

# Abstração

segunda-feira, 10 de outubro de 2022 11:39

## Conceito

O conceito de abstração consiste em esconder os detalhes de algo, no caso, os detalhes desnecessários.

No mundo real, utilizamos abstrações o tempo todo. Tudo que não sabemos como funciona por baixo dos panos pode ser considerado uma abstração.

Para exemplificar melhor, vamos tomar como exemplo a concessionária que realiza manutenções no seu carro. Você leva ele até lá com um problema e ele volta funcionando.

Em suma, pouco importa os detalhes do que aconteceu durante a manutenção do seu carro, o que importa é que ele voltou funcionando.

## Abstração na orientação a objetos

Dentro da OOP (Object Oriented Programming -- Programação Orientada a Objetos) temos diversos conceitos de abstração, como por exemplo as interfaces e classes que escondam algo.

```
public class Order
{
    ...
    public decimal Total { get; set; }
    public void CalculateTotal() {...}
    public void AddItem(Item item)
    {
        Items.Add(item);
        CalculateTotal();
    }
    ...
}
```

★ Quanto mais contato sua classe tem com o mundo externo, maiores são os impactos das mudanças quando algo for alterado nela

Neste exemplo temos uma classe Order que representa um pedido, e dentro dela temos uma propriedade chamada Total que define o valor total do pedido, e dois métodos, CalculateTotal e AddItem.

Desta forma, privamos o acesso externo utilizando o modificador de acesso private, tornando método CalculateTotal inacessível externamente.

Abstração é isso, deixar acoplado em uma entidade as operações dela, deixando externo somente o que pode fazer, não como acontece

## Abstração por interfaces

- ★ As interfaces são como contratos que definem o que as implementações devem conter e fazer.
- ★ Sempre que puder, dependa de abstrações ao invés de implementações.
  - ✍ Possibilita várias implementações de um mesmo contrato
- ★ As interfaces dizem "O que fazer" e não "Como fazer", sendo assim, podemos dizer que o "Como" são detalhes e o "O que fazer" como uma abstração.

```
public interface ICustomerRepository
{
    void Save(Customer customer);
}
```



Tomando como base a interface, tomamos ela como um contrato que diz que **Customer** pode ser salvo, mas ela não diz **como** isto deve ser feito.



```
public class CustomerRepository : ICustomerRepository
{
    public void Save(Customer customer)
    {
        _context.Customers.Add(customer);
        _context.SaveChanges();
    }
}
```

Implementação



Contrato

Mas nos testes de unidade, não podemos depender de dados externos, logo, precisamos **simular** nosso repositório.

Sendo assim, temos que gerar outra implementação baseada no contrato **ICustomerRepository** que vai "enganar" os testes.

Outra implementação: simula um save

erigir os testes.

Outra implementação: simula um save

```
public class FakeCustomerRepository : ICustomerRepository
{
    public void Save(Customer customer)
    {
    }
}
```

Contrato

Pronto, neste momento temos duas implementações (ICustomerRepository e FakeCustomerRepository) se baseado no contrato ICustomerRepository.

## Exemplo prático: Testes de Unidade

OrderHandler -> Manipula pedidos

Este objeto precisa se comunicar com a base de clientes para salvar os dados lá, porém, durante os testes, não teremos uma base para salvar.

## 2.1. Por que aprender e usar Spring?

terça-feira, 31 de janeiro de 2023

15:13

- O que é Spring?

Conjunto de projetos que resolvem vários problemas no dia a dia de desenvolvimento, pode ser referida ao ecossistema Spring

- Por que usar Spring?
  - Canivete suíço, uma tech que resolve vários problemas diferentes
  - Simplicidade
  - Modularidade -> organizado por projetos (spring security, data jpa, webflux)
  - Evolução constante
  - Open source
  - Comunidade boa

## 2.2. Conhecendo o ecossistema Spring

terça-feira, 31 de janeiro de 2023

15:19

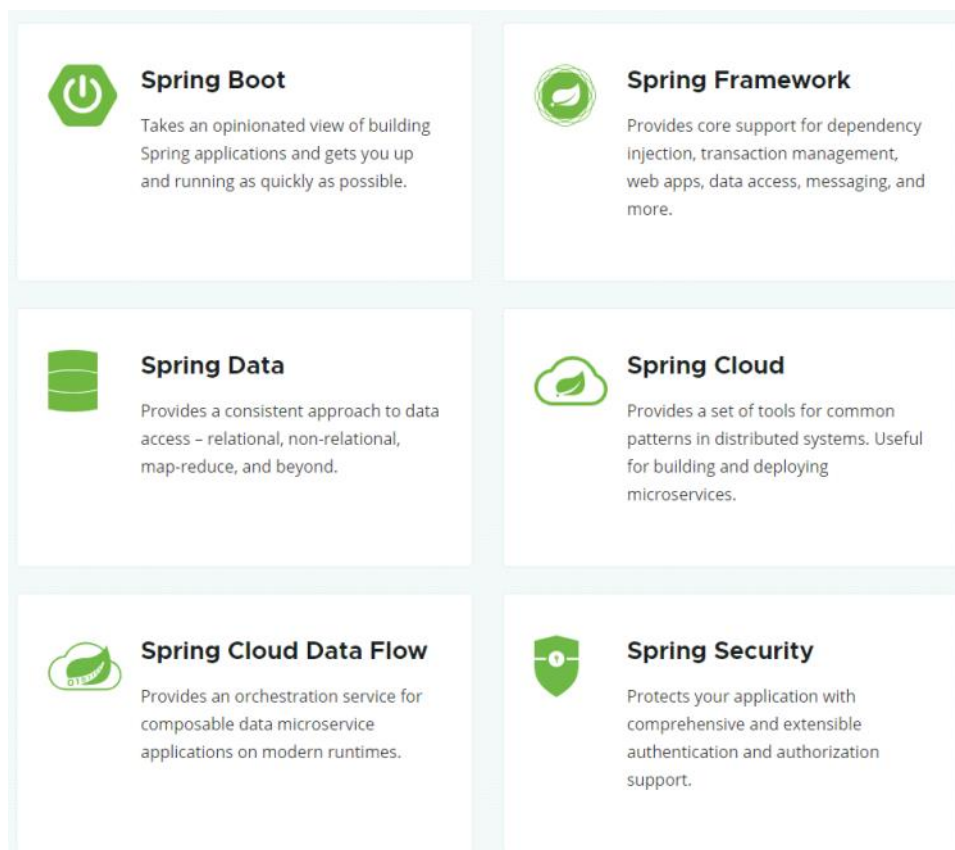
site oficial

<https://spring.io/>

Inicialmente desenvolvido para criação de aplicações web escritas em Java, e anteriormente denominado como Spring Framework, o Spring é um ecossistema de desenvolvimento para facilitar a criação de aplicações Java utilizando diversos módulos independentes.

No início, o Spring Framework possuía diversos recursos integrados em uma mesma plataforma, porém o projeto cresceu tanto que alguns de seus recursos foram copiados por outros projetos. A partir daí, surgiu a ideia de modularizar as principais funcionalidades do Spring Framework e assim facilitar a integração com outros projetos que utilizavam outros frameworks.

Com essa modularização, surgiu o projeto Spring, composto por diversos módulos, cada um com a sua especialidade. Estes módulos podem ser utilizados em conjunto com outros ou até com Frameworks que não façam parte do ecossistema Spring.



### Principais módulos do Spring

- **Spring Boot:** Módulo do Spring que facilita a configuração e publicação das aplicações desenvolvidas, visando reduzir a quantidade de configurações iniciais que geralmente são fundamentais em projetos java;

- **Spring Data:** Módulo do Spring que visa facilitar a forma de acesso aos dados por parte da aplicação. Possui suporte desde o JDBC até o JPA;
- **O Spring MVC:** Permite a criação e desenvolvimento de aplicações utilizando o padrão MVC;
- **Spring Security:** Módulo do Spring responsável por gerenciar toda a parte de autenticação e autorização de uma aplicação;

## 2.3. Spring vs Jakarta EE (Java EE)

terça-feira, 31 de janeiro de 2023

15:37

### Spring vs Jakarta EE (Java EE)

Oracle não estava dando atenção ao Java EE nos últimos anos e transferiu o código da especificação para o Eclipse Foundation, nesse processo foi rebatizado para Jakarta, pois o nome Java já estava registrada. Famoso JEE.



O Jakarta EE é uma especificação e a ideia é padronizar uma tecnologia, não é um ambiente aberto a inovações, o objetivo é padronizar o que já existe. Ex: JPA é uma especificação que está dentro do JEE, e o JPA é a especificação de algo que já existia, no caso a especificação do Hibernate (Não existia JPA na época) após isso, o Hibernate passou a implementar o produto (especificação) o JPA. Mas aí existem outras implementações do JPA, o EclipseLink

## 2.4. Conhecendo o Spring Boot

terça-feira, 31 de janeiro de 2023

15:55

Antes o Spring era de difícil configuração inicial e um simples erro de digitação poderia acarretar em uma dor de cabeça

A partir da versão 3, as configurações puderam ser feitas de forma programática. Isso evitou gerar erros pois a classe de configuração tem que ser compilada, e os erros apareciam no momento da compilação.



# Spring Boot

Com todo esse problema de preconfiguração, o Spring boot veio para simplificar, possibilitando o foco no desenvolvimento e a mínima atenção para as configurações, pois ele utiliza o conceito de Convention Over Configuration. O Spring Boot configura quase tudo abordando uma visão opinativa (convenções que grande parte do mercado utiliza), trabalhando com Maven para trabalhar com dependências. O Spring Boot simplifica a configuração no Maven com seus Starters, são dependências que agrupam outras dependências, por exemplo, o JPA precisa de várias dependências e o Starter adiciona todas as dependências e subdependências necessárias com as versões que funcionam entre si.

## Spring Boot ou Spring MVC?

Spring Boot não é framework para desenvolvimento web, ele é só uma camada de abstração até o desenvolvimento web. Spring MVC sim trabalha com web, porém, contém o Spring Boot como subprojeto.

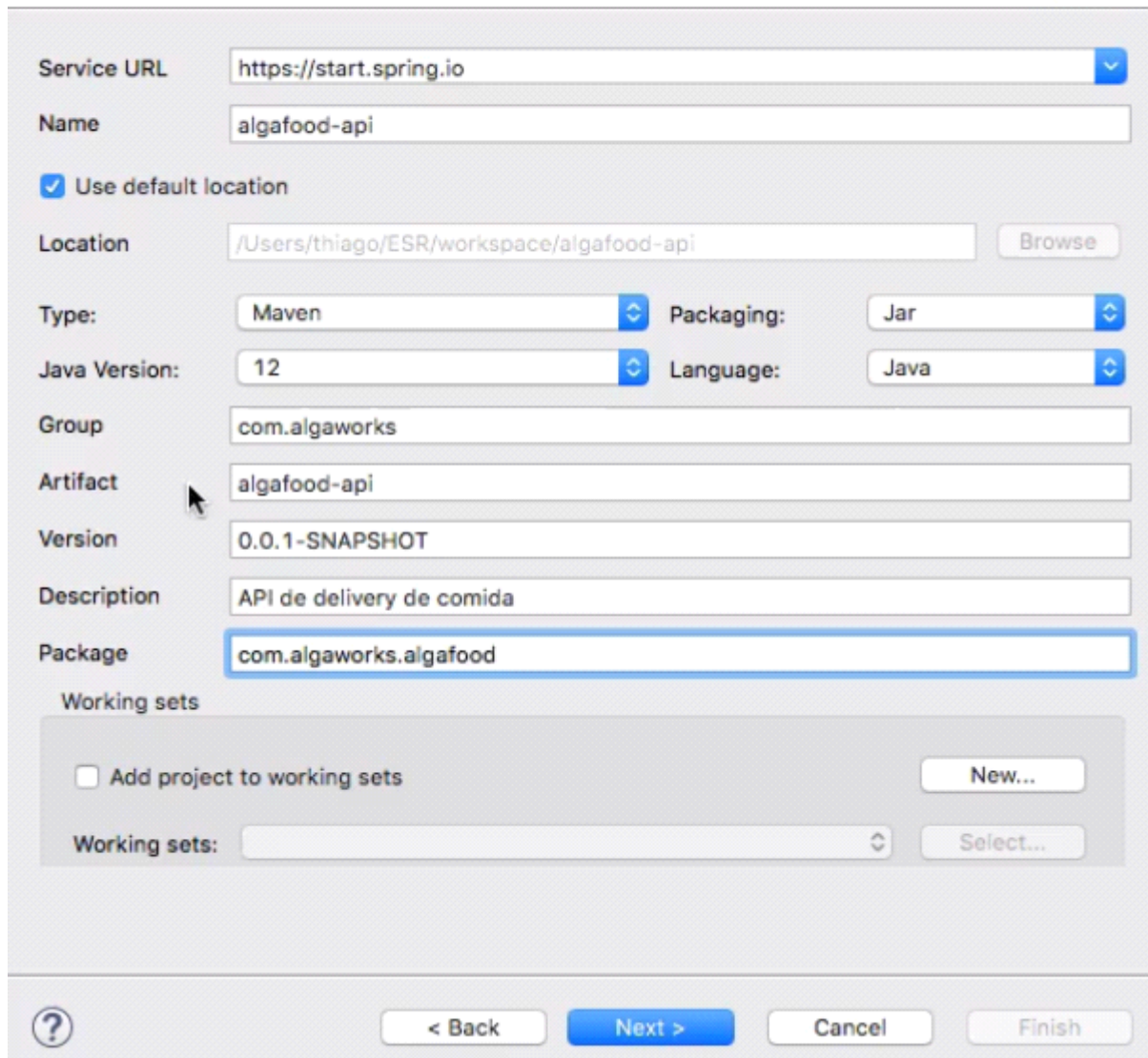
## 2.5. Criando um projeto Spring Boot com Spring Initializr

terça-feira, 31 de janeiro de 2023

19:36

Foi criado um projeto pelo Spring Tool Suite com pom.xml na versão 2.1.7.RELEASE ou utilizar o start.spring.io

### New Spring Starter Project

The image shows the 'New Spring Starter Project' dialog box in the Spring Tool Suite. It contains various fields for configuring a new project. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'algafood-api'. The 'Use default location' checkbox is checked. The 'Location' is '/Users/thiago/ESR/workspace/algafood-api'. The 'Type' is 'Maven', 'Packaging' is 'Jar', 'Java Version' is '12', and 'Language' is 'Java'. The 'Group' is 'com.algaworks', 'Artifact' is 'algafood-api', 'Version' is '0.0.1-SNAPSHOT', and 'Description' is 'API de delivery de comida'. The 'Package' is 'com.algaworks.algafood'. There is a 'Working sets' section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below that is a 'Working sets:' dropdown and a 'Select...' button. At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Finish'. A help icon (?) is on the bottom left.

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



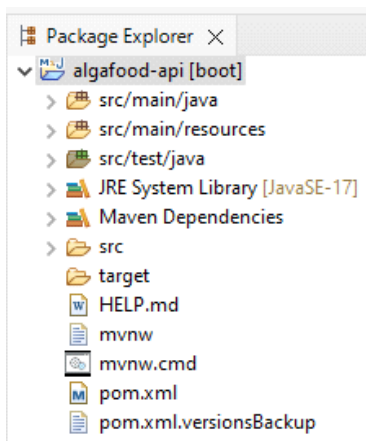
## 2.6. Conhecendo o Maven e o pom.xml de um projeto Spring Boot

quarta-feira, 1 de fevereiro de 2023 09:10

O que é Maven?

Gerenciamento de dependência e automação de build de projetos Java. Não precisa ser necessariamente com Spring

Arquivos do projeto no STS



O projeto Maven começa com uma convenção de pastas

POM.XML

Project Object Model -> arquivo de configuração

Maven Wrappers

fazer build do projeto/executa um projeto sem ter maven instalado no computador

mvnw

mvnw.cmd

O Eclipse/STS já tem o Maven embutido dentro

Como gerar um build do projeto usando Maven:

Botão direito no projeto -> Run As -> Maven Build -> em Goals digitar -> "package" para empacotar o projeto.

### Edit configuration and launch.

Name:

☒ Main ☐ JRE ☐ Refresh ☐ Source ☐ Environment ☐ Common

Base directory:

Goals:

Profiles:

User settings:

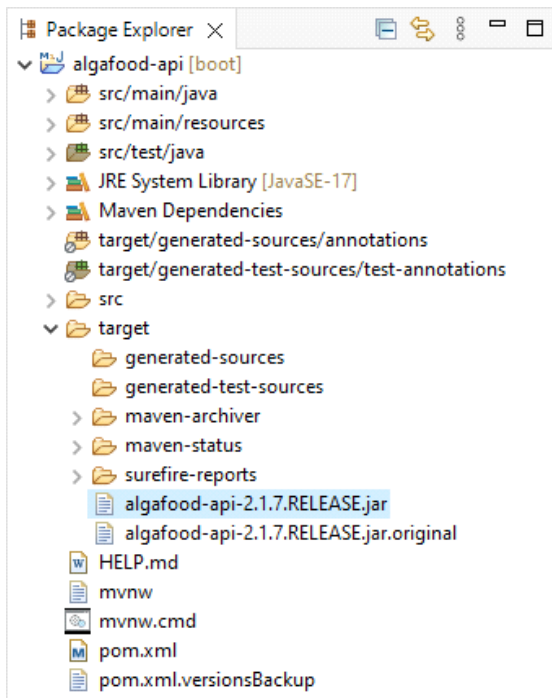
☐ Offline ☐ Update Snapshots  
☐ Debug Output ☐ Skip Tests ☐ Non-recursive  
☐ Resolve Workspace artifacts

Threads:  Color Output:

Parameter Name	Value

Arquivo gravado em:

[illegible]



Para fazer o build do projeto por linha de comando:


```
C:\Users\Guto1\stsworkspace\AlgaWorks\algafood-api>mvnw package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.algaworks:algafood-api >-----
[INFO] Building algafood-api 2.1.7.RELEASE
[INFO] -----[ jar ]-----
[INFO]
```

nvnw package

Para rodar o arquivo em qualquer computador com java:

```
java -jar algafood-api-2.1.7.RELEASE.jar
```

```
C:\Users\Guto1\stsworkspace\AlgaWorks\algafood-api\target>java -jar algafood-api-2.1.7.RELEASE.jar
```



```
:: Spring Boot :: (v2.7.8)
```

## Comando para limpar as builds

mvnw clean

## Conhecendo o POM.XML



```
algafood-api/pom.xml X
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.7.8</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.algaworks</groupId>
12  <artifactId>algafood-api</artifactId>
13  <version>2.1.7.RELEASE</version>
14  <name>algafood-api</name>
15  <description>API de delivery de comida</description>
16  <properties>
17    <java.version>17</java.version>
18  </properties>
19  <dependencies>
20    <dependency>
21      <groupId>org.springframework.boot</groupId>
22      <artifactId>spring-boot-starter-web</artifactId>
23    </dependency>
24
25    <dependency>
26      <groupId>org.springframework.boot</groupId>
27      <artifactId>spring-boot-starter-test</artifactId>
28      <scope>test</scope>
29    </dependency>
30  </dependencies>
31
32  <build>
33    <plugins>
34      <plugin>
35        <groupId>org.springframework.boot</groupId>
```

a tag parent

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.8</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
▼ ▲ parent
  ▲ groupId
  ▲ artifactId
  ▲ version
  ▲ relativePath
```

significa que estamos herdando configurações de um outro local, do groupId, do ArtifactId, da versão.

ctrl + seta do mouse para abrir o pom.xml do groupId

```
algafood-api/pom.xml  C:\Users\Guto1\m2\repository\org\springframework\boot\spring-boot-starter-parent\2.7.8\spring-boot-starter-parent-2.7.8.pom
20 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsd/maven-4.0.0.xsd" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>org.springframework.boot</groupId>
6     <artifactId>spring-boot-dependencies</artifactId>
7     <version>2.7.8</version>
8   </parent>
9   <artifactId>spring-boot-starter-parent</artifactId>
10  <packaging>pom</packaging>
11  <name>spring-boot-starter-parent</name>
12  <description>Parent pom providing dependency and plugin management for applications built with Maven</description>
13  <properties>
14    <java.version>1.8</java.version>
15    <resource.delimiter>@</resource.delimiter>
16    <maven.compiler.source>${java.version}</maven.compiler.source>
17    <maven.compiler.target>${java.version}</maven.compiler.target>
18    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
20  </properties>
21  <url>https://spring.io/projects/spring-boot</url>
22  <licenses>
23    <license>
24      <name>Apache License, Version 2.0</name>
25      <url>https://www.apache.org/licenses/LICENSE-2.0</url>
26    </license>
27  </licenses>
28  <developers>
29    <developer>
30      <name>Spring</name>
31      <email>ask@spring.io</email>
32      <organization>VMware, Inc.</organization>
33      <organizationUrl>https://www.spring.io</organizationUrl>
34    </developer>
35  </developers>
36  <scm>
37    <url>https://github.com/spring-projects/spring-boot</url>
```

a tag parent carrega todos os arquivos desse pom.xml através do conjunto de especificações.

```
algafood-api/pom.xml  C:\Users\Guto1\m2\repository\org\springframework\boot\spring-boot-starter-parent\2.7.8\spring-boot-starter-parent-2.7.8.pom
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLoc
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsd/maven-4.0.0.xsd" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3    <modelVersion>4.0.0</modelVersion>
4    <parent>
5      <groupId>org.springframework.boot</groupId>
6      <artifactId>spring-boot-starter-parent</artifactId>
7      <version>2.7.8</version>
8      <relativePath/> <!-- lookup parent from repository -->
9    </parent>
10    <groupId>com.algaworks</groupId>
11    <artifactId>algafood-api</artifactId>
12    <version>2.1.7.RELEASE</version>
13    <name>algafood-api</name>
14    <description>API de delivery de comida</description>
15  <properties>
16    <java.version>17</java.version>
17  </properties>
```

A propriedade da versão do java está sendo sobrescrita do pom.xml do groupId org.spring... pois a versão padrão do Spring 2.7.8 é a versão 1.8 do Java

algafood-api/pom.xml C:\Users\Guto1\m2\repository\org\springframework\boot\spring-boot-starter

```

https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xsi:schemaLocation="http://m
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>org.springframework.boot</groupId>
6     <artifactId>spring-boot-dependencies</artifactId>
7     <version>2.7.8</version>
8   </parent>
9   <artifactId>spring-boot-starter-parent</artifactId>
10  <packaging>pom</packaging>
11  <name>spring-boot-starter-parent</name>
12  <description>Parent pom providing dependency and plugin management for applic
13  <properties>
14    <java.version>1.8</java.version>
15    <resource.delimiter>@</resource.delimiter>
16    <maven.compiler.source>${java.version}</maven.compiler.source>
17    <maven.compiler.target>${java.version}</maven.compiler.target>
18    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
20  </properties>

```

caso a propriedade não esteja especificada no pom.xml do projeto, o maven assume a configuração do Spring

```

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

```





dependências do nosso projeto



algafood-api/pom.xml X

## Dependencies

Filter:

**Dependencies**

 spring-boot-starter-web (managed:2.7.8)  
 spring-boot-starter-test [test] (managed:2.7.8)




Add...

Remove

Properties...

Manage...

**Dependency Management**

Add...

Remove

Properties...

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.





Overview **Dependencies** Dependency Hierarchy Effective POM pom.xml

## HIERARQUIA DE DEPENDÊNCIA

## Dependency Hierarchy [test]




Filter:

**Dependency Hierarchy**

- spring-boot-starter-web : 2.7.8 [compile]
  - spring-boot-starter : 2.7.8 [compile]
    - spring-boot : 2.7.8 [compile]
      - spring-core : 5.3.25 [compile]
      - spring-context : 5.3.25 [compile]
    - spring-boot-autoconfigure : 2.7.8 [compile]
    - spring-boot-starter-logging : 2.7.8 [compile]
      - jakarta.annotation-api : 1.3.5 [compile]
      - spring-core : 5.3.25 [compile]
      - snakeyaml : 1.30 [compile]
    - spring-boot-starter-json : 2.7.8 [compile]
    - spring-boot-starter-tomcat : 2.7.8 [compile]
    - spring-web : 5.3.25 [compile]
    - spring-webmvc : 5.3.25 [compile]
  - spring-boot-starter-test : 2.7.8 [test]

**Resolved Dependencies**

- accessors-smart : 2.4.8 [test]
- android-json : 0.0.20131108.vaadin1 [test]
- apiguardian-api : 1.1.2 [test]
- asm : 9.1 [test]
- assertj-core : 3.22.0 [test]
- byte-buddy : 1.12.22 [test]
- byte-buddy-agent : 1.12.22 [test]
- hamcrest : 2.2 [test]
- jackson-annotations : 2.13.4 [compile]
- jackson-core : 2.13.4 [compile]
- jackson-databind : 2.13.4.2 [compile]
- jackson-datatype-jdk8 : 2.13.4 [compile]
- jackson-datatype-jsr310 : 2.13.4 [compile]
- jackson-module-parameter-names : 2.13.4 [compile]
- jakarta.activation-api : 1.2.2 [test]
- jakarta.annotation-api : 1.3.5 [compile]
- jakarta.xml.bind-api : 2.3.3 [test]
- jsonassert : 1.5.1 [test]
- json-path : 2.7.0 [test]
- json-smart : 2.4.8 [test]
- jul-to-slf4j : 1.7.36 [compile]
- junit-jupiter : 5.8.2 [test]
- junit-jupiter-api : 5.8.2 [test]
- junit-jupiter-engine : 5.8.2 [test]
- junit-jupiter-params : 5.8.2 [test]

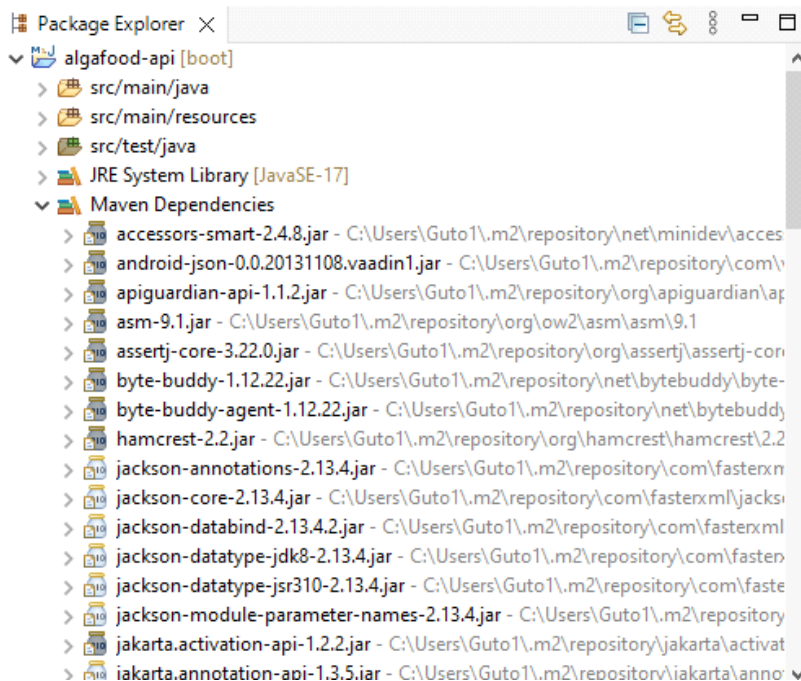
Overview **Dependencies** **Dependency Hierarchy** Effective POM pom.xml



As duas dependências adicionadas no momento da configuração do projeto adicionaram mais dependências necessárias para o funcionamento das mesmas, estas dependências adicionadas são chamadas de **dependências transitivas**.

Em **Dependências Resolvidas** mostra as dependências compiladas individualmente.

Outra forma de visualizar todas as dependências é pelo explorador de projetos



para ver as dependências por linha de comando

hierarquia de dependência:

mvnw dependency:tree

mvnw dependency:resolve

Essas dependências ficam adicionadas localmente na pasta .m2 dentro da pasta do usuário no windows



## 2.7. Criando um controller com Spring MVC

quarta-feira, 1 de fevereiro de 2023 10:02

Spring MVC é um projeto Spring para trabalhar com web, controlando requisições web e o que for necessário para o desenvolvimento web.

## 2.8. Restart mais rápido da aplicação com DevTools

quarta-feira, 1 de fevereiro de 2023 10:05

Dependência DevTools adicionada

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-devtools</artifactId>  
</dependency>
```

Os aplicativos que usam spring-boot-devtools serão reiniciados automaticamente sempre que os arquivos no caminho de classe forem alterados. Isso pode ser um recurso útil ao trabalhar em um IDE, pois fornece um loop de feedback muito rápido para alterações de código. Por padrão, qualquer entrada no caminho de classe que aponte para uma pasta será monitorada quanto a alterações.

A tecnologia de reinicialização fornecida pelo Spring Boot funciona usando dois classloaders. As classes que não mudam (por exemplo, aquelas de jars de terceiros) são carregadas em um classloader base. As classes que você está desenvolvendo ativamente são carregadas em um classloader de reinicialização. Quando o aplicativo é reiniciado, o carregador de classe de reinicialização é descartado e um novo é criado. Essa abordagem significa que as reinicializações do aplicativo geralmente são muito mais rápidas do que as "inicializações a frio", pois o carregador de classe base já está disponível e preenchido.

## 2.9. O que é injeção de dependências

segunda-feira, 10 de outubro de 2022 10:06

Injeção de Dependência é um tipo de Inversão de Controle (IoC)
É uma técnica onde um objeto fornece as dependências de outro objeto. Uma dependência é um objeto que pode ser usado (um serviço).
É uma técnica que implementa o padrão IoC

### Compreendendo o Conceito

- ★ Quando a classe A usa alguma funcionalidade da classe B, diz-se que a classe A tem uma dependência da classe B.
- ★ Em Java, antes de poder usar métodos de outras classes, primeiro precisamos criar o objeto daquela classe (ou seja, a classe A precisa criar uma instância da classe B).
- ★ Desse modo, transferir a tarefa de criação do objeto a outra entidade e usar diretamente a dependência é chamado de injeção de dependência.

### Basicamente, existem três tipos de injeção de dependências:

Injeção do construtor	As dependências são fornecidas por meio de um construtor da classe. Obrigando a passar uma instância/implementação da abstração
Injeção pelo Setter	O cliente expõe o método <i>setter</i> que o injetor usa para injetar a dependência.
Injeção de interface	a dependência fornece um método injetor, que injetará uma dependência em qualquer cliente que for passado a ele. Os clientes devem implementar uma interface que expõe um método <i>setter</i> que aceita a dependência.

```
public class Teste {
    public static void main(String[] args) {
        Pessoa pessoa = new Pessoa("Gustavo", "gustavo@gmail.com", "123456789");
        Pessoa pessoa2 = new Pessoa("Maria", "maria@gmail.com", "987654321");

        AtivacaoCliente ativacaoClienteEmail = new AtivacaoCliente(new NotificadorEMAIL());
        AtivacaoCliente ativacaoClienteSMS = new AtivacaoCliente(new NotificadorSMS());

        ativacaoClienteEmail.ativar(pessoa);
        ativacaoClienteSMS.ativar(pessoa2);
    }
}

public class AtivacaoCliente {
    private Notificador notificador;

    public AtivacaoCliente(Notificador notificador) {
        this.notificador = notificador;
    }

    public void ativar(Pessoa pessoa) {
        notificador.notificar(pessoa, "Usuário cadastrado no sistema");
    }
}
```

Tempos um serviço de notificação de clientes novos no sistema, para cada tipo de notificação, temos uma dependência necessária no construtor da Classe de AtivacaoCliente, para notificarmos por SMS é necessário uma dependência da implementação de um Notificador (geralmente uma interface) que utiliza configurações de envio por SMS

De modo geral, é responsabilidade da injeção de dependência:

- Criar objetos
- Saber quais classes necessitam desses objetos
- Fornecer todos esses objetos

Se há alguma mudança nos objetos, a DI investiga isso e isso não deve preocupar a classe que usa esses objetos. Dessa forma, se os objetos mudarem no futuro, é a responsabilidade da DI fornecer os objetos apropriados à classe.

### Inversão de dependência

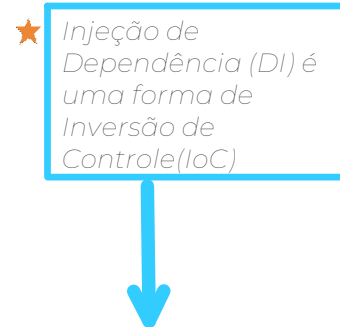
De <<https://learn.microsoft.com/pt-br/dotnet/architecture/modern-web-apps-azure/architectural-principles>>  
<https://campuscode.com.br/conteudos/s-o-l-i-d-principio-de-inversao-de-dependencia>

Fontes:  
<https://balta.io/artigos/dependency-injection>

## 2.10. Conhecendo o IoC Container do Spring

segunda-feira, 10 de outubro de 2022

13:51



★ Uma das funcionalidades mais importantes do ecossistema Spring

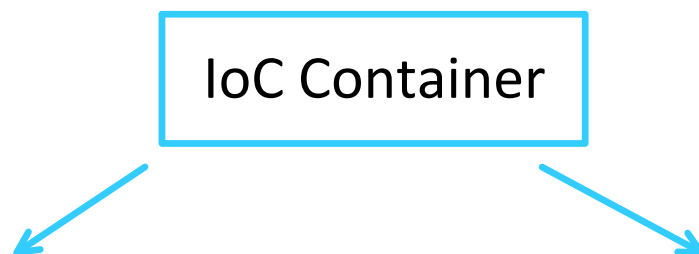
O Spring implementa o princípio de Inversão de Controle (IoC) que, abraça a Injeção de Dependência. É um processo pelo qual os objetos definem suas dependências, ou seja, outros objetos com os quais trabalham, apenas por meio de argumentos do construtor, argumentos para um método ou propriedades são definidas no momento da instanciação.

É responsável por instanciar, configurar e montar os objetos. O contêiner IoC obtém informações do arquivo XML e funciona de acordo. As principais tarefas executadas pelo container IoC são:

Instanciar a classe de aplicação

Configurar o objeto

Montar as dependências entre os objetos



### Bean Factory Container

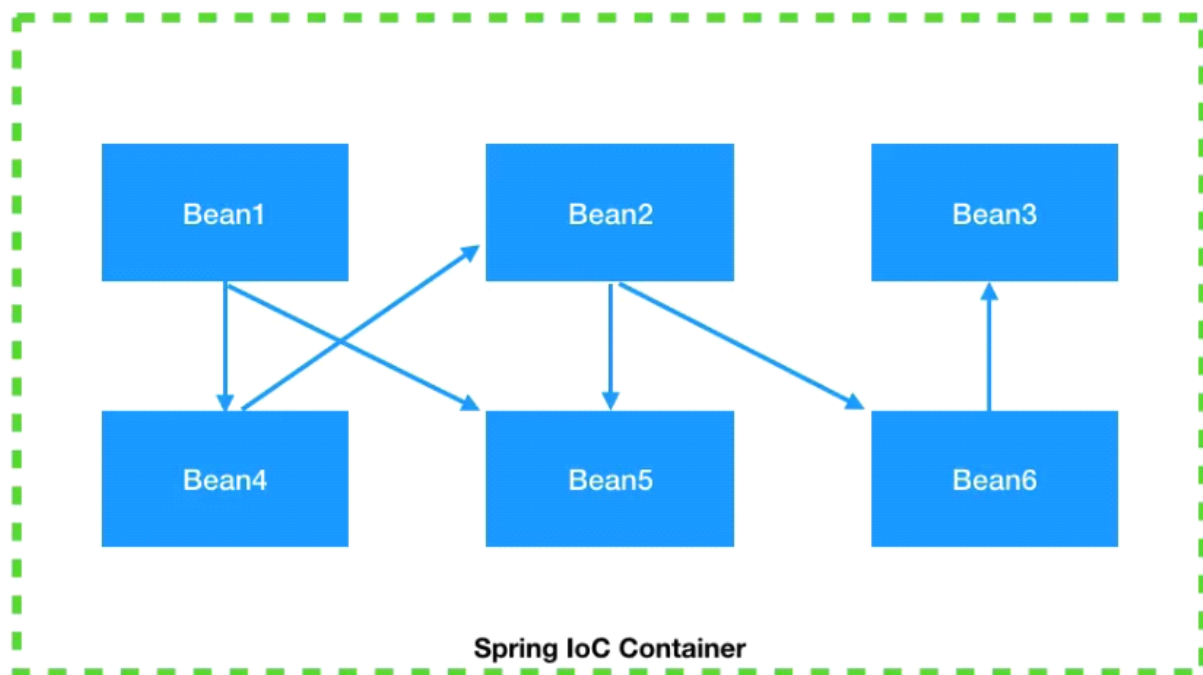
- Este é o container mais simples que fornece o suporte básico p/ ID e definido pelo `org.springframework.beans.factory.BeanFactory`.
- Várias implementações da Interface `BeanFactory`. `XmlBeanFactory` é a implementação mais usada. Esse container lê metadados de config de arquivo xml para criar sistemas configurados.
- `BeanFactory` geralmente é preferido onde os recursos são limitados, como dispositivos móveis ou aplicativos baseados em applet. Usar sempre `ApplicationContext`, a menos que não

### ApplicationContext

- Container avançado do Spring. Carrega definições de bean, conecta beans e dispensa beans mediante solicitação.
- Este contêiner é definido pela interface `org.springframework.context.ApplicationContext`.
- Implementações mais usadas:
  - `FileSystemXmlApplicationContext`
  - `ClassPathXmlApplicationContext`
  - `WebXmlApplicationContext`

## BeanFactory vs ApplicationContext

As interfaces `org.springframework.beans.factory.BeanFactory` e `org.springframework.context.ApplicationContext` atuam como o contêiner IoC.



No Spring, os objetos que formam a espinha dorsal da sua aplicação e que sejam gerenciados pelo Spring são chamados de beans. Um bean é um objeto que é instanciado, montado e gerenciado pelo Spring IoC container. O relacionamento entre esses beans é chamado de **Injeção de Dependência**.

- Bean1 é injetado em Bean4 e em Bean5
- Bean4 é injetado em Bean2

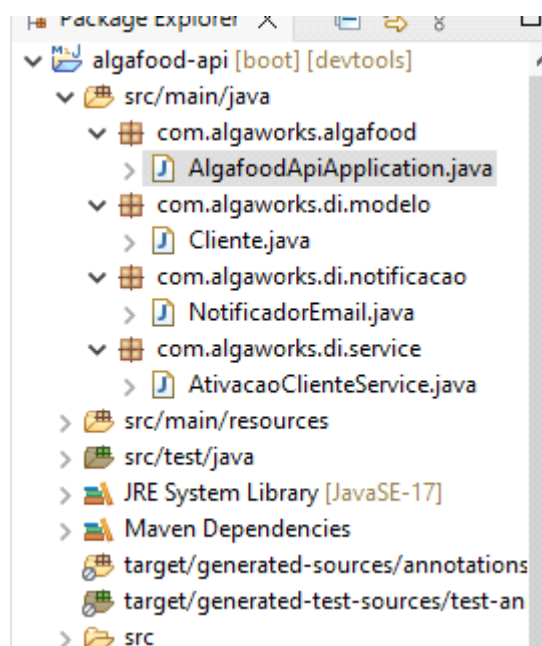
## 2.11. Definindo beans com @Component

quarta-feira, 1 de fevereiro de 2023 15:46

Ao trabalhar com Spring, podemos anotar nossas classes para transformá-las em Beans do Spring. Além disso, podemos dizer ao Spring onde procurar essas classes anotadas, pois nem todas elas devem se tornar beans nessa execução específica.

O Spring verifica classes anotadas a nível de pacote e subpacote onde a anotação **@SpringBootApplication** e o método **SpringApplication.run(AlgafoodApiApplication.class, args);** está definido

```
1 package com.algaworks.algafood;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @SpringBootApplication
8 @ComponentScan(basePackages = "com.algaworks.di")
9 public class AlgafoodApiApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(AlgafoodApiApplication.class, args);
13     }
14
15 }
```



A classe **NotificadorEmail** está anotada com **@component** e tecnicamente, seria um bean gerenciado pelo **Container do Spring IoC**, mas o pacote da classe que inicia o Spring Context está no nível de **com.algaworks.algafood** e a classe anotada com **@component** está fora do escopo do pacote em **com.algaworks.di.notificacao**. Para surtir efeito é necessário adicionar um **basePackage** para mapear uma classe bean fora do nível do pacote padrão

## 2.12. Injetando dependências (beans Spring)

quarta-feira, 1 de fevereiro de 2023

17:19

1ª opção para injeção de dependência é pelo construtor. Temos 2 classes anotadas indicando que as classes são componentes do Spring. As classes são gerenciáveis e podem ser instanciadas pelo Spring virando objetos bean.

```
@Component
public class AtivacaoClienteService {

    private NotificadorEmail notificadorEmail;

    public AtivacaoClienteService(NotificadorEmail notificadorEmail) {
        this.notificadorEmail = notificadorEmail;
        System.out.println("Bean AtivacaoClienteService gerenciado pelo Spring: " + this);
        System.out.println("Bean injetado em AtivacaoClienteService: " + notificadorEmail);
    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificadorEmail.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

```
@Component
public class NotificadorEmail {

    public NotificadorEmail() {
        System.out.println("Bean NotificadorEmail gerenciado pelo Spring: " + this);
    }

    public void notificar(Cliente c, String m) {
        System.out.printf("Notificando %s através do email %s : %s\n", c.getNome(), c.getEmail(), m);
    }
}
```

Para injetar um objeto em uma classe, as duas classes tem que ser reconhecidas como partes do Spring, e podem ser injetadas pelo construtor de uma delas, como a baixo:

```
@Component
public class AtivacaoClienteService {

    private NotificadorEmail notificadorEmail;

    public AtivacaoClienteService(NotificadorEmail notificadorEmail) {
        this.notificadorEmail = notificadorEmail;
        System.out.println("Bean AtivacaoClienteService gerenciado pelo Spring: " + this);
        System.out.println("Bean injetado em AtivacaoClienteService: " + notificadorEmail);
    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificadorEmail.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

O Spring, quando vai instanciar um AtivacaoClienteService, necessita de um objeto NotificadorEmail e ele mesmo faz isso para instanciar AtivacaoClienteService.

```
2023-02-01 19:10:41.639 INFO 7276 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload
Bean NotificadorEmail gerenciado pelo Spring: com.algaworks.di.notificacao.NotificadorEmail@4acf90a1
Bean AtivacaoClienteService gerenciado pelo Spring: com.algaworks.di.service.AtivacaoClienteService@58fc9fc8
Bean injetado em AtivacaoClienteService: com.algaworks.di.notificacao.NotificadorEmail@4acf90a1
```



um Bean de NotificadorEmail foi injetado no construtor de AtivacaoClienteService.

## 2.13. Usando @Configuration e @Bean para definir beans

quinta-feira, 2 de fevereiro de 2023 09:09

Para definir beans personalizados é necessário ter uma classe de configuração de Bean anotada por @Configuration e um método que retorna um objeto instanciado e configurado dentro de tal método, além da anotação @Bean no método.

A classe onde se encontra o objeto para injetar a dependência não precisa estar anotada com anotações de configuração e @component.

```
@RestController
@RequestMapping("/api")
public class Controller {

    private AtivacaoClienteService ativacaoClienteService;

    public Controller(AtivacaoClienteService ativacaoClienteService) {
        this.ativacaoClienteService = ativacaoClienteService;

        System.out.println("MeuPrimeiroController: " + ativacaoClienteService);
    }

    @GetMapping("/hello")
    @ResponseBody
    public String hello() {
        Cliente joao = new Cliente("João", "joao@xyz.com", "3499998888");

        ativacaoClienteService.ativar(joao);

        return "Hello!";
    }
}
```

A classe Controller precisa de uma instância de AtivacaoClienteService no construtor e o IoC Container precisa ter no seu contexto um Bean para essa dependência.

```
@Component
@Qualifier(value = "NotificadorZap")
public class AtivacaoClienteService {

    private Notificador notificadorEmail;

    public AtivacaoClienteService(Notificador notificadorEmail) {
        this.notificadorEmail = notificadorEmail;
        System.out.println("Bean AtivacaoClienteService gerenciado pelo Spring: " + this);
        System.out.println("Bean injetado em AtivacaoClienteService: " + notificadorEmail);
    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificadorEmail.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

A classe AtivacaoClienteService precisa de um Bean de Notificador e caso haver uma classe gerenciada pelo IoC que implemente a interface Notificador e anotada com @Component, o IoC vai criar e gerenciar esse Bean de tal implementação.

```

public class NotificadorEmail implements Notificador {

    private boolean caixaAlta;
    @SuppressWarnings("unused")
    private String hostServidorSmtp;

    public NotificadorEmail(String hostServidorSmtp) {
        this.hostServidorSmtp = hostServidorSmtp;
        System.out.println("Notificador email");
    }

    @Override
    public void notificar(Cliente c, String m) {
        if(this.caixaAlta)
            m = m.toUpperCase();
        System.out.printf("Notificando %s através do email %s : %s\n", c.getNome(), c.getEmail(), m);
    }

    public void setCaixaAlta(boolean caixaAlta) {
        this.caixaAlta = caixaAlta;
    }

}

```

A classe NotificadorEmail (Implementação de Notificador) anotada com @Component, o @Component sinaliza que é uma classe gerenciada pelo IoC e cria e gerência o Bean da classe quando for necessário usá-la. O Bean é registrado em BeanFactory (contêiner Spring) e podemos inicializar a aplicação Spring e solicitar o Bean. Mas o IoC não consegue instanciar pois há um parâmetro no construtor da classe implementadora que o IoC não consegue criar o objeto (pois tal tipo de objeto esperado não está registrando no BeanFactory), que não é gerenciado pelo IoC. A partir disso podemos criar e configurar nossos próprios Beans com @Bean e retornar uma instância de um objeto que é necessário para a criação de outro Bean. O Spring é inteligente o suficiente para chamar a implementação.

```

8  @Configuration
9  public class AlgaConfig {
10
11  @Bean
12  NotificadorEmail notificadorEmail() {
13      NotificadorEmail notificadorEmail = new NotificadorEmail("smtp.algamail.com.br");
14      notificadorEmail.setCaixaAlta(true);
15      return notificadorEmail;
16  }
17
18 }

```

Agora a injeção dessa dependência no construtor da classe NotificadorEmail poderá ser realizada pois estamos criando e configurando manualmente um objeto e o @Bean se encarrega de registrar na interface BeanFactory do IoC Spring. Essa estratégia também simplifica a injeção de dependência pois estamos criando manualmente.

## CRIANDO UM BEAN QUE PRECISA DE OUTRO BEAN PARA SER CONFIGURADO

```

public class AtivacaoClienteService {

    private NotificadorEmail;

    public AtivacaoClienteService(NotificadorEmail notificadorEmail) {
        this.notificadorEmail = notificadorEmail;
        System.out.println("Bean AtivacaoClienteService gerenciado pelo Spring: " + this);
        System.out.println("Bean injetado em AtivacaoClienteService: " + notificadorEmail);
    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificadorEmail.notificar(cliente, "Seu cadastro no sistema está ativo");
    }

}

```

A classe não é anotada e iremos criar o Bean manualmente

```

@Bean
NotificadorEmail notificadorEmail() {
    NotificadorEmail notificadorEmail = new NotificadorEmail("smtp.algamail.com.br");
    notificadorEmail.setCaixaAlta(true);
    return notificadorEmail;
}

@Bean
AtivacaoClienteService ativacaoClienteService() {

    return new AtivacaoClienteService(notificadorEmail());
}

```

Para criar um objeto de `AtivacaoClienteService` é necessário ter uma instância de `NotificadorEmail`, e podemos chamar o método da mesma classe que retorna um objeto de `NotificadorEmail`, mas o objeto retornado é um Bean, mesmo que estejam na mesma classe de configuração.

## MÉTODOS DE DEFINIÇÃO DE BEAN EM CLASSES DE CONFIGURAÇÃO DIFERENTES

```

9
10 @Configuration
11 public class ServiceConfig {
12
13     @Bean
14     AtivacaoClienteService ativacaoClienteService(Notificador notificador) {
15
16         return new AtivacaoClienteService(notificador);
17     }
18
19 }

```

A classe de configuração tem um método de definição de Bean que precisa de um Bean de outra classe, o Spring conhece o tipo de Bean que precisa para passar no método.

```
@Configuration
public class SmtplibConfig {

    @Bean
    NotificadorEmail notificadorEmail() {
        NotificadorEmail notificadorEmail = new NotificadorEmail("smtp.algamil.com.br");
        notificadorEmail.setCaixaAlta(true);
        return notificadorEmail;
    }
}
```

Contado que esteja anotadas.

## 2.14. Conhecendo os pontos de injeção e a anotação @Autowired

quinta-feira, 2 de fevereiro de 2023 10:34

Os pontos de injeção são pontos na classe onde acontece a injeção dos Beans (configurados manualmente ou pelo IoC) por exemplo injetando no construtor pois para o IoC container utilizar a classe, precisa de uma instância no construtor, e o IoC sabe qual é a instância.

```
@Component
public class AtivacaoClienteService {

    private Notificador notificador;

    public AtivacaoClienteService(Notificador notificador) {
        this.notificador = notificador;
    }

    public AtivacaoClienteService(String outroObj) {

    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

Na classe, existem dois construtores, e o Spring não sabe qual construtor chamar, então vai tentar instanciar pelo construtor padrão (vazio) mas não há.

```
org.springframework.beans.factory.NoSuchMethodException: No default constructor found; nested exception is java.lang.NoSuchMethodException: com.algaworks.di.service.AtivacaoClienteService()
at (SimpleInstantiationStrategy.java:83) ~[spring-beans-5.3.25.jar:5.3.25]
```

```
@Component
public class AtivacaoClienteService {

    private Notificador notificador;

    @Autowired
    public AtivacaoClienteService(Notificador notificador) {
        this.notificador = notificador;
    }

    public AtivacaoClienteService(String outroObj) {

    }

    public void ativar (Cliente cliente) {
        cliente.ativar();
        notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

Usando a anotação @Autowired, especifica qual ponto de injeção o IoC deve utilizar para injetar.

SETTER COMO PONTO DE INJEÇÃO

```

1 @Component
2 public class AtivacaoClienteService {
3
4     private Notificador notificador;
5
6     public void ativar (Cliente cliente) {
7         cliente.ativar();
8         notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
9     }
10
11     public Notificador getNotificador() {
12         return notificador;
13     }
14
15     @Autowired
16     public void setNotificador(Notificador notificador) {
17         this.notificador = notificador;
18     }
19
20 }

```

Ao adicionar a anotação @Autowired em um setter que recebe um objeto gerenciado pelo Spring, o IoC injeta o Bean necessário para a utilização da interface notificadora.

### ATRIBUTO COMO PONTO DE INJEÇÃO

```

1 @Component
2 public class AtivacaoClienteService {
3
4     @Autowired
5     private Notificador notificador;
6
7     public void ativar (Cliente cliente) {
8         cliente.ativar();
9         notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
10    }
11
12    public Notificador getNotificador() {
13        return notificador;
14    }
15 }

```

## 2.15. Dependência opcional com @Autowired

quinta-feira, 2 de fevereiro de 2023

10:57

Nós podemos sinalizar que uma dependência em um ponto de injeção pode ser opcional

```
9 @Component
10 public class AtivacaoClienteService {
11
12     @Autowired
13     private Notificador notificador;
14
15     public void ativar (Cliente cliente) {
16         cliente.ativar();
17         notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
18     }
19
20     public Notificador getNotificador() {
21         return notificador;
22     }
23
24 }
```

ou sinalizar para que o IoC ignore o ponto de injeção na hora de criar um objeto Bean.

Ex:

```
5
6 // @Component
7 public class NotificadorEmail implements Notificador {
8
9     @Override
10     public void notificar(Cliente c, String m) {
11         System.out.printf("Notificando %s através do email %s : %s\n", c.getNome(), c.getEmail(), m);
12     }
13
14 }
15 }
```

comentando @Component não há uma implementação de um Notificador que o IoC gerencie para criação de um Bean.

Ao tentar inicializar a aplicação, ocorre um erro:

Field notificador in com.algaworks.di.service.AtivacaoClienteService required a bean of type 'com.algaworks.di.notificacao.Notificador

Falha na classe pois requer um Bean do tipo Notificador. Mas podemos sinalizar que é opcional com @Autowired(required = false)

```
@Autowired(required = false)
private Notificador notificador;

public void ativar (Cliente cliente) {
    cliente.ativar();
    notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
}

public Notificador getNotificador() {
    return notificador;
}
```



## 2.16. Ambiguidade de beans e injeção de lista de beans

quinta-feira, 2 de fevereiro de 2023 12:54

Quanto conter múltiplos Beans para uma classe injetável, o Spring lança um erro de ambiguidade de Beans, e uma das soluções é criando uma lista de objetos para receber esses múltiplos beans.

```
2
3 import java.util.List;
10
11 @Component
12 public class AtivacaoClienteService {
13
14     @Autowired(required = false)
15     private List<Notificador> notificadores;
16
17     public void ativar(Cliente cliente) {
18         cliente.ativar();
19
20         for(Notificador n : notificadores) {
21             n.notificar(cliente, "Seu cadastro no sistema está ativo");
22         }
23     }
24 }
25
26
```

Problems | Javadoc | Declaration | Console | Progress | Terminal

algafood-api - AlgafoodApiApplication [Spring Boot App] C:\dev\spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86\_64.self-extra

2023-02-02 14:05:05.316 INFO 9116 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root

MeuPrimeiroController: com.algaworks.di.service.AtivacaoClienteService@4a4f3645

2023-02-02 14:05:05.823 INFO 9116 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Live

2023-02-02 14:05:05.898 INFO 9116 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomc

2023-02-02 14:05:05.920 INFO 9116 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication : Start

2023-02-02 14:05:23.217 INFO 9116 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Init

2023-02-02 14:05:23.217 INFO 9116 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init

2023-02-02 14:05:23.219 INFO 9116 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Compl

Notificando João por SMS através do telefone 3499998888 : Seu cadastro no sistema está ativo

Notificando João através do email joao@xyz.com : Seu cadastro no sistema está ativo

```
NotificadorEmail.java
1 package com.algaworks.di.notificacao;
2
3 import org.springframework.stereotype.Component;
4
5 import com.algaworks.di.modelo.Cliente;
6
7 @Component
8 public class NotificadorEmail implements Notificador {
9
10     @Override
11     public void notificar(Cliente c, String m) {
12         System.out.printf("Notificando %s por SMS através do telef
13     }
14 }
15
16
```

```
NotificadorSMS.java
1 package com.algaworks.di.notificacao;
2
3 import org.springframework.stereotype.Component;
4
5 import com.algaworks.di.modelo.Cliente;
6
7 @Component
8 public class NotificadorSMS implements Notificador {
9
10     @Override
11     public void notificar(Cliente c, String m) {
12         System.out.printf("Notificando %s através do email %s : %s
13     }
14 }
15
16
```

## 2.17. Desambiguação de beans com @Primary

quinta-feira, 2 de fevereiro de 2023 14:09

Podemos anotar a classe que servirá como Bean com @Primary para dar prioridade na injeção de dependência

## 2.18. Desambiguação de beans com @Qualifier

quinta-feira, 2 de fevereiro de 2023 14:11

Podemos usar um identificador do Bean para injetar na nossa classe gerenciada

```
@Component
public class AtivacaoClienteService {

    @Autowired
    @Qualifier("Email")
    private Notificador notificador;

    public void ativar(Cliente cliente) {
        cliente.ativar();

        notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

```
8
9 @Component
10 @Qualifier("Email")
11 public class NotificadorEmail implements Notificador {
12
13     @Override
14     public void notificar(Cliente c, String m) {
15         System.out.printf("Notificando %s por SMS através do telefone %s : %s\n", c.getNome(), c.getTelefone(), m);
16     }
17
18 }
19
```

## 2.19. Desambiguação de beans com anotação customizada

quinta-feira, 2 de fevereiro de 2023

14:16

Podemos criar uma anotação customizada para solucionar o problema de ambiguação de forma elegante usando enumerações.

```
7
8 @Retention(RetentionPolicy.RUNTIME)
9 @Qualifier
10 public @interface TipoDoNotificador {
11     NivelUrgencia value();
12 }
13
```

primeiros criamos uma anotação que sobrescreve Qualifier e que pode ser identificada por uma enumeração.

```
public enum NivelUrgencia {
    URGENTE,
    NORMAL
}
```

Depois definimos uma enumeração para identificar os tipos de Beans para o Qualifier

```
7 @Component
8 @TipoDoNotificador(NivelUrgencia.URGENTE)
9 public class NotificadorSMS implements Notificador {
10
11     @Override
12     public void notificar(Cliente c, String m) {
13         System.out.printf("Notificando %s através do email %s : %s\n", c.getNome(), c.getEmail(), m);
14     }
15
16 }
```

Podemos sinalizar cada Bean com a anotação customizada e sua enumeração para identificação

```
@Component
public class AtivacaoClienteService {

    @Autowired
    @TipoDoNotificador(NivelUrgencia.NORMAL)
    private Notificador notificador;

    public void ativar(Cliente cliente) {
        cliente.ativar();

        notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
    }
}
```

Para utilizar, do mesmo modo que definíamos a classe do Bean como Qualifier e colocamos um id e no objeto que será injetado o Bean com Qualifier também, desse mesmo modo, colocamos a anotação customizada nas duas ocasiões, na classe do

Bean e no ponto de injeção.

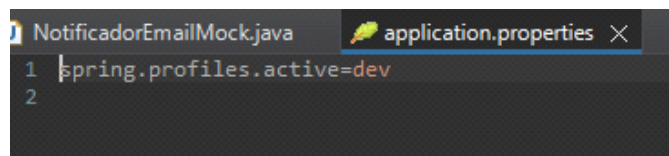
## 2.20. Mudando o comportamento da aplicação com Spring Profiles

quinta-feira, 2 de fevereiro de 2023 15:23

É uma forma de separar componentes de forma que a aplicação se comporte de maneira diferente em determinados ambientes pré-configurados.

```
8 @Component
9 @Profile(value = "dev")
10 @TipoDoNotificador(NivelUrgencia.NORMAL)
11 public class NotificadorEmailMock implements Notificador {
12
13     @Override
14     public void notificar(Cliente c, String m) {
15         System.out.printf("Notificando %s por EmailFake %s : %s\n", c.getNome(), c.getEmail(), m);
16     }
17 }
18 }
```

Quando anotamos uma classe com `@Profile` e especificamos em uma String qual o perfil está contido aquela classe, o container cria um Bean normalmente caso o mesmo perfil esteja ativo em `application.properties`, caso o perfil da classe não esteja ativo, o IoC não gerencia a classe



```
1 spring.profiles.active=dev
2
```

caso haja classes anotadas com perfis de ambiente e nenhuma configuração no `app.properties` definida, dá erro de compilação pois o container do Spring ignora todas as classes com perfis de ambiente que não esteja definida com `spring.profiles`.

```
3 import org.springframework.context.annotation.Profile;
7
8 @Component
9 @Profile(value = "dev")
10 @TipoDoNotificador(NivelUrgencia.NORMAL)
11 public class NotificadorEmailMock implements Notificador {
12
13     @Override
14     public void notificar(Cliente c, String m) {
15         System.out.printf("Notificando %s por EmailFake %s : %s\n", c.getNome(), c.getEmail(), m);
16     }
17 }
18
19
```

Problems Javadoc Declaration Console Progress Terminal

algafood-api - AlgafoodApiApplication [Spring Boot App] C:\dev\spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86\_64.self-extr

```
2023-02-02 15:40:00.675 INFO 1564 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Init
2023-02-02 15:40:00.675 INFO 1564 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root
MeuPrimeiroController: com.algaworks.di.service.AtivacaoClienteService@6648f022
2023-02-02 15:40:01.051 INFO 1564 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Live
2023-02-02 15:40:01.162 INFO 1564 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomc
2023-02-02 15:40:01.174 INFO 1564 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication : Stan
2023-02-02 15:40:04.711 INFO 1564 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Init
2023-02-02 15:40:04.711 INFO 1564 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init
2023-02-02 15:40:04.712 INFO 1564 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Comp
Notificando João por EmailFake joao@xyz.com : Seu cadastro no sistema está ativo
```

estava configurada para perfil de dev e a anotação que estava sinalizada com perfil de dev foi gerenciada pelo Spring.

Para alterar a classe gerenciada a partir de um perfil, basta apenas mudar na configuração

```
NotificadorEmailMock.java application.properties NotificadorEmail.java
1 spring.profiles.active=prod
2
```

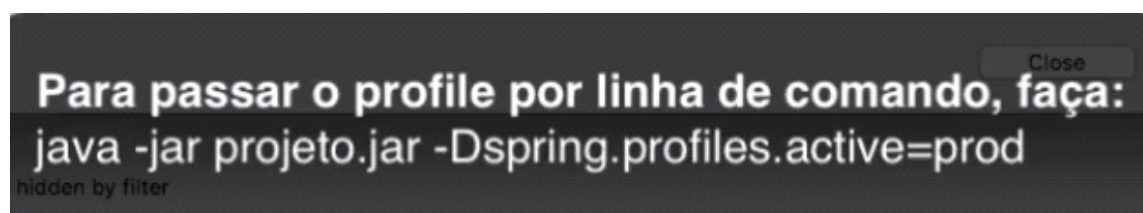
```
2
3 import org.springframework.context.annotation.Profile;
4
5
6
7
8 @Component
9 @TipoDoNotificador(NivelUrgencia.NORMAL)
10 @Profile(value = "prod")
11 public class NotificadorEmail implements Notificador {
12
13     @Override
14     public void notificar(Cliente c, String m) {
15         System.out.printf("Notificando %s por SMS através do telefone %s : %s\n",
16             c.getNome(), c.getTelefone(), m);
17     }
18 }
19
20
```

Problems Javadoc Declaration Console X Progress Terminal

algafood-api - AlgafoodApiApplication [Spring Boot App] C:\dev\spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86\_64

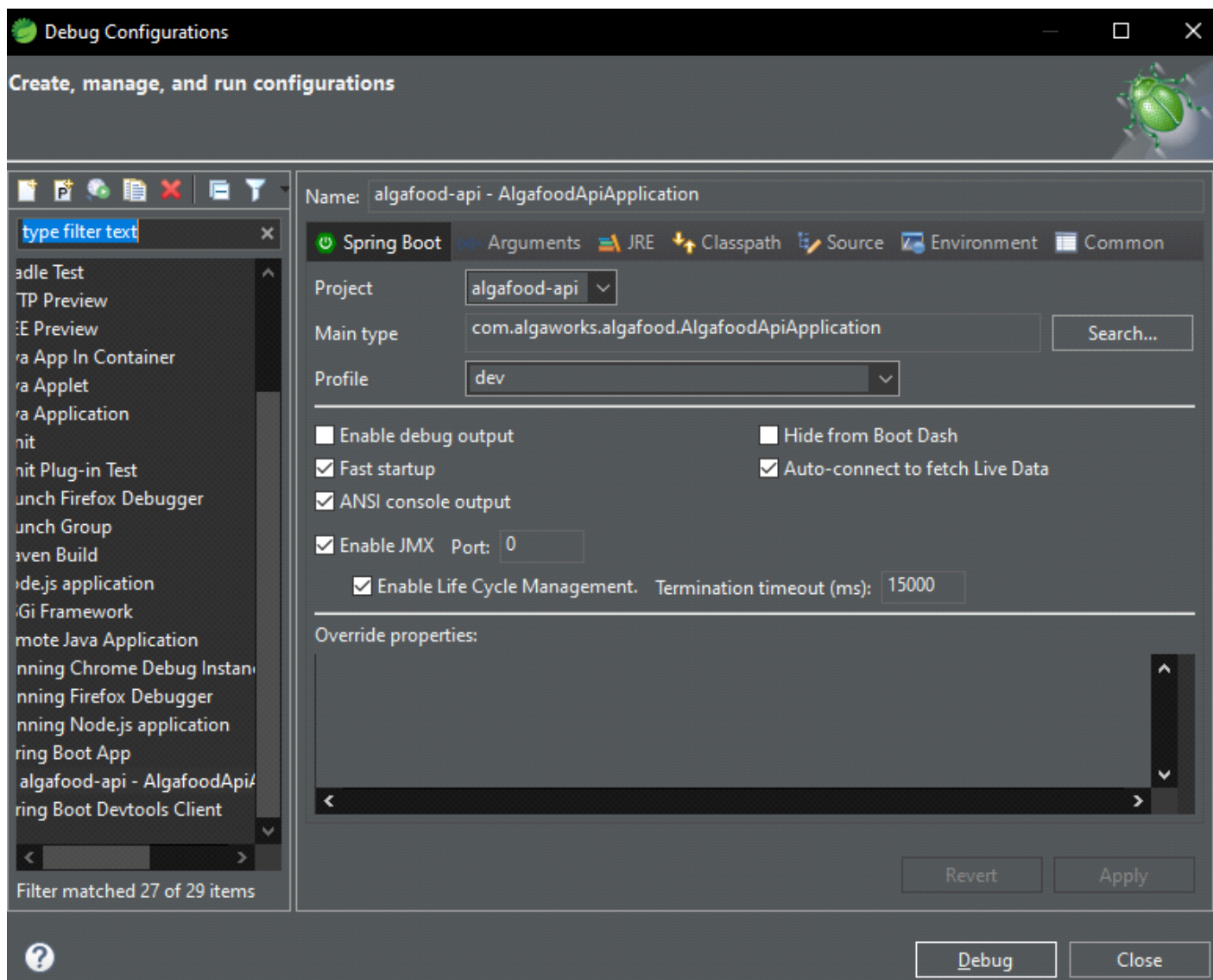
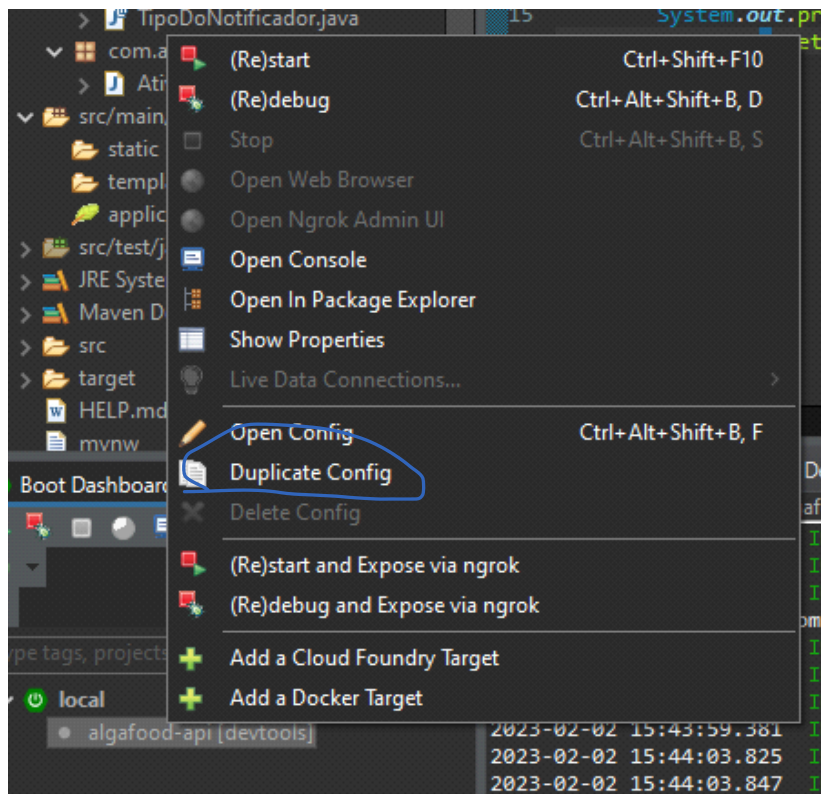
```
2023-02-02 15:41:50.795 INFO 1564 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext
MeuPrimeiroController: com.algaworks.di.service.AtivacaoClienteService@d50d75b
2023-02-02 15:41:50.948 INFO 1564 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2023-02-02 15:41:50.964 INFO 1564 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2023-02-02 15:41:50.969 INFO 1564 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication
2023-02-02 15:41:50.976 INFO 1564 --- [ restartedMain] .ConditionEvaluationDeltaLoggingList
2023-02-02 15:42:23.661 INFO 1564 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat-1].[localhost].[/]
2023-02-02 15:42:23.661 INFO 1564 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet
2023-02-02 15:42:23.662 INFO 1564 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet
Notificando João por SMS através do telefone 3499998888 : Seu cadastro no sistema está ativo
```

Para executar uma aplicação com build e com um perfil de ambiente, ou seja, aplicação já construída, podemos adicionar pelo terminal o seguinte comando



ou





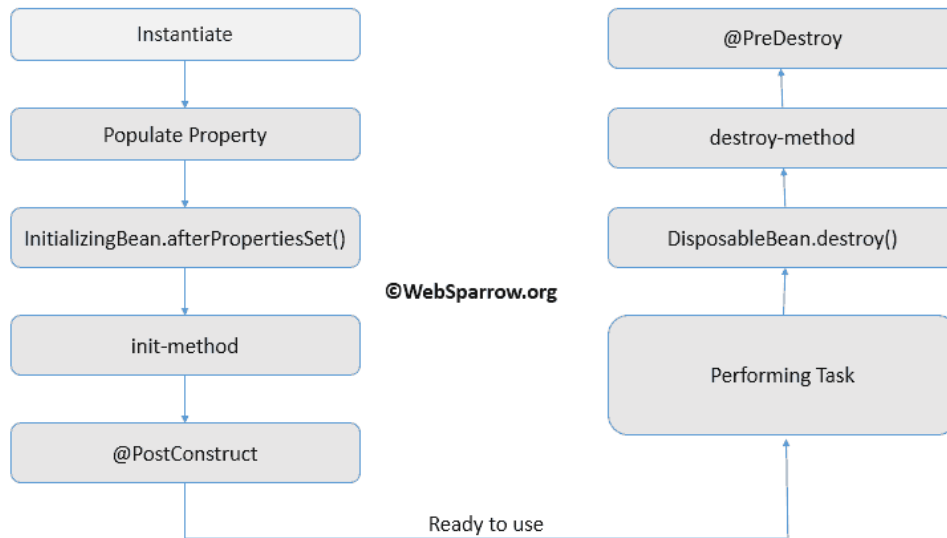
e definir o profile que irá rodar na aplicação.



## 2.21. Criando métodos de callback do ciclo de vida dos beans

quinta-feira, 2 de fevereiro de 2023 17:35

Beans tem ciclos de vida, desde quando ele é registrado no BeanFactory até quando ele deixa de existir no container IoC. Basicamente, existem 3 fases; inicialização, utilização e destruição.



Podemos criar métodos para serem chamados no momento da construção/inicialização dos beans, com a anotação @PostConstruct e @PreDestroy dentro da classe onde o Bean é injetado.

## 2.22. Publicando e consumindo eventos customizados

quinta-feira, 2 de fevereiro de 2023 17:59

Existe um padrão de projeto chamado Observer que permite as classes terem um baixo acoplamento entre elas, essa característica é uma das boas consequências e o Spring implementa esse padrão, o que é chamado de EventHandler.

Por exemplo, temos essa classe:

```
10
11 @Component
12 public class AtivacaoClienteService {
13
14     @Autowired
15     @TipoDoNotificador(NivelUrgencia.NORMAL)
16     private Notificador notificador;
17
18     public void ativar(Cliente cliente) {
19         cliente.ativar();
20
21         notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
22     }
23 }
24
25 }
```

uma classe que é injetada por uma dependência do Spring.

O método faz duas coisas, altera o estado do cliente para ativo e notifica que o cadastro está ativo. O que infere em um acoplamento com a interface notificadora e a responsabilidade de notificar um cliente. Uma classe está com uma coesão baixa, temos que ter menos responsabilidades nessa classe, como boas práticas sugerem.

Com o padrão Observer e o tratamento de eventos do Spring, podemos ter menos responsabilidades nessa classe .

```
10
11 @Component
12 public class AtivacaoClienteService {
13
14     // @Autowired
15     // @TipoDoNotificador(NivelUrgencia.NORMAL)
16     // private Notificador notificador;
17
18     public void ativar(Cliente cliente) {
19         cliente.ativar();
20
21         //notificador.notificar(cliente, "Seu cadastro no sistema está ativo");
22     }
23 }
24
25 }
```

Disparando um evento na classe gerenciada pelo Spring, e o evento ficará disponível por toda a aplicação através da interface ApplicationEventPublisher

```

12 @Component
13 public class AtivacaoClienteService {
14
15     @Autowired
16     private ApplicationEventPublisher eventPublisher;
17
18     public void ativar(Cliente cliente) {
19         cliente.ativar();
20
21         eventPublisher.publishEvent(new ClienteAtivadoEvent());
22     }
23
24 }
25

```

o método `publishEvent` aceita como um argumento um `Object`, e esse `Object` ficará disponível para ser capturado através de um Ouvinte.

```

4
5 public class ClienteAtivadoEvent {
6
7     Cliente cliente;
8
9     public ClienteAtivadoEvent(Cliente cliente) {
10         this.cliente = cliente;
11     }
12
13     public Cliente getCliente() {
14         return cliente;
15     }
16
17 }
18

```

a classe foi criada para representar o cliente que foi ativado que recebe um objeto `ar cliente` no construtor

```

9 @Component
10 public class AtivacaoClienteService {
11
12     @Autowired
13     private ApplicationEventPublisher eventPublisher;
14
15     public void ativar(Cliente cliente) {
16         cliente.ativar();
17
18         eventPublisher.publishEvent(new ClienteAtivadoEvent(cliente));
19     }
20
21 }
22

```

Agora não é a classe `AtivacaoClienteService` que tem a responsabilidade de notificar o cliente que ele foi ativado ou qualquer outra notificação, a classe ficou mais coesa com sua responsabilidade única de ativar um cliente

```

8 @Component
9 public class NotificacaoService {
10
11     @EventListener
12     public void clienteAtivadoListener(ClienteAtivadoEvent event) {
13         System.out.println("Cliente " + event.getCliente().getNome() + " agora está ativo no sistema");
14     }
15 }
16

```

Podemos configurar um Ouvinte para receber o evento assim que ele for disparado pelo Spring, tendo sua classe anotada com `@Component` e o método de qualquer tipo e qualquer retorno com qualquer nome com `@EventListener`, recebendo no construtor a instância do objeto instanciado no método de `EventPublisher`

```

12 @Component
13 public class NotificacaoService {
14
15     @Autowired
16     @TipoDoNotificador(NivelUrgencia.NORMAL)
17     private Notificador notificador;
18
19     @EventListener
20     public void clienteAtivadoListener(ClienteAtivadoEvent event) {
21         notificador.notificar(event.getCliente(), "Seu cadastro está ativo no sistema");
22     }
23 }
24

```

A classe AtivacaoClienteService dispara um evento através da interface injetada com @Autowired, chamando o método .publishEvent e passando no argumento do método uma nova instância da classe ClienteAtivadoEvent com um objeto cliente no construtor. E a classe NotificacaoService é um ouvinte que recebe o evento pelo método anotado com @EventListener e recebendo um ClienteAtivadoService no parâmetro do método.

```

2023-02-03 00:49:43.428 INFO 9016 --- [ restartedMain] org.apache.catalina.core.StandardEngine
2023-02-03 00:49:43.451 INFO 9016 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]
2023-02-03 00:49:43.452 INFO 9016 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext
MeuPrimeiroController: com.algaworks.algafood.di.service.AtivacaoClienteService@70713995
2023-02-03 00:49:43.593 INFO 9016 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2023-02-03 00:49:43.607 INFO 9016 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2023-02-03 00:49:43.617 INFO 9016 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication
2023-02-03 00:49:43.621 INFO 9016 --- [ restartedMain] .ConditionEvaluationDeltaLoggingListener
2023-02-03 00:54:05.011 INFO 9016 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2023-02-03 00:54:05.011 INFO 9016 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2023-02-03 00:54:05.012 INFO 9016 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
Notificando João por EmailFake joao@xyz.com : Seu cadastro está ativo no sistema

```

Agora a classe AtivacaoClienteService tem uma alta coesão por conta da sua responsabilidade de somente ativar o cliente e baixo acoplamento com uma implementação de notificador para notificar. Quem agora recebe a responsabilidade de notificar é o evento capturado

Podemos ter mais de um ouvinte recebendo o evento da classe

```

12 @Component
13 public class EmissaoNotaFiscalService {
14
15     @Autowired
16     @TipoDoNotificador(NivelUrgencia.NORMAL)
17     Notificador notificador;
18
19     @EventListener
20     public void emitirNotaFiscalClienteEvent(ClienteAtivadoEvent event) {
21         notificador.notificar(event.getCliente(), "Emitindo nota fiscal");
22     }
23 }

```

```

2023-02-03 01:00:02.524 INFO 10816 --- [ restartedMain] org.apache.catalina.core.StandardEngine
2023-02-03 01:00:02.629 INFO 10816 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]
2023-02-03 01:00:02.630 INFO 10816 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext
MeuPrimeiroController: com.algaworks.algafood.di.service.AtivacaoClienteService@7a605e8c
2023-02-03 01:00:03.181 INFO 10816 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2023-02-03 01:00:03.284 INFO 10816 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer
2023-02-03 01:00:03.311 INFO 10816 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication
2023-02-03 01:00:08.265 INFO 10816 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2023-02-03 01:00:08.265 INFO 10816 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2023-02-03 01:00:08.268 INFO 10816 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
Notificando João por EmailFake joao@xyz.com : Emitindo nota fiscal
Notificando João por EmailFake joao@xyz.com : Seu cadastro está ativo no sistema

```

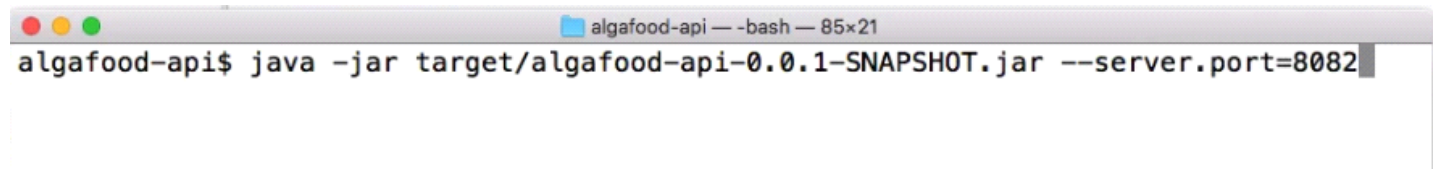
## 2.23. Configurando projetos Spring Boot com o application.properties

sexta-feira, 3 de fevereiro de 2023 08:49

Apenas conhecendo o arquivo de propriedades do Spring

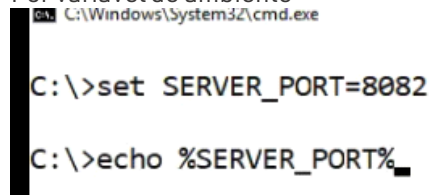
## 2.24. Substituindo propriedades via linha de comando e variáveis de ambiente

sexta-feira, 3 de fevereiro de 2023 08:53



```
algafood-api$ java -jar target/algafood-api-0.0.1-SNAPSHOT.jar --server.port=8082
```

Por variável de ambiente



```
C:\Windows\System32\cmd.exe  
C:\>set SERVER_PORT=8082  
C:\>echo %SERVER_PORT%
```



## 2.25. Criando e acessando propriedades customizadas com @Value

sexta-feira, 3 de fevereiro de 2023

09:25

```
@Value("${notificador.email.smtp}")  
private String hostsmtp;
```

```
notificador.email.smtp=br.com.google|
```

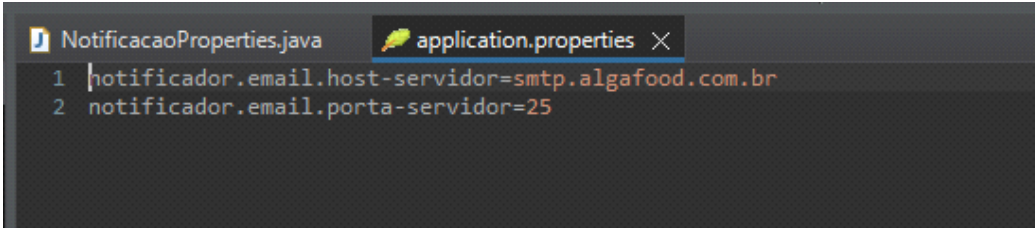
## 2.26. Acessando propriedades com @ConfigurationProperties

sexta-feira, 3 de fevereiro de 2023 09:31

Quando se tem muitas variáveis de configuração para serem acessadas, podemos definir uma classe de propriedades vindas do application.properties utilizando a anotação @ConfigurationProperties

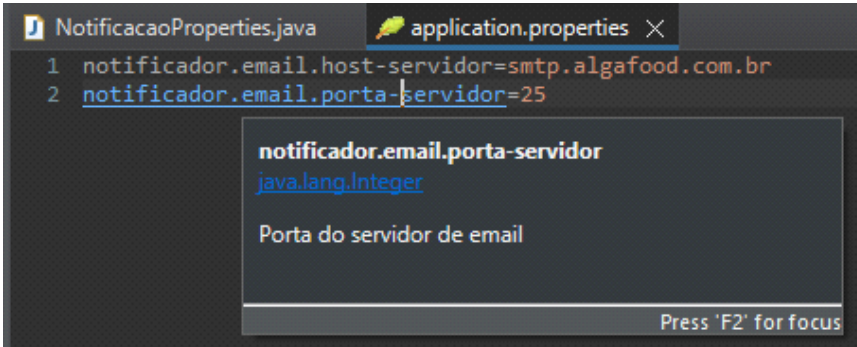
```
5
6 @Component
7 @ConfigurationProperties("notificador.email")
8 public class NotificacaoProperties {
9
10     /**
11      * Host do servidor de email
12      */
13     private String hostServidor;
14
15     /**
16      * Porta do servidor de email
17      */
18     private Integer portaServidor;
19
20     public String getHostServidor() {
21         return hostServidor;
22     }
23     public void setHostServidor(String hostServidor) {
24         this.hostServidor = hostServidor;
25     }
26     public Integer getPortaServidor() {
27         return portaServidor;
28     }
29     public void setPortaServidor(Integer portaServidor) {
30         this.portaServidor = portaServidor;
31     }
32 }
```

Podemos colocar o prefixo no valor da anotação e tudo que conter após o prefixo é adicionado nas variáveis



```
1 notificador.email.host-servidor=smtp.algafood.com.br
2 notificador.email.porta-servidor=25
```

Com o bloco de comentário de doc



```
1 notificador.email.host-servidor=smtp.algafood.com.br
2 notificador.email.porta-servidor=25
```

**notificador.email.porta-servidor**  
[java.lang.Integer](#)  
Porta do servidor de email  
Press 'F2' for focus

Podemos atribuir um valor padrão para as variáveis caso não haja configuração no arquivo properties

```
@Component
@ConfigurationProperties("notificador.email")
public class NotificacaoProperties {

    /**
     * Host do servidor de email
     */
    private String hostServidor;

    /**
     * Porta do servidor de email
     */
    private Integer portaServidor = 50;

    public String getHostServidor() {
        return hostServidor;
    }
}
```

caso haja valor para a propriedade porta-servidor, o valor é substituído

Agora temos uma instância de uma classe de propriedades de notificação

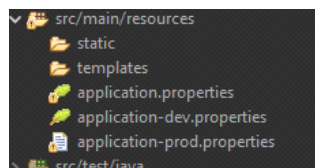
```
// @Value("${notificador.email.smtp}")
// private String hostsmtp

@Autowired
private NotificacaoProperties notificacaoProperties;

@Override
public void notificar(Cliente c, String m) {
    System.out.printf("Notificando %s através do email %s : %s\n", c.getNome(), c.getEmail(), m);
    System.out.println("Host: " + notificacaoProperties.getHostServidor());
    System.out.println("Porta: " + notificacaoProperties.getPortaServidor());
}
```

## 2.27. Alterando a configuração do projeto dependendo do ambiente (com Spring Profiles)

sexta-feira, 3 de fevereiro de 2023 15:53



```
application-dev.properties ×
1 mail.porta-servidor=21
2 mail.host-servidor=br.com.localhost
3

application-prod.properties ×
1 for.email.porta-servidor=81
2 for.email.smtp=br.com.algafood.remote
3

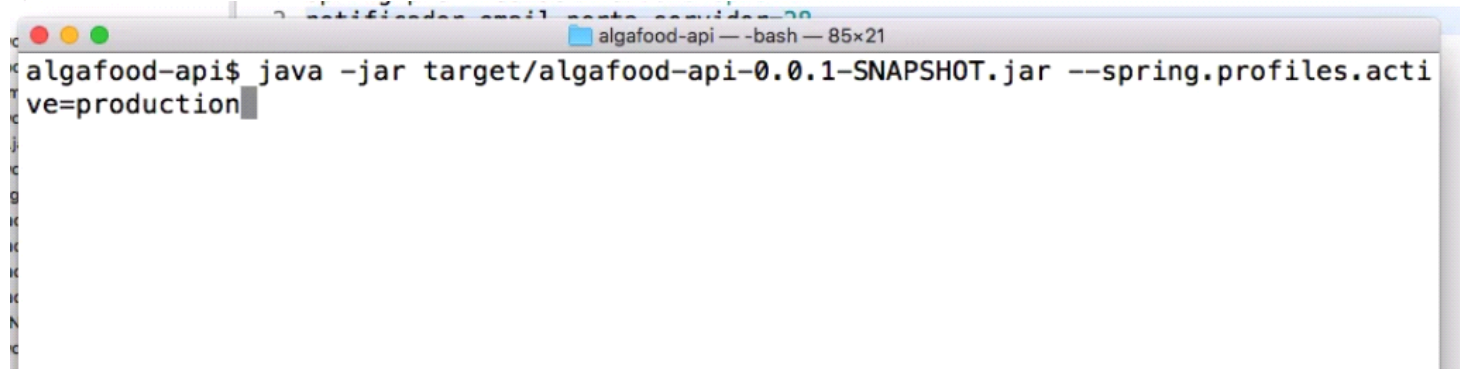
application.properties ×
1 notificador.email.porta-servidor=81
2 notificador.email.smtp=br.com.google
3
4 spring.profiles.active=dev

Problems Javadoc Declaration Console × Progress Terminal Debug
<terminated> algafood-api - AlgafoodApiApplication [Spring Boot App] C:\dev\spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86_64.self-extracting\contents\sts-4.17.2.RELEASE\plugins\org.
2023-02-03 16:03:44.793 INFO 7940 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-02-03 16:03:44.793 INFO 7940 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.71]
2023-02-03 16:03:44.886 INFO 7940 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-02-03 16:03:44.886 INFO 7940 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization comple
MeuPrimeiroController: com.algaworks.algafood.di.service.AtivacaoClienteService@1642daa3
2023-02-03 16:03:50.941 INFO 7940 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-02-03 16:03:51.002 INFO 7940 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with conte
2023-02-03 16:03:51.018 INFO 7940 --- [ restartedMain] c.a.algafood.AlgafoodApiApplication : Started AlgafoodApiApplication in 7.921 seconds (
2023-02-03 16:03:54.433 INFO 7940 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcher
2023-02-03 16:03:54.433 INFO 7940 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-02-03 16:03:54.434 INFO 7940 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Notificando João por SMS através do telefone 3499998888 : Emitindo nota fiscal
Notificando João através do email joao@xyz.com : Seu cadastro está ativo no sistema
2023-02-03 16:04:03.850 INFO 7940 --- [on(7)-127.0.0.1] inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown requested.
2023-02-03 16:04:03.880 INFO 7940 --- [on(7)-127.0.0.1] o.apache.catalina.core.StandardService : Stopping service [Tomcat]
2023-02-03 16:04:03.883 INFO 7940 --- [on(7)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Destroying Spring FrameworkServlet 'dispatcherSer
```

## 2.28. Ativando o Spring Profile por linha de comando e variável de ambiente

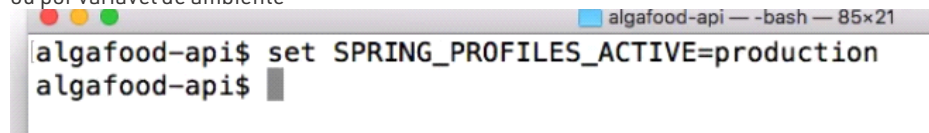
sexta-feira, 3 de fevereiro de 2023 16:11

Rodando perfis de projeto por linha de comando quando não estamos acessando a aplicação

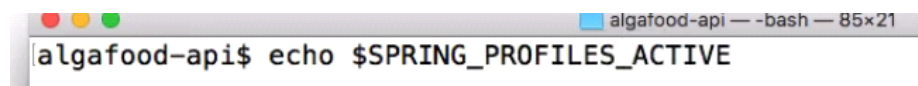


```
algafood-api$ java -jar target/algafood-api-0.0.1-SNAPSHOT.jar --spring.profiles.active=production
```

ou por variável de ambiente



```
algafood-api$ set SPRING_PROFILES_ACTIVE=production
algafood-api$
```



```
algafood-api$ echo $SPRING_PROFILES_ACTIVE
```