

Spring boot

Desenvolvimento Web IV - Prof. Felipe
2021/2

Objetivos do spring framework

O spring surgiu para resolver alguns problemas da plataforma de desenvolvimento JavaEE, tais como:

- Desenvolvimento web enterprise produtivo.
- Foco na lógica de negócio, e menos em configuração.
- Segue a filosofia: **convenção** sobre configuração.
- Oferece um ecossistema de projetos para diferentes tipos de aplicações: desenvolvimento mobile, big data e segurança.

IDE - Spring tools

Spring Tools Suite - <https://spring.io/tools>

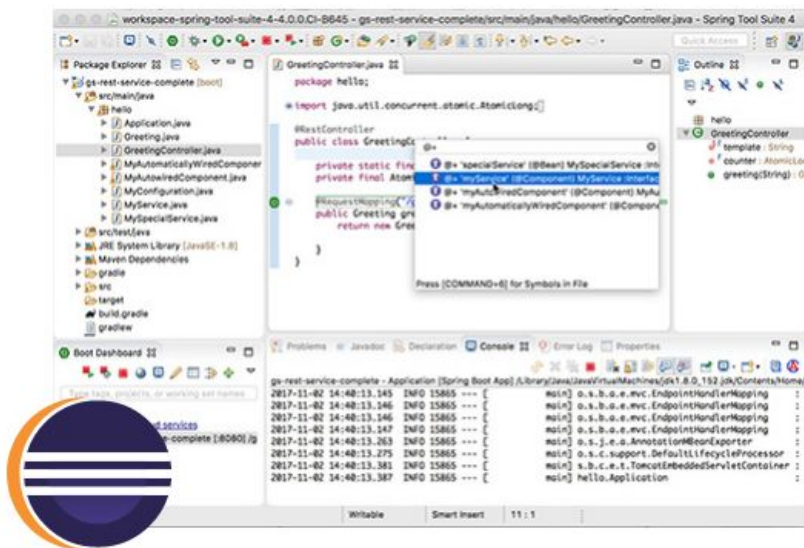
Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.
Free. Open source.

4.9.0 - LINUX 64-BIT

4.9.0 - MACOS 64-BIT

4.9.0 - WINDOWS 64-BIT



Instalação pelo Eclipse

Configuração alternativa se **não** tiver o spring tools.

Configurar e baixar zip: <https://start.spring.io/>

Importar como existing maven project

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools

DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Thymeleaf

TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database

SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Projeto cinema

Criar um projeto spring boot para gerenciamento de filmes e séries.

Cadastro e consulta de filmes e entidades relacionadas:

- Atores, diretores, ...

Exposição da base de dados como serviço usando web services.

Criando um projeto spring

File > New > Spring Starter project

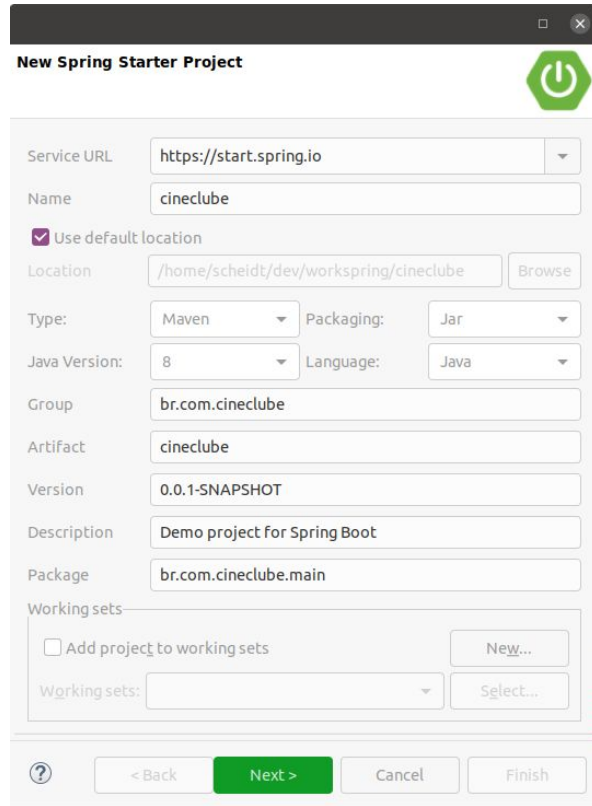
Verificar:

- Java version: 11
- Type: Maven

Group: br.com.cineclube

Artifact: cineclube

Package: br.com.cineclube



The screenshot shows the 'New Spring Starter Project' dialog box. The fields are filled with the following values:

- Service URL: `https://start.spring.io`
- Name: `cineclube`
- Use default location: ☒
- Location: `/home/scheidt/dev/workspring/cineclube`
- Type: `Maven`
- Packaging: `Jar`
- Java Version: `8`
- Language: `Java`
- Group: `br.com.cineclube`
- Artifact: `cineclube`
- Version: `0.0.1-SNAPSHOT`
- Description: `Demo project for Spring Boot`
- Package: `br.com.cineclube.main`

At the bottom, 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 menu and a 'Select...' button. The bottom navigation bar includes a help icon, '< Back', 'Next >' (highlighted in green), 'Cancel', and 'Finish'.

Artifact

É o nome do arquivo jar sem a extensão.

A escolha do nome pode ser aleatória, porém usar somente letra minúscula e sem caracteres especiais.

<https://reitoria.ifpr.edu.br/>

artifact: reitoria

groupID: br.edu.ifpr.reitoria

GroupID

É um identificador (único) do projeto, que permite distingui-lo de outras bibliotecas. A escolha do nome do pacote deve seguir a convenção de nomeação de pacotes do java: Inicia com o nome reverso do domínio da aplicação. Exemplo:

<https://reitoria.ifpr.edu.br/>

groupId: br.edu.ifpr.reitoria

Seleção de dependências (starter)

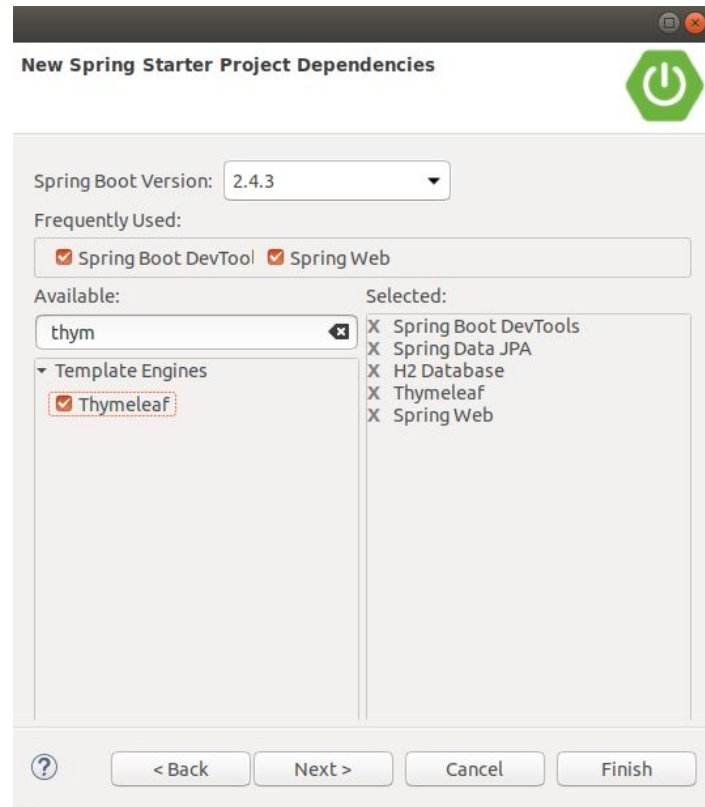
DevTools

JPA

H2

Thymeleaf

Web

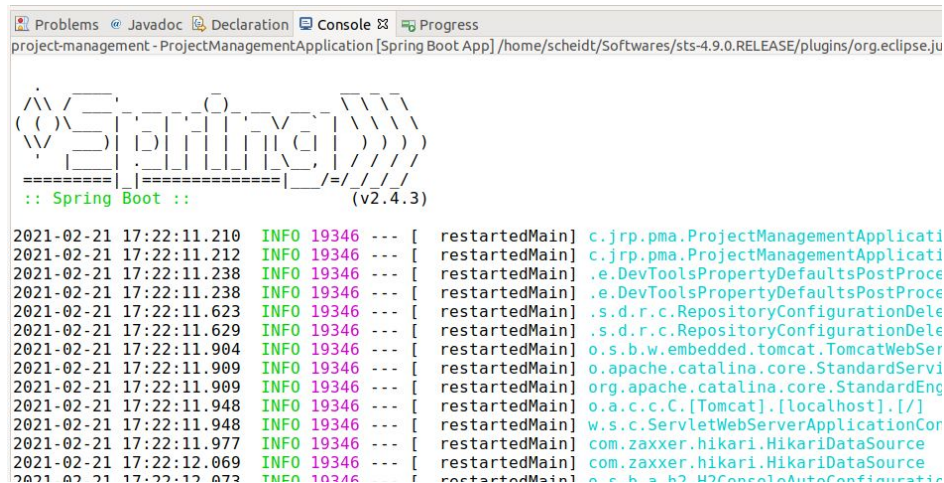
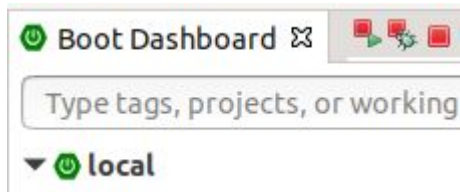


Run and test

Run web server

Accesse:

<http://localhost:8080>



Configuração Maven

- Após a seleção das dependências o assistente de criação do projeto gera o arquivo **pom.xml**
- Esse arquivo de configuração é onde especificamos todas as dependências do projeto.
- Outras bibliotecas devem ser adicionadas neste arquivo.
- Para outras bibliotecas consulte: <https://search.maven.org/>

```
cineclube/pom.xml 83
19  <dependencies>
20    <dependency>
21      <groupId>org.springframework.boot</groupId>
22      <artifactId>spring-boot-starter-data-jpa</artifactId>
23    </dependency>
24    <dependency>
25      <groupId>org.springframework.boot</groupId>
26      <artifactId>spring-boot-starter-thymeleaf</artifactId>
27    </dependency>
28    <dependency>
29      <groupId>org.springframework.boot</groupId>
30      <artifactId>spring-boot-starter-web</artifactId>
31    </dependency>
32
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-devtools</artifactId>
36      <scope>runtime</scope>
37      <optional>true</optional>
38    </dependency>
39    <dependency>
40      <groupId>com.h2database</groupId>
41      <artifactId>h2</artifactId>
42      <scope>runtime</scope>
43    </dependency>
```

Classe starter

@SpringBootApplication

```
CineclubeApplication.java 88
1 package br.com.cineclube.main;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class CineclubeApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(CineclubeApplication.class, args);
11     }
12 }
```

Entidade Filme

Criar uma classe para armazenar as informações do filme:

ID - chave primária (long)

Nome - nome/título do filme

Categoria - drama, ação, .

Ano - ano de lançamento

Nota - nota do filme (float)

```
@Entity
public class Filme {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long filmeId;
    private String nome; //
    private Integer ano; // 2016
    private String categoria; // drama, action, ...
    private Float nota; // 0..10
}
```

Criando a controller

Etapas:

- Anotar a classe FilmeController
 - @Controller
 - @RequestMapping
- Criar um método (newForm) que redireciona uma requisição para uma página específica
- Fazer o *binding* (ligação) entre o objeto newFilme com a página new.html usando o objeto model.

```
@Controller
@RequestMapping("/filmes")
public class FilmeController {

    @RequestMapping("/new")
    public String newForm(Model model) {
        Filme newFilme = new Filme();
        model.addAttribute("filme", newFilme);
        return "new.html";
    }
}
```

Criando o Layout

Template da página

<https://getbootstrap.com/docs/5.0/examples/navbar-fixed/>

Template para formulário:


















<https://getbootstrap.com/docs/5.0/forms/overview/#overview>

Recursos estáticos

Colocar na pasta src/main/resources os arquivos do front-end do projeto: html, css e js

Na sub-pasta static criar duas pastas: css e js

Na sub-pasta templates colocar os arquivos html

- ▼  src/main/resources
 - ▼  static
 - ▼  css
 -  app.css
 -  bootstrap.min.css
 - ▼  js
 -  bootstrap.bundle.min.js
 - ▼  templates
 - ▼  filme
 -  list.html
 -  new.html
 - ▶  pessoa
 -  header.html
 -  index.html
 -  navbar.html
 -  application.properties
- ▶  src/test/java

Criar link css e js no html

Usar th:href

```
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
```

```
<link th:href="@{/css/app.css}" rel="stylesheet" />
```

```
<script type="text/javascript" th:src="@{/js/bootstrap.bundle.min.js}"></script>
```

Passando variáveis para view

Controller:

Usar o objeto Model

```
model.addAttribute("apelido_objeto", referencia)
```

View (html):

```
<span th:text="${today}"></span>
```

Receber parâmetro do formulário

Passar no método da controller:

@RequestParam Type name