Vậy, Spring là gì?

[**Spring**](https://www.bitdegree.org/course/spring-cloud) là một khung framework mã nguồn mở, dựa trên Java. Nó được thiết kế để giúp các nhà phát triển Java tạo các ứng dụng web cụ thể.

**Câu hỏi 2: Có bao nhiêu phiên bản Spring?**

Tổng cộng, có **ba** phiên bản chính của Spring - **Spring 2.5; Spring 3.0** và **Spring 4.0**.

### Câu hỏi 3: Kể tên những lợi thế chính của Spring

**open-source** (mã nguồn mở), **premade templates** (các mẫu có sẵn), **dễ để sử dụng** và **kiểm tra**, v.v.

### Câu hỏi 4: Những tính năng của Spring Framework là gì?

Java Spring có ít nhất một số tính năng chính sau:

* Nhẹ khi nói về dung lượng (size) và tính minh bạch (transparency).
* Trong Spring framework, các đối tượng cung cấp các phụ thuộc của chúng thay vì tạo hoặc tìm kiếm các đối tượng phụ thuộc. Quá trình này được gọi là Inversion of Control (IOC).
* Lập trình hướng theo khía cạnh trong Spring framework hỗ trợ phát triển gắn kết bằng cách tách logic kinh doanh ứng dụng khỏi các dịch vụ hệ thống.
* Spring framework tạo và quản lý vòng đời và cấu hình của các đối tượng ứng dụng.
* Khung framework ứng dụng web Model-View-Control trong Java Spring có cấu hình cao. Ngoài ra, nó cho phép sử dụng các khung framework khác thay vì nó.
* Spring framework cung cấp một lớp trừu tượng chung để quản lý giao dịch.
* Spring cung cấp lớp trừu tượng kết nối cơ sở dữ liệu Java để đơn giản hóa chiến lược xử lý lỗi.

### Câu hỏi 5: 'MVC' là gì?

**MVC** là viết tắt của Model View Controller. Nó là một phần của Spring framework. Học Spring MVC để sử dụng cho việc tiếp tục tạo ra các ứng dụng web khác nhau.

### Câu hỏi 6: ‘dependency injection’ là gì?

**Dependency injection** được sử dụng để cung cấp các phụ thuộc cụ thể nhất định cho các đối tượng. Đó là một mẫu thiết kế làm cho các dự án mượt mà hơn và phù hợp hơn cho các hoạt động như thử nghiệm.

### Câu hỏi 7: ‘Boot’ là gì?

**Spring Boot** là một phiên bản của Spring mà mục đích để làm cho ứng dụng tạo ra quá trình sắp xếp hợp lý hơn. Một trong những tính năng chính của nó là loại bỏ nhu cầu xác định cấu hình boilerplate (bản mẫu) - chắc chắn, điều này sẽ khiến rất nhiều nhà phát triển thích thú khi học Spring boot hay sử dụng nó.

### Câu hỏi 8: ‘XML file’ là gì?

**XML** là một tệp cấu hình Spring. Đây là loại tệp chính chứa tất cả thông tin về các loại lớp khác nhau và cách chúng tương quan với nhau.

### Câu hỏi 9: Spring có bao nhiêu mô-đun khác nhau?

Spring có khoảng **hai mươi** mô-đun và đây là thông tin rất quan trọng cần biết trong các câu hỏi phỏng vấn Java Spring. Tất cả chúng được phân thành các container (vùng chứa) riêng biệt. Các container, như sau: **core** container, **web** container, **data** access container, **aspect** và  **instrumentation** container, và  **test** container. Cũng có một vài container nhỏ hơn.

### Câu hỏi 10: 'IOC' là gì?

**IOC** là viết tắt của *Inversion of Control*. Nó là core container của Java Spring. Nó sử dụng phép *dependency injection* được đề cập trước đó để quản lý và định cấu hình các ứng dụng tích hợp khác nhau. Hiện tại có **hai** loại IOC có thể được đặt trong Java Spring - **ApplicationContext** và **BeanFactory**.

### Câu 11: ‘bean’ là gì?

“**Bean**” là *một đối tượng được tích hợp và cấu hình bởi IOC container*.

### Câu hỏi 12: Tại sao nên sử dụng ‘constructor injection’ cho injecting bean?

Đừng để bị lừa bởi những câu hỏi phỏng vấn Java Spring như vậy - lưu ý rằng constructor injection chỉ được sử dụng để inject mandatory dependencies.

Lý do tại sao nên sử dụng loại injection này đơn giản là vì kiểu injection này làm cho quá trình kiểm tra sau đó dễ dàng hơn rất nhiều.

### Câu hỏi 13: ‘join point’ là gì?

**Join point** thể hiện một điểm trong chương trình nơi khung AOP sẽ được khởi tạo và áp dụng.

### Câu hỏi 14: AOP là gì?

**AOP** (Aspect-Oriented Programming) là viết tắt của *Lập trình hướng theo khía cạnh*. Nó khác với OOP (*Object-Oriented Programming - Lập trình hướng đối tượng*) ở chỗ OOP tập trung vào **các lớp**, trong khi đơn vị mô đun chính của AOP là **khía cạnh**. Trong AOP, các khía cạnh thực hiện và nhấn mạnh mối liên quan xuyên suốt.

### Câu hỏi 15: Boot 'initilizr' sử dụng làm gì?

'**Initializr**' là một chủ đề có xu hướng xuất hiện thường xuyên trong các câu hỏi phỏng vấn Java Spring. Đây là một thành phần khá quan trọng, vì vậy hãy nhanh chóng tổng hợp lý do tại sao chúng ta cần sử dụng.

Các nhà phát triển sử dụng '**initilizr**' vì họ muốn tối đa hóa năng suất trong khoảng thời gian ngắn nhất có thể. Công cụ này tuyệt vời ở chỗ nó cung cấp một khuôn mẫu ban đầu cho dự án và đơn giản hóa quá trình xây dựng tập lệnh.

### Câu hỏi 16: ‘advice’ là gì?

Hãy nhớ khi chúng ta nói về ‘*join points’*? Những nơi mà AOP được khởi tạo? Bất cứ khi nào một khía cạnh thực hiện một số loại hành động trong joinpoint đó, thì hành động đó được gọi là một “**advice**”.

**Câu hỏi 17: Có bao nhiêu cách để thực hiện Dependency Injection?**

Có ba cách để làm điều đó:

* Constructor Injection
* Setter Injection
* Interface Injection

Trong Spring framework, chủ yếu là constructor và setter injection được sử dụng.

**Câu hỏi 18: Các cách sử dụng Spring Framework là gì?**

* Dùng như một khung framework trang web của bên thứ ba bằng cách sử dụng Spring framework ở giữa.
* Để sử dụng từ xa.
* Dùng như một ứng dụng web Spring chính thức.
* Dùng như một Bean Java Enterprise có thể bao bọc các Plain Old Java Object hiện có. (POJO)

**Câu hỏi 19: AOP (Aspect oriented programming- lập trình hướng khía cạnh) là gì?**

* JoinPoint (Điểm chương trình)
* Advice (Xử lý phụ)
* Pointcut (Điểm cắt - để xác định JoinPoint)
* Aspect (Khía cạnh, tương tự như lớp trong Java)
* Introduction (Giới thiệu)
* Target Object (Đối tượng mà advice áp dụng)
* Interceptor (Can thiệp)
* AOP Proxy
* Weaving (Quá trình liên kết thành phần aspect và non-aspect)

### Câu hỏi 20: JoinPoint là gì?

Đó là bất kỳ điểm nào trong một chương trình như trường truy cập, thực thi phương thức, xử lý ngoại lệ hoặc tương tự.

### Câu hỏi 1:Sự khác biệt giữa hai loại IOC là gì?

Nếu bạn còn nhớ, một trong những câu hỏi phỏng vấn Java Spring cơ bản là giải thích IOC. Hai loại IOC là ApplicationContext và BeanFactory.

Có sự khác biệt chính giữa hai loại IOC - ApplicationContext là container **nâng cao**, trong khi BeanFactory là một loại **cơ bản**. Điều này có nghĩa là các chức năng của BeanFactory được tiếp tục bởi ApplicationContext.

### Câu hỏi 2: Tại sao ‘annotation wiring’ không hoạt động?

Đây có lẽ sẽ là một trong những câu hỏi mà bạn sẽ được đưa ra một tình huống và phải giải thích nó. Ở đây, nhà tuyển dụng đang chỉ ra rằng chức năng ‘**annotation wiring**’ không hoạt động vì một số lý do - và lý do có lẽ cũng khá đơn giản.

Theo mặc định, hệ thống annotation wiring sẽ bị **tắt**. Để làm cho nó hoạt động, bạn sẽ phải kích hoạt nó trong các tập tin cấu hình của bạn.

### Câu hỏi 3: Sự khác biệt giữa mối quan tâm ‘concern’ và ‘cross-cutting’ là gì?

Có rất nhiều so sánh trong các câu hỏi phỏng vấn Spring, đặc biệt là đối với các nhà phát triển nâng cao. ‘**Concern**’ được sử dụng khi muốn triển khai một chức năng cụ thể, duy nhất vào dự án (ứng dụng) của bạn. ‘**Cross-cutting**’ concerns là các chức năng sẽ được áp dụng thông qua toàn bộ dự án - tất cả các mô-đun trong ứng dụng sẽ bị tác động.

### Câu hỏi 4: ‘autowriting’ là gì?

‘**Autowriting**‘ cho phép nhà phát triển tự động inject bean vào ứng dụng của mình mà không cần can thiệp thủ công.

### Câu hỏi 5:‘weaving’ được sử dụng để làm gì?

‘**Weaving**’ được sử dụng để tạo ra các các đối tượng đã được xử lý phụ (Advised object). Điều này được thực hiện bằng cách liên kết các khía cạnh nhất định với các đối tượng cụ thể khác trong ứng dụng.

### Câu hỏi 6: Làm thế nào để quản lý các giao dịch trong Spring?

Spring cho phép **hai** loại quản lý giao dịch: **declarative** (khai báo) và **programmatic**(lập trình).

Quản lý giao dịch **declarative** (khai báo) là khi tách mã code khỏi các giao dịch. Một trong những câu hỏi phỏng vấn Java Spring trước đây của chúng tôi đã nói về XML. Điều này rất quan trọng vì nhiều nhà phát triển chuyển sang cấu hình XML để quản lý các giao dịch của họ theo cách đó.

Quản lý giao dịch theo **programmatic**(lập trình) liên quan đến mã code doanh nghiệp trong quản lý giao dịch. Loại này là sự lựa chọn ít phổ biến hơn trong việc quản lý các giao dịch. Có một vài lý do nhưng lý do chính phải làm việc với mã code. Nhiều lập trình viên không muốn làm với mã code để không phá vỡ một cái gì đó trong quy trình.

### Câu hỏi 7: ‘target object’ là gì?

Khi một đối tượng trở thành trung tâm của sự chú ý cho các khía cạnh, nó sẽ tự động trở thành một **target object**. Một số người cũng thích coi nó như một “*advised object*’.

### Câu hỏi 8: @Controll làm gì?

**@Controll** đặt tên một lớp Spring được chỉ định cụ thể như lớp trình điều khiển.

### Câu hỏi 9: DAO là gì?

Trong các câu hỏi phỏng vấn Spring, **DAO** viết tắt là *Data Access Object*. Công cụ này cho phép các nhà phát triển tiếp cận và làm việc dễ dàng hơn với các công cụ truy cập dữ liệu, đặc biệt là các công cụ có nguồn gốc Java.

### Câu hỏi 10: @Autowired làm gì?

Lệnh **@Autowired** cho phép bạn linh hoạt quyết định nơi muốn wire đối tượng của mình. Lệnh này làm cho quá trình dễ dàng và mượt mà hơn, cho phép độ chính xác cao hơn.

### Câu hỏi 11: @RequestMapping làm gì?

Lệnh này được sử dụng bất cứ khi nào bạn muốn sắp xếp  một phương thức HTTP cụ thể đến một lớp cụ thể. Bạn có thể sử dụng lệnh này trong cả hai cấp độ lớp và phương thức.

### Câu 12: MVC Interceptor là gì?

Trong Spring MVC, **interceptor** có thể được sử dụng để xử lý yêu cầu của khách hàng trước, trong và thậm chí sau khi thao tác. Đây là một công cụ tuyệt vời để tránh mọi sự lặp lại không mong muốn của mã code.

### Câu hỏi 13: introduction’ được dùng là gì?

Lệnh **‘introduction’** cho phép bạn thêm các thuộc tính mới vào một lớp đã được gửi lại trong Spring.

### Câu hỏi 14: Điều gì xảy ra khi sử dụng @Autowire cùng với @Qualifier?

Kiểu kết hợp này được sử dụng khi có nhiều loại bean đơn khác nhau trong ứng dụng. Sự kết hợp này phân biệt giữa từng bean.

### Câu hỏi 15: Cấu hình Annotation-based container là gì?

Nó là một cấu hình container thay thế cho XML đã đề cập trước đó. Kiểu cấu hình này chủ yếu tập trung vào *siêu dữ liệu mã byte*.