



Aula 2: Spring Framework

Abstrai a complexidade de configuração com o Spring Boot

Objetivos

1. Por que Spring?
 2. Beans
 3. Inversão de Controle (IoC)
 4. Injeção de Dependência (DI)
-



Aula 2 | Etapa 1:

Por que Spring?

Abstraia a complexidade de
configuração com o Spring Boot

Plataforma Spring

O que é o *Spring*?

Spring nada mais é que uma plataforma com diversos recursos para construção de aplicativos *Java*, facilitando assim o desenvolvimento em *Java EE* com módulos que facilitam a construção de *softwares* reduzindo o tempo de desenvolvimento.



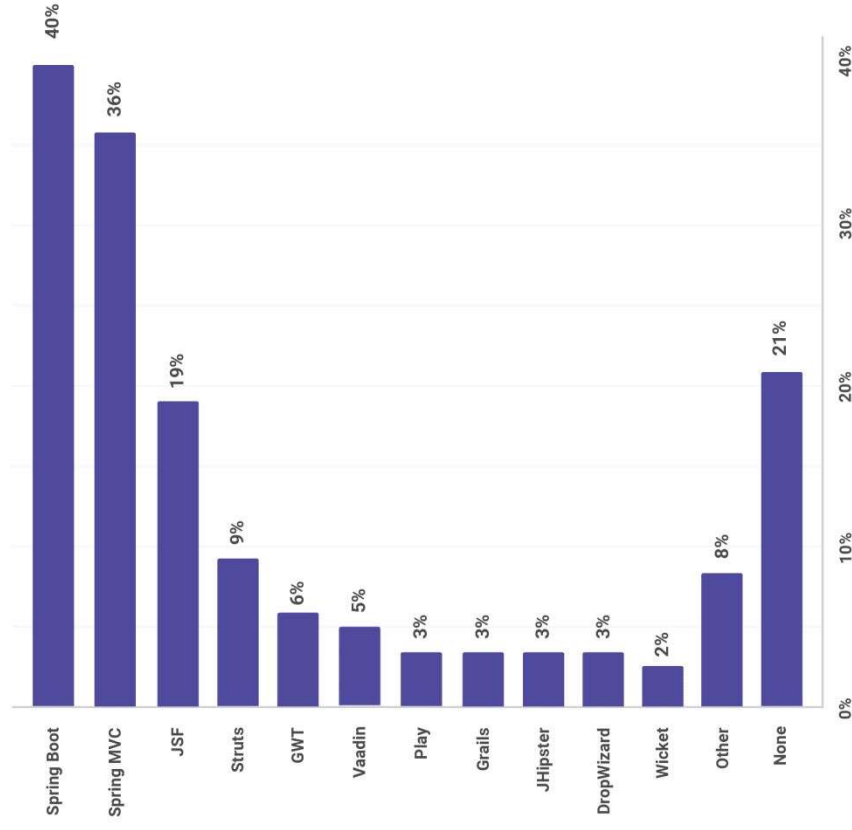
Spring Framework





DIGITAL
INNOVATION
ONE

Dados Relevantes



Fonte: Blog Snyk - JVM Ecosystem report 2018

Para saber mais

<https://spring.io/why-spring>

<https://snyk.io/blog/ivm-ecosystem-report-2018-platform-application/>



Aula 2 | Etapa 2:

Beans

Abstraia a complexidade de
configuração com o Spring Boot

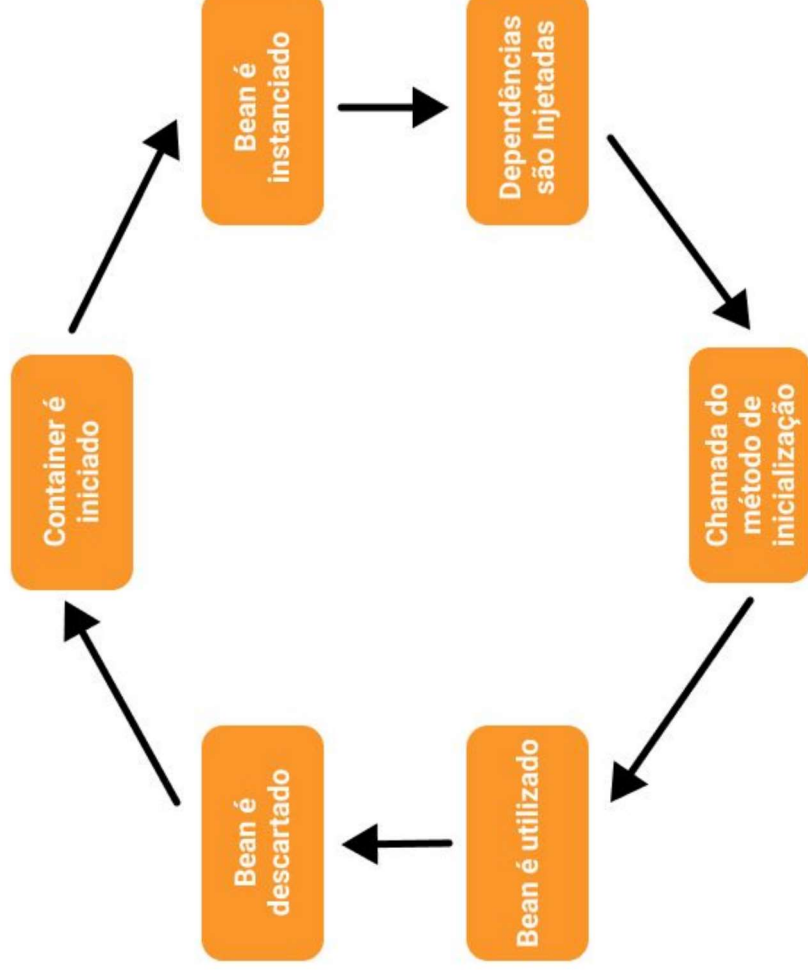
Beans

Um bean se trata de um objeto que é instanciado, montado e gerenciado por um container do Spring através de Inversão de Controle (IoC) e Injeção de Dependências (DI).





Ciclo de vida de um Bean





Configurando Beans

É possível configurar um Bean de duas formas por arquivos XML ou através de anotações.

Em XML seria preciso definir a tag `<bean>` dentro de uma tag principal `<beans>` passando o path da classe assim o Spring saberá quais classes gerenciar a criação de instâncias e a injeção de dependências.



DIGITAL
INNOVATION
ONE

Configurando Beans

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.springframework.org/schema/beans  
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  
  
  <bean id="..." class="..." />  
  
</beans>
```

BeanFactory ou ApplicationContext?

- BeanFactory fornece um mecanismo de configuração avançada capaz de gerenciar objetos de qualquer natureza.
- ApplicationContext se baseia na BeanFactory (é uma subinterface) e adiciona outras funcionalidades, como integração mais fácil com os recursos AOP do Spring, manipulação de recursos de mensagem (para uso na internacionalização), propagação de eventos e contextos específicos da camada de aplicativo, como o WebApplicationContext para uso em aplicativos da web.

Aula 2 | Etapa 3:

Inversão de Controle (IoC)

Abstraia a complexidade de
configuração com o Spring Boot



Inversão de Controle

Inversão de Controle (ou IoC - Inversion of Control) é um processo onde se inverte o fluxo de comando de um programa. É uma ideia desacoplar ou remover dependências do objeto e fornecer controle para outra camada. Este objeto delega a tarefa de construir dependências para um contêiner IoC.



Desafio

Implemente a inversão de controle em um programa simples, seguindo as orientações da aula.

<https://github.com/Re04nan/dio-experts-spring-boot-java>

Aula 2 | Etapa 4:

Injeção de Dependência (DI)

Abstraia a complexidade de
configuração com o Spring Boot



Injeção de Dependência

A Injeção de Dependência (ou DI - Dependency Injection) define quais classes serão instanciadas e onde serão injetadas quando for necessário. Existem três formas de aplicar o DI, por injeção de construtor, setter e interface. O Spring Framework aplica a IoC quando necessário também utilizando o DI.



Desafio

Implemente a injeção de dependência em um programa simples, seguindo as orientações da aula.

<https://github.com/Re04nan/dio-experts-spring-boot-java>



DIGITAL
INNOVATION
ONE

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)