉ cần quan trâm đến những gì bạn cần và không quan tâm đến phần còn lại
  + Spring hỗ trợ sử dụng khá nhiều công nghệ như ORM Framework, các logging framework, JEE,, các thư viện tạo lịch trình (Quartz và JDK timer )….
  + Module Web của Spring được thiết kế theo mô hình MVC nên nó cung cấp đầy đủ các tính năng giúp thay thế các web framework khác như Struts
  + Lightweight và very little overhead trong việc phát triển ứng dụng của bạn.
  + Dependency Injection hoặc Inversion of Control được sử dụng để giúp các component tách rời, độc lập với nhau. Spring container sẽ giúp gắn kết những components này lại với nhau theo đặc tả business của bạn.
  + Spring IoC container quản lý vòng đời của Spring Bean và các cấu hình của project chẳng hạn như JNDI lookup.
  + Spring MVC framework được sử dụng cho phát triển ứng dụng web rất dễ dàng với việc hỗ trợ rất tốt các tính năng web services, json,…
  + Hỗ trợ quản lý transaction, JDBC operations, File uploading, Exception Handling,… rất dễ dàng bằng cách cấu hình được rút gọn, thay vào đó là sử dụng annotation hoặc spring bean configuration file.
  + Và một vài ưu điểm khi sử dụng Spring framework:
  + Làm giảm đi sự phụ thuộc giữa các components khác nhau của ứng dụng, Spring IoC container làm nhiệm vụ khởi tạo resources hoặc beans và “tiêm – inject” chúng theo sự phụ thuộc khác nhau.
  + Thực hiện unit test case rất dễ bởi vì business logic của bạn không có sự phụ thuộc trực tiếp. Việc thực hiện chỉ là viết test configuration và inject mock bean cho các mục đích test khác nhau.
  + Làm giảm đi khối lượng code rất nhiều, chẳng hạn như việc khởi tạo đối tượng, open/close các resources,…
  + Spring framework chia thành nhiều module riêng biệt, do đó việc sử dụng các features trong Spring framework rất tự do… Ví dụ như ứng dụng không sử dụng tính năng transaction, thì không cần thiết phải thêm dependency này vào.

Spring framework hỗ trợ hầu hết các tính năng của Java EE, thậm chí còn nhiều hơn nữa.

* 1. Khuyết điểm

***Nhược điểm***: các khớp nối mô-đun vẫn còn lỏng lẻo và cần cải thiện.

1. **Demo một ứng dụng Framework Spring**
2. **Các nguồn tham khảo**

Các nguồn thông tin từ các Website :

<https://viblo.asia/p/tong-quan-ve-spring-framework-YWOZryEyKQ0>

<https://viblo.asia/p/aop-la-gi-uu-nhuoc-diem-cua-aop-aspect-oriented-programming-trong-java-3P0lPk2oZox>

<https://o7planning.org/vi/10129/huong-dan-lap-trinh-spring-mvc-cho-nguoi-moi-bat-dau>

<https://en.wikipedia.org/wiki/Spring_Framework>

<https://stackjava.com/spring/spring-aop-1.html>

<https://xuanthulab.net/nguyen-ly-lap-trinh-ioc-inversion-of-control-ioc.html?fbclid=IwAR1F6kPRCsbfGPj5gMolIDOkRJo3dbmJmcCkznqMgfqjL8a6kiIuR67wXvk>

<https://huongdanjava.com/vi/spring-mvc>

<https://www.codehub.vn/Tim-Hieu-Ve-Dependency-Injection-Qua-Vi-Du-Cu-The>

<https://viblo.asia/p/dependency-injection-la-gi-va-khi-nao-thi-nen-su-dung-no-LzD5d0d05jY>

<https://stackoverflow.com/questions/42863382/java-io-filenotfoundexception-class-path-resource-cannot-be-opened-because-it-d/51100124>

<https://loda.me/spring-giai-thich-dependency-injection-di-va-io-c-bang-ngoc-trinh-loda1553326013583/>

<https://huongdanjava.com/vi/inversion-control-va-dependency-injection.html>