

Disciplina Regular 3

# Desenvolvimento Web com Java EE

Graduação em Engenharia de Software - 2020

# Etapa 5 Aula 1

Spring Boot Web Applications

## Etapa 5 Aula 1 - Competências

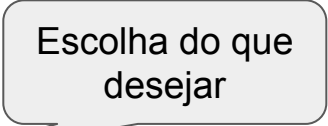
- **Competências** Trabalhadas Nesta Etapa
  - **Compreender e utilizar o Spring Boot:**
    - Criar projetos Spring Boot com o Tomcat Embutido.
    - Implementar Controllers do Spring MVC.

# Etapa 5 Aula 1 - Checklist

- **Checklist DR3:**
  - Ter revisado os slides e **os exercícios**, principalmente **JPA**.
  - Ter implementado exemplos de CRUD com os relacionamentos 1-N e N-M.
    - Construção de telas com JSP + JSTL e EL.
    - Construção de Controllers com validações do Spring MVC.
    - Construção de Repositories do JPA com inserir, alterar, excluir e as consultas.

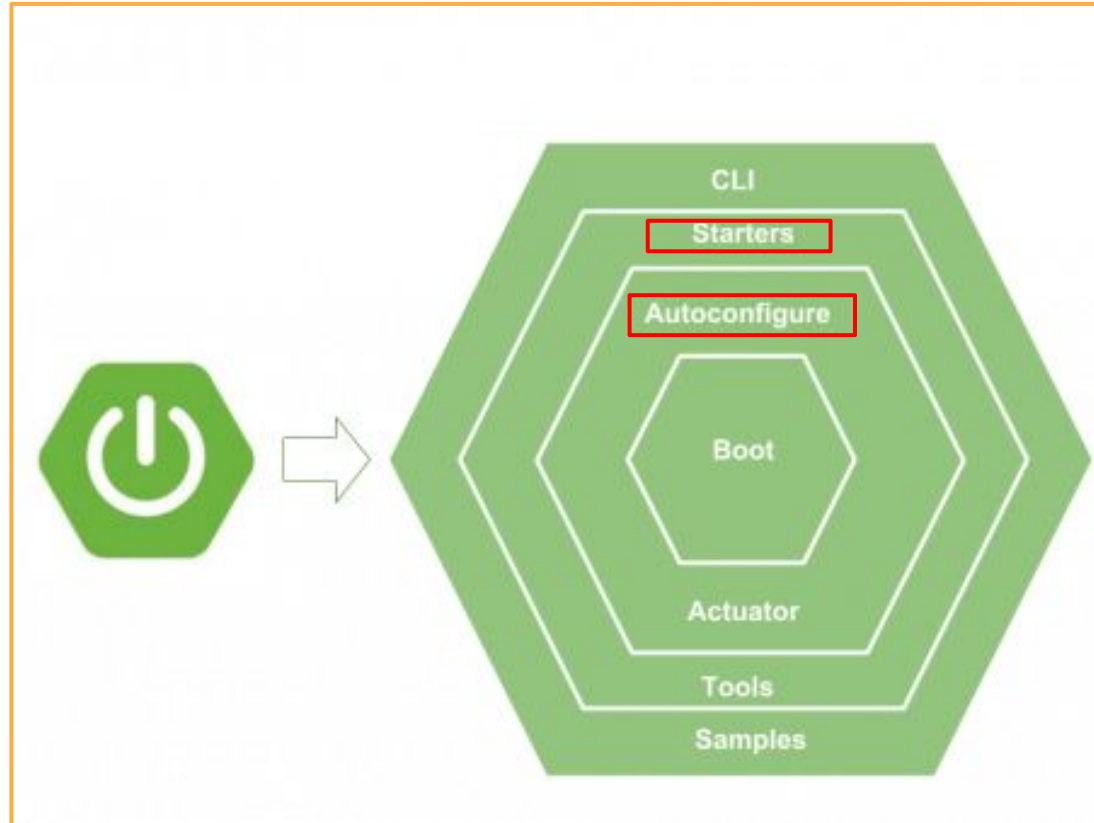
# Spring Boot

# Spring Boot

- O Spring Boot facilita a criação de **aplicativos autônomos** baseados em Spring e com grau de produção onde pode-se "simplesmente executar".

Escolha do que desejar
- Tem-se uma **visão opinativa** da plataforma Spring e de bibliotecas de terceiros para que se possa começar com o mínimo de confusão.
- A maioria dos aplicativos Spring Boot precisa de muito pouca configuração do Spring.

# Spring Boot



# Spring Boot

- Criação de aplicativos Spring independentes e autônomos.
- **Tomcat**, Jetty ou Undertow incorporados diretamente (não há necessidade de implantar arquivos WAR).
- Fornece dependências iniciais para simplificar sua configuração de construção.
- Configuração automática das bibliotecas Spring e de terceiros sempre que possível.
- Fornecimento de recursos prontos para produção, como métricas, verificações de integridade e configuração externalizada.
- Absolutamente nenhuma geração de código e nenhum requisito para configuração em XML.



# Spring Boot

- Requisitos de Sistema do Spring Boot 2.1.8.RELEASE:
  - Java 8 até Java 12.
  - Spring Framework 5.1.9.RELEASE ou superior.
  - Maven 3.3+ ou Gradle 4.4+.
  - Suporta os containeres:
    - Tomcat 9.0 - Jetty 9.4 - Undertow 2.0 .
- O Apache Maven é uma ferramenta de gerenciamento de projetos de software. Com base no conceito de um project object model POM, o Maven pode gerenciar a construção, as dependências e a documentação de um projeto a partir de um repositório central.

# Spring Boot

- **Starters do Spring Boot** são modelos que contêm uma coleção de todas as dependências transitivas relevantes necessárias para iniciar uma funcionalidade específica.
- Por exemplo, se você deseja criar um aplicativo Spring MVC em uma configuração tradicional, você deverá incluir todas as dependências necessárias.
- Spring Boot evita conflitos de versão, que resultam em exceções de tempo de execução.
- Com o Spring boot, para criar um aplicativo MVC, tudo o que você precisa importar é a dependência do **spring-boot-starter-web**.

# Spring Boot

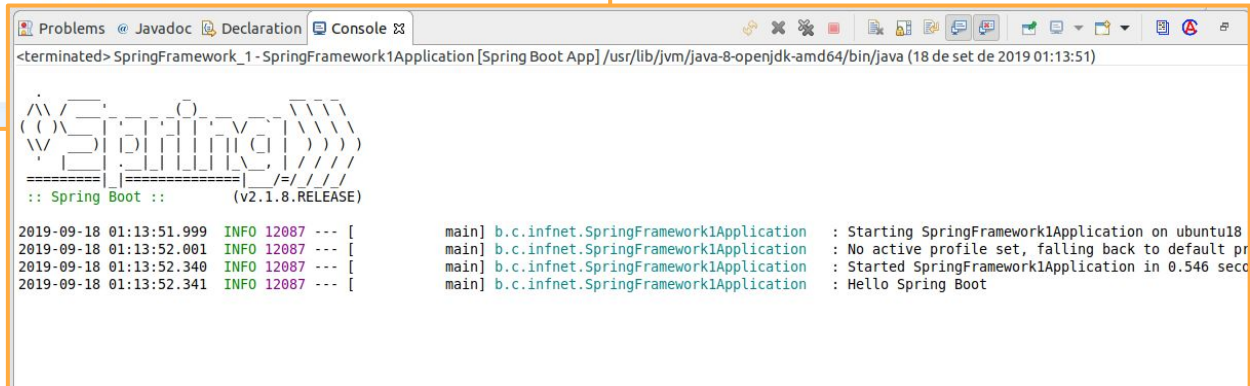
- Para executar um aplicativo, precisamos usar a anotação **@SpringBootApplication**.
- Nos bastidores, isso é equivalente a @Configuration, @EnableAutoConfiguration e @ComponentScan juntos.
  - @Configuration = Indica que existem beans a processar.
  - @EnableAutoConfiguration = Varre o classpath e gera um modelo inteligente de configuração para a aplicação.
  - @ComponentScan = Busca beans nos pacotes da aplicação.
- No exemplo a seguir, a execução começa com o método main (). Ele carrega todos os arquivos de configuração, configura-os e inicializa o aplicativo com base nas propriedades no arquivo **application.properties** que está na pasta / resources.

# Spring Boot

SpringFramework1Application.java

```
1 package br.com.infnet;
2
3 import org.slf4j.Logger;
4
5 @SpringBootApplication
6 public class SpringFramework1Application implements CommandLineRunner {
7
8     private static Logger LOG = LoggerFactory.getLogger(SpringFramework1Application.class);
9
10    public static void main(String[] args) {
11
12        SpringApplication.run(SpringFramework1Application.class, args);
13    }
14
15    @Override
16    public void run(String... args) throws Exception {
17
18        LOG.info("Hello Spring Boot");
19    }
20 }
21
22
23
24
25
```

Dica: coloque a `@SpringBootApplication` no pacote a partir do qual você deseja que os componentes sejam escaneados.



```
<terminated> SpringFramework_1 - SpringFramework1Application [Spring Boot App] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (18 de set de 2019 01:13:51)

:: Spring Boot :: (v2.1.8.RELEASE)

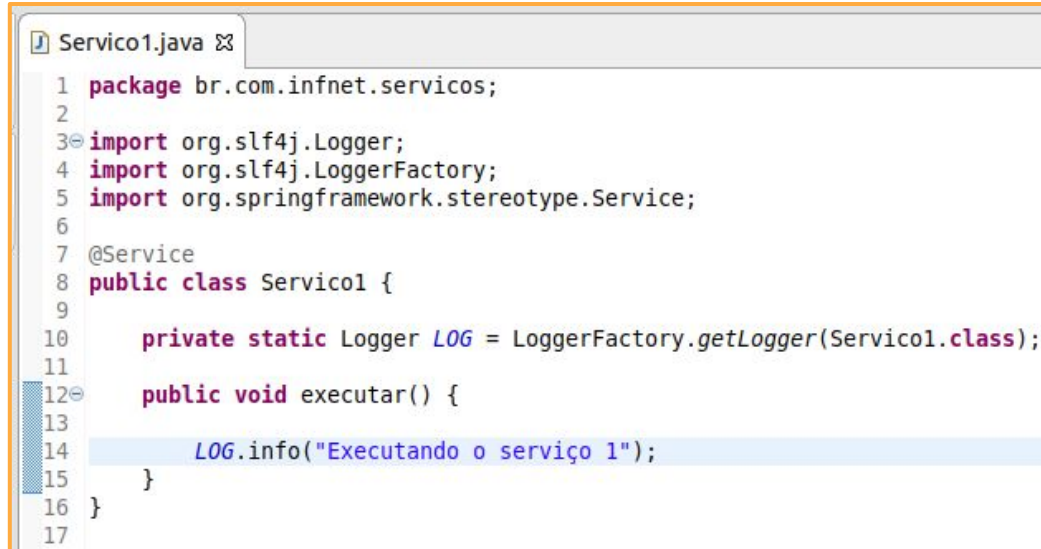
2019-09-18 01:13:51.999 INFO 12087 --- [main] b.c.infnet.SpringFramework1Application : Starting SpringFramework1Application on ubuntu18
2019-09-18 01:13:52.001 INFO 12087 --- [main] b.c.infnet.SpringFramework1Application : No active profile set, falling back to default pr
2019-09-18 01:13:52.340 INFO 12087 --- [main] b.c.infnet.SpringFramework1Application : Started SpringFramework1Application in 0.546 seco
2019-09-18 01:13:52.341 INFO 12087 --- [main] b.c.infnet.SpringFramework1Application : Hello Spring Boot
```

# Exercício

- Crie um projeto e uma classe com o código do primeiro programa Spring Boot.
- Exiba uma mensagem na console (mude a mensagem original).
- Execute a classe gerada.

# Exercício

- Altere o projeto incluindo uma classe de serviço, como mostra a figura.



```
1 package br.com.infnet.servicos;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.stereotype.Service;
6
7 @Service
8 public class Servico1 {
9
10     private static Logger LOG = LoggerFactory.getLogger(Servico1.class);
11
12     public void executar() {
13
14         LOG.info("Executando o serviço 1");
15     }
16 }
17
```

- Utilize o serviço criado na classe principal.

# Application Context

- É o contêiner avançado do Spring.
- Semelhante ao BeanFactory, ele pode carregar definições de beans, ligar beans e distribuir beans mediante solicitação.
- Ele adiciona funcionalidades específicas de aplicações corporativas, como a capacidade de resolver mensagens de texto de um arquivo de propriedades e a capacidade de publicar eventos de aplicativo para ouvintes de eventos interessados.
- Definido pela interface `org.springframework.context.ApplicationContext`.

SpringFramework1Application.java 83

```
1 package br.com.infnet;
2
3 import java.util.Arrays;
15
16 @SpringBootApplication
17
18 public class SpringFramework1Application implements CommandLineRunner {
19
20     private static Logger LOG = LoggerFactory.getLogger(SpringFramework1Application.class);
21
22     @Autowired
23     private Service1 service1;
24
25     @Autowired
26     private ApplicationContext appContext;
27
28     public static void main(String[] args) {
29
30         SpringApplication.run(SpringFramework1Application.class, args);
31     }
32
33     @Override
34     public void run(String... args) throws Exception {
35
36         LOG.info("Hello Spring Boot");
37         // -----
38         service1.executar();
39         // -----
40         String[] beans = appContext.getBeanDefinitionNames();
41         Arrays.sort(beans);
42         for (String bean : beans) {
43
44             LOG.info(bean);
45         }
46     }
47 }
48
```



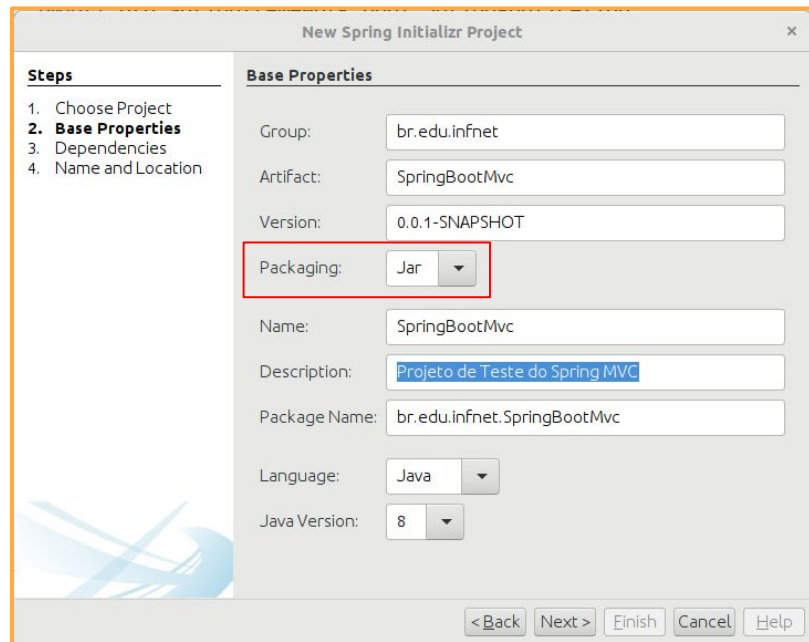
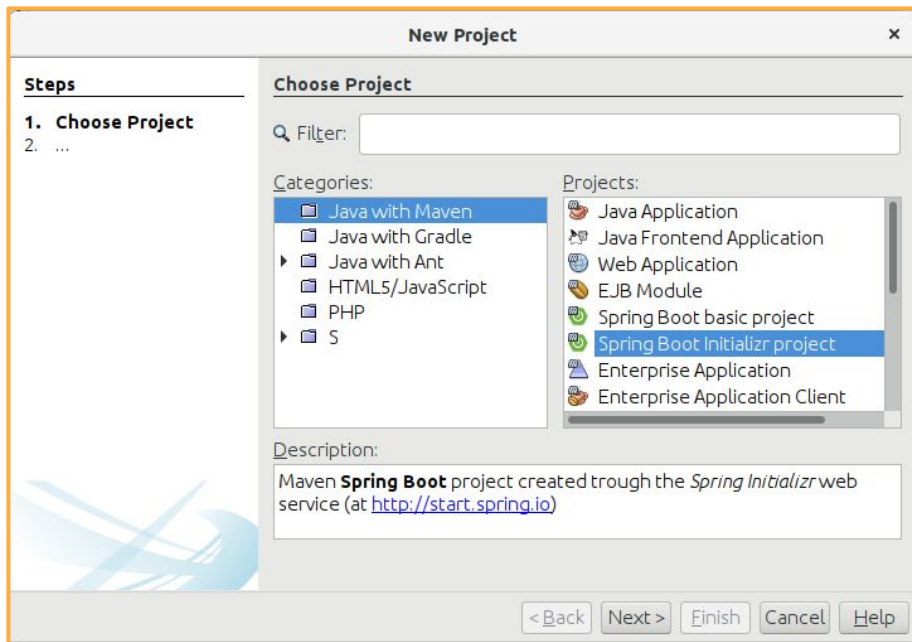
# Exercício

- Altere o projeto para listar os beans que estão registrados no Application Context.

# Spring Boot MVC

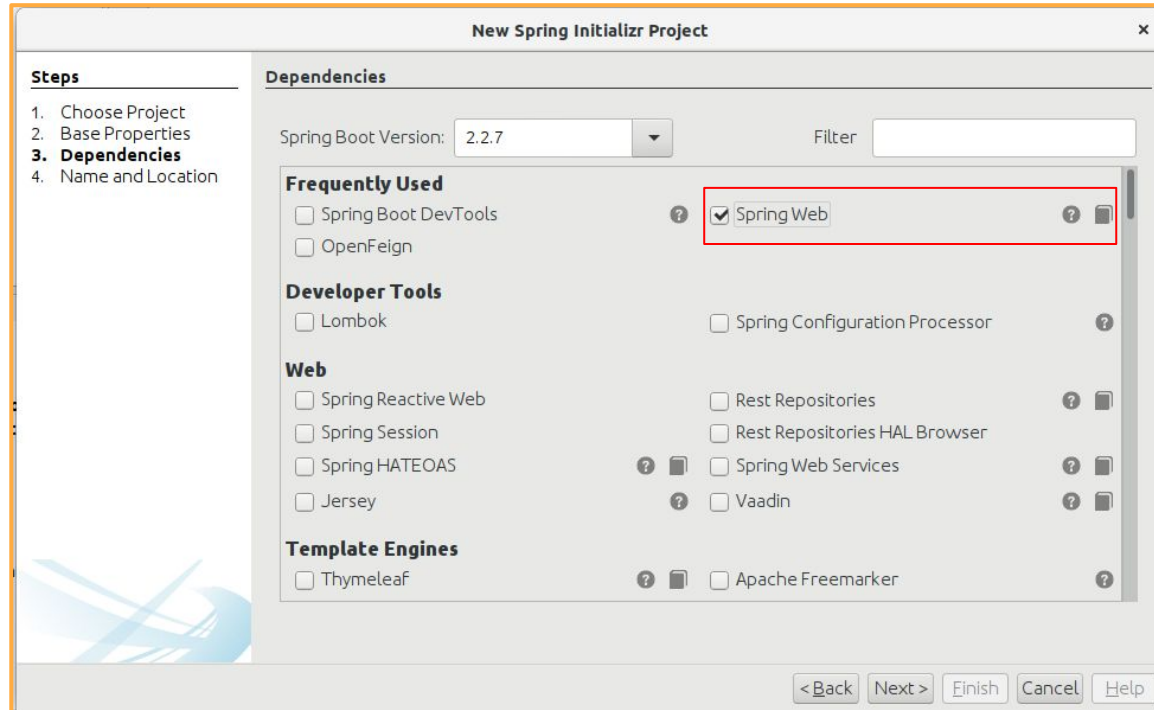
# Receita para o Spring Boot MVC

- Criar um projeto Spring Boot com o Initializr.



# Receita para o Spring Boot MVC

- Criar um projeto Spring Boot com o Initializr.



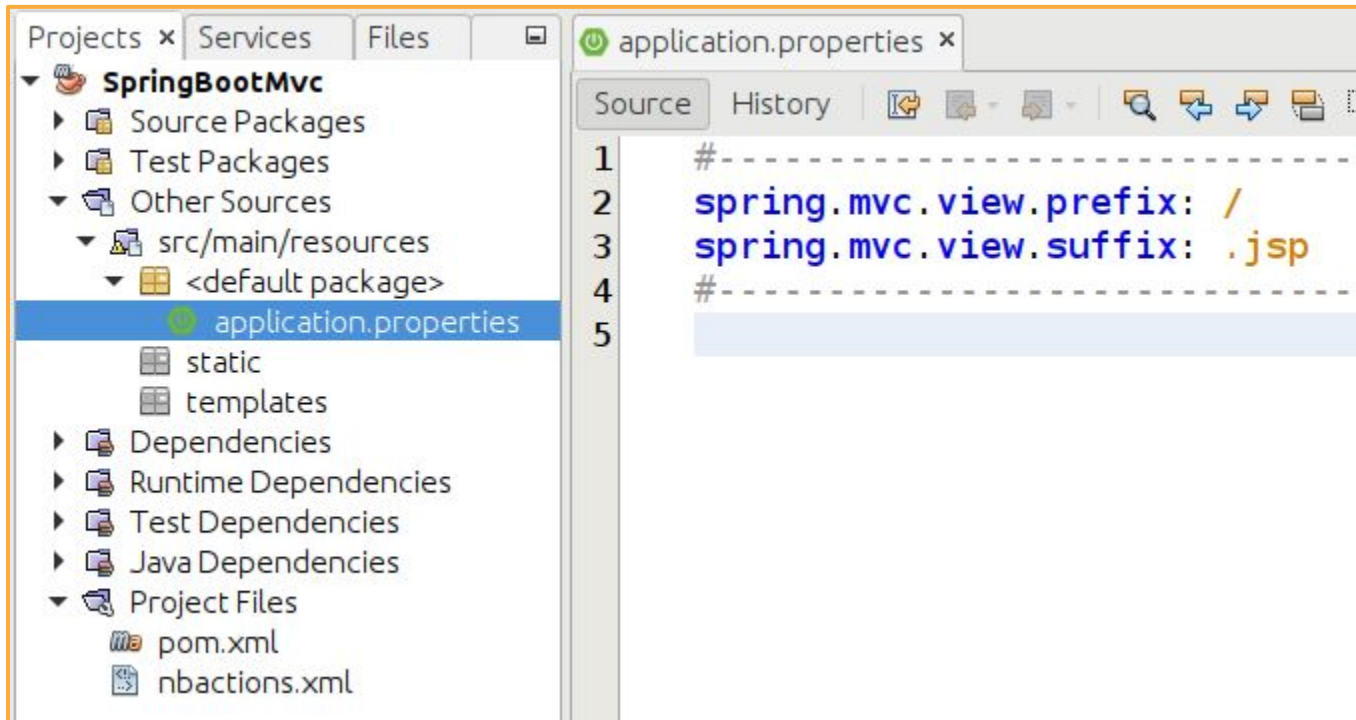
# Receita para o Spring Boot MVC

- Incluir explicitamente o Tomcat no `pom.xml`:

```
38      <!-- ===== -->
39      <dependency>
40          <groupId>org.springframework.boot</groupId>
41          <artifactId>spring-boot-starter-tomcat</artifactId>
42          <scope>provided</scope>
43      </dependency>
44      <dependency>
45          <groupId>org.apache.tomcat.embed</groupId>
46          <artifactId>tomcat-embed-jasper</artifactId>
47          <scope>provided</scope>
48      </dependency>
49      <!-- ===== -->
```

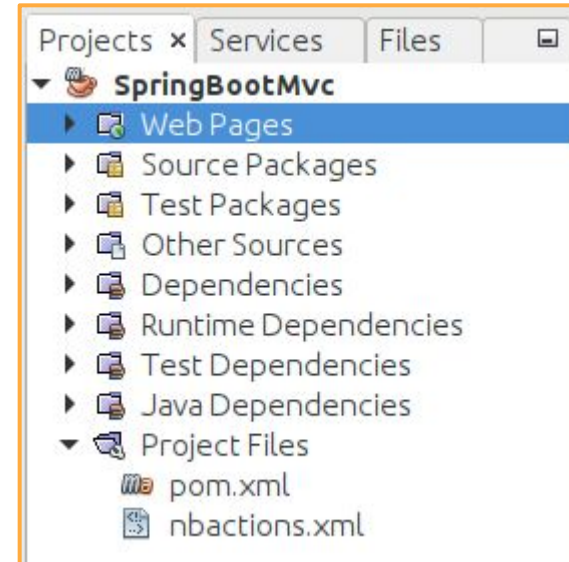
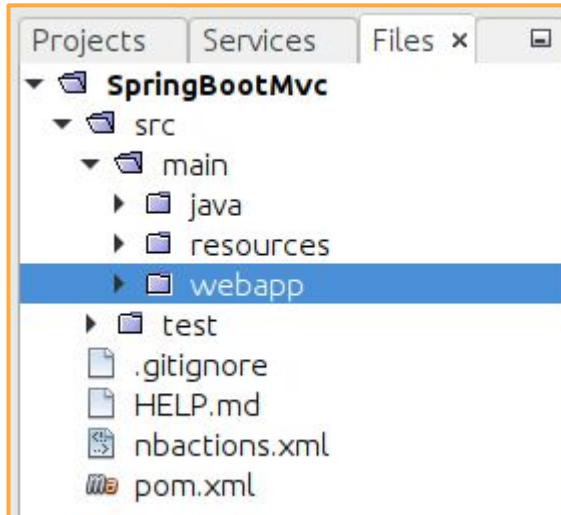
# Receita para o Spring Boot MVC

- Incluir as configurações do MVC no `application.properties`:



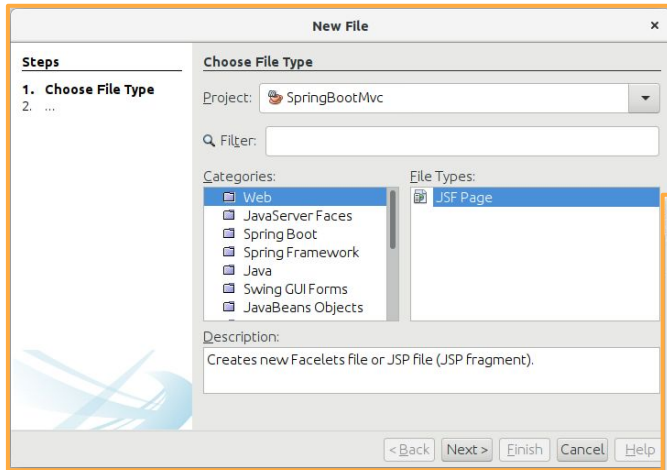
# Receita para o Spring Boot MVC

- Incluir a pasta webapp nos fontes, onde vamos colocar a camada web da aplicação e em seguida faça Clean and Build para o Netbeans reconhecer essa nova pasta:



# Receita para o Spring Boot MVC

- Na pasta webapp é possível incluir JSP, CSS e JS, conforme a necessidade. Para nosso teste, vamos criar um JSP e um CSS:

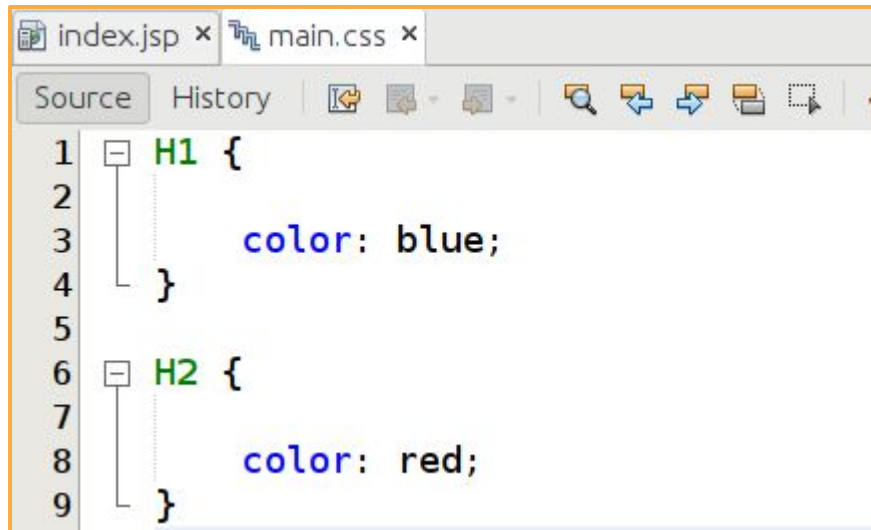
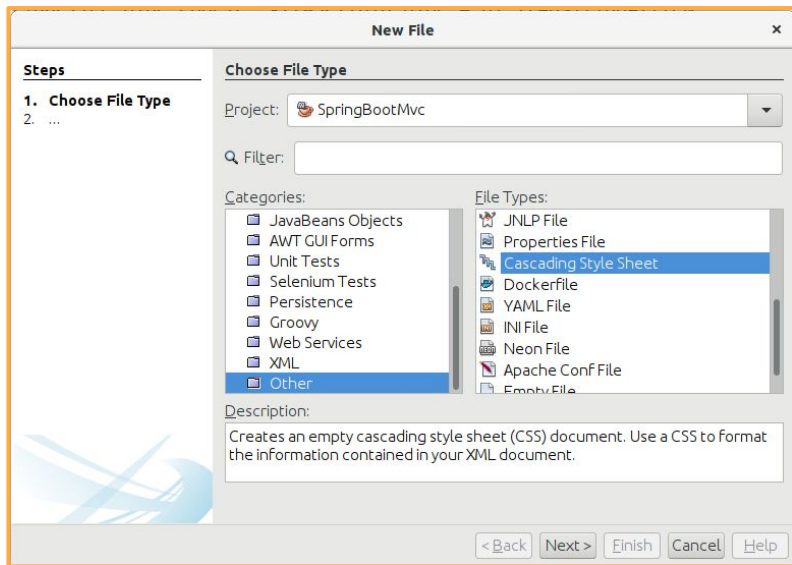


```
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3   "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5   <head>
6     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
7     <title>Spring Boot MVC</title>
8   </head>
9   <body>
10    <h1>Hello Spring Boot MVC World!</h1>
11    <h2>Instituto Infnet</h2>
12    <h3>Spring Boot MVC</h3>
13  </body>
14 </html>
```



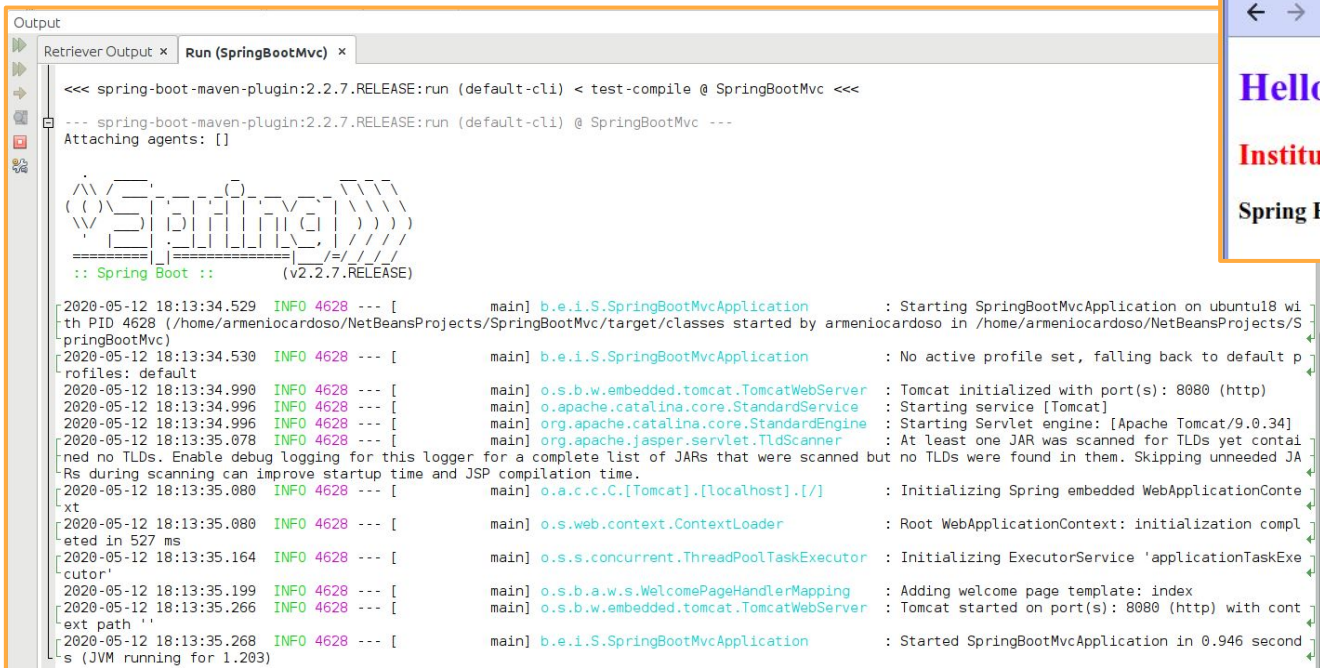
# Receita para o Spring Boot MVC

- Na pasta webapp é possível incluir JSP, CSS e JS, conforme a necessidade. Para nosso teste, vamos criar um JSP e um CSS:



# Receita para o Spring Boot MVC

- Vamos rodar a aplicação e na console vamos notar a inicialização do Tomcat na porta 8080:

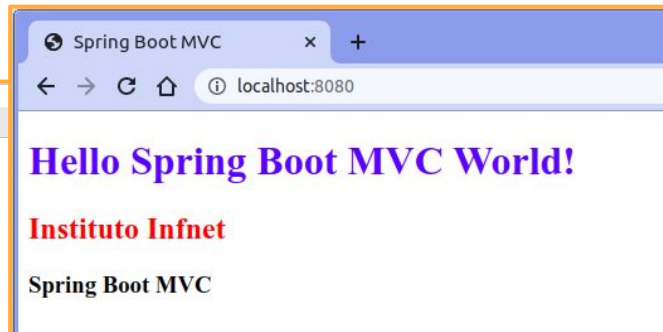


```
Output
Retriever Output x Run (SpringBootMvc) x

<<< spring-boot-maven-plugin:2.2.7.RELEASE:run (default-cli) < test-compile @ SpringBootMvc <<<
--- spring-boot-maven-plugin:2.2.7.RELEASE:run (default-cli) @ SpringBootMvc ---
Attaching agents: []

:: Spring Boot ::
(v2.2.7.RELEASE)

2020-05-12 18:13:34.529 INFO 4628 --- [main] b.e.i.S.SpringBootMvcApplication : Starting SpringBootMvcApplication on ubuntu18 wi
th PID 4628 (/home/armeniocardoso/NetBeansProjects/SpringBootMvc/target/classes started by armeniocardoso in /home/armeniocardoso/NetBeansProjects/S
pringBootMvc)
2020-05-12 18:13:34.530 INFO 4628 --- [main] b.e.i.S.SpringBootMvcApplication : No active profile set, falling back to default p
rofiles: default
2020-05-12 18:13:34.990 INFO 4628 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-05-12 18:13:34.996 INFO 4628 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-05-12 18:13:34.996 INFO 4628 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.34]
2020-05-12 18:13:35.078 INFO 4628 --- [main] org.apache.jasper.servlet.TldScanner : At least one JAR was scanned for TLDs yet contain
ed no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JA
Rs during scanning can improve startup time and JSP compilation time.
2020-05-12 18:13:35.080 INFO 4628 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-05-12 18:13:35.080 INFO 4628 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization compl
eted in 527 ms
2020-05-12 18:13:35.164 INFO 4628 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExe
cutor'
2020-05-12 18:13:35.199 INFO 4628 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page template: index
2020-05-12 18:13:35.266 INFO 4628 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with cont
ext path ''
2020-05-12 18:13:35.268 INFO 4628 --- [main] b.e.i.S.SpringBootMvcApplication : Started SpringBootMvcApplication in 0.946 second
s (JVM running for 1.203)
```





# Exercício

- Crie o projeto básico do Spring MVC com uma página JSP e um arquivo CSS.
  - Rode-o no Netbeans e na console do SO.