Sumário

[Criar o projeto spring 1](#_heading=h.gjdgxs)

[Instalar as dependências 1](#_heading=h.30j0zll)

[Importar o projeto 2](#_heading=h.1fob9te)

[Camada controller 4](#_heading=h.3znysh7)

[Criar a controller de produto 7](#_heading=h.tyjcwt)

[Criar a Model 7](#_heading=h.3dy6vkm)

[Criar a view 8](#_heading=h.4d34og8)

[Dependência: 8](#_heading=h.nu5d31d2d138)

[Incluir o thymeleaf 9](#_heading=h.2s8eyo1)

[Criar a chamada da controller > view 10](#_heading=h.17dp8vu)

[Chamada na controller produto 10](#_heading=h.3rdcrjn)

[Criar a view no documento ”produto.index.hml” 10](#_heading=h.26in1rg)

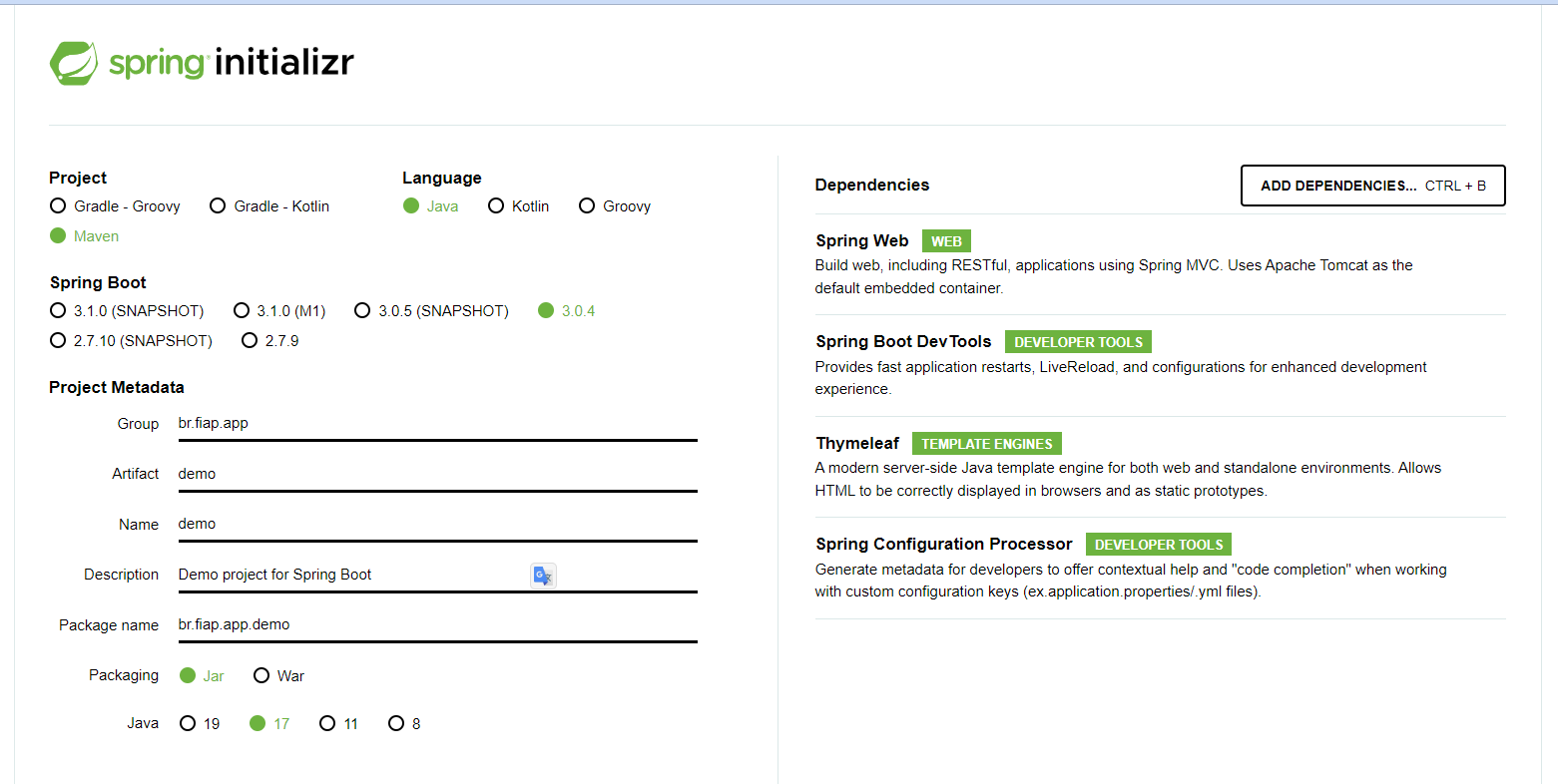
# Criar o projeto spring

1. Acessar <https://start.spring.io/>

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

## Instalar as dependências



Clique em gerar (Generate Ctrl +)

## Importar o projeto

1. Para importar o projeto no eclipse, basta clicar em file > import

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1. Procure a pastinha Maven e clique em Existing Maven Projects

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

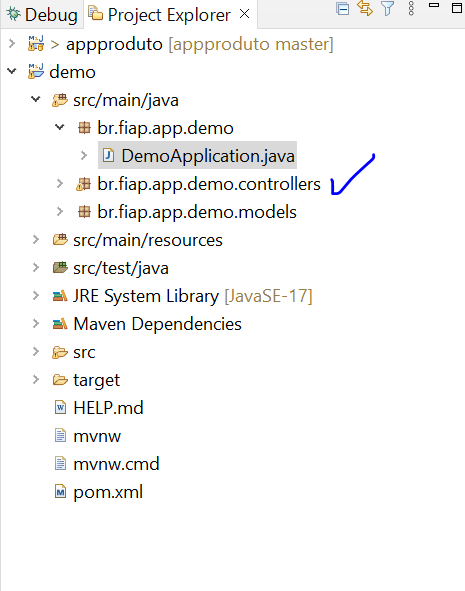
1. Para finalizar, selecione o projeto e clique em finish

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

# Camada controller

Para começarmos, vamos criar um pacote para organizarmos nossas controllers.



Annotations spring MVC

| Annotations | Descrição | Exemplo |
| --- | --- | --- |
| @Controller | Esta anotação é utilizada para definir uma classe como uma controller do Spring MVC |  |
| @RequestMapping  @GetMapping  @PostMapping  @DeleteMapping  @UpdateMapping | Esta anotação é utilizada para mapear uma requisição URL em um método da controller |  |
| @PathVariable | Utilizada para vincular uma variável da URL a um parâmetro no método |  |
| @ModelAttribute | Utilizamos para vincular atributos do model ao método parameter | View  Método para receber parâmetro |
| @ResponseBody | Anotação utilizada para indicar o retornar o valor que deve ser serializado para Json pelo HTTP response Body  Exemplo frontend | Response body |
| @ResponseStatus | Possui o status da requisição |  |

# Criar a controller de produto

Para criar uma controller, basta criar uma class e incluir a anotação @Controller

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente com confiança média

# Criar a Model

1. Vamos criar todas as propriedades que farão parte da classe

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* Não esqueça de realizar os imports e adicionar os getters e setters.

# Criar a view

Antes de falarmos sobre o front, precisamos entender um pouco mais sobre o thymeleaf

## Dependência:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

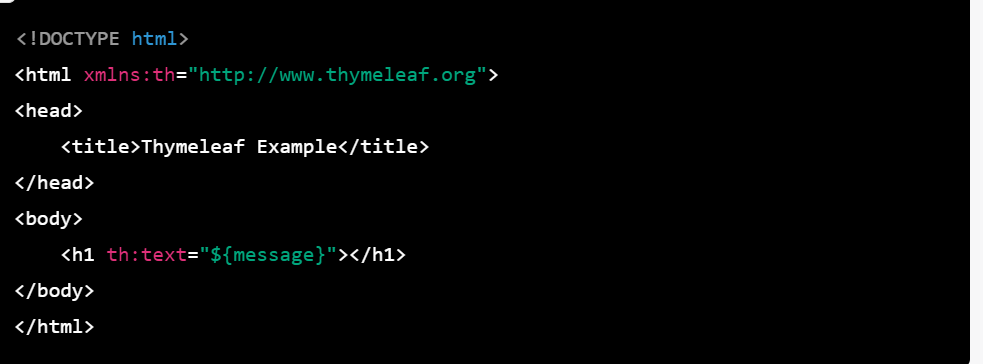
Definição de thymeleaf

Thymeleaf é um mecanismo de modelagem baseado em Java usado para criar páginas da Web dinâmicas.



O thymleaf nos auxilia na integração entre a Controller e a view. Enviando informações do servidor para o cliente.

Exemplo enviando texto do servidor (controller) para o html (view):



As visualizações ficam dentro da pasta chamada *templates*

*Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente*

Aqui podemos inserir páginas HTML

## Incluir o thymeleaf

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

Vamos incluir o HTML normal, como se fosse uma página html simples.

Texto

Descrição gerada automaticamente

# Criar a chamada da controller > view

## Chamada na controller produto

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

## Criar a view no documento ”produto.index.hml”

**Tome nota:**

* O th utilizado é do thymeleaf, ele é um tipo de variável utilizado para receber informações da controller;
* Th:each é uma forma de fazermos interações com objetos enviados pela controller.

Exercícios

1 - Criar a controller, Model e view para clientes.

Premissas:

* Campos: nome, idade e documento;
* Criar a lista de clientes (pode ser dados fakes);
* Criar uma tela para edição.

2 - Dado os seguintes códigos,

1. Cliente inadimplente;
2. Restrições ao crédito;
3. Score OK.

Criar um método que recebe o código e devolva um json com a seguinte estrutura: