2019/2020

Introdução ao Spring Boot

MAM

Introdução ao Spring Boot

Spring Boot torna fácil a criação de um stand-alone, aplicações baseadas no Spring que você pode executar. Você pode utilizar o Spring e suas bibliotecas de um modo mais fácil. As aplicações Spring Boot não precisam de muita configuração.

Você pode usar o Spring Boot para criar aplicações Java que podem ser inicializadas com o comando java -jar ou do modo tradicional que é o deploy do arquivo war. Há também os comandos que executam “spring scripts”.

Os objetivos são:

- Fornece uma rápida inicialização do projeto para os desenvolvedores.
- Que seja opcional as configurações, mas que também seja fáceis para fazê-las.
- Fornece uma grande variedade de recursos não funcionais que são comuns em grandes classes de projeto (tais como servidores incorporados, segurança, métricas, verificações de integridade e configurações externas).
- Não há a necessidade de gerar e configurar um arquivo XML.

Pode-se utilizar o Spring e suas bibliotecas de um modo mais fácil. As aplicações Spring Boot não precisam de muita configuração.

Versões Necessárias

O Spring Boot 2.1.6.RELEASE precisa do Java 8 e é compatível até o Java 11, estes requerimentos também são válidos para o Spring Framework 5.1.8.RELEASE.

As versões necessárias das ferramentas de build são: para o Maven a versão 3.3 ou superior, para o Gradle a versão 4.4 ou superior.

As versões para os servlet containers: Tomcat 9.0 precisa do servlet 4.0, Jetty 9.4 precisa do servlet 3.1 e o Undertow 2.0 precisa do servlet 4.0.

Pode-se fazer um deploy das aplicações com Spring Boot utilizando o servlet 3.1 ou superior.

Instalações

Spring Boot pode ser utilizado da maneira “clássica” com as ferramentas de desenvolvimento Java ou instalado via linha de comando. De qualquer maneira irá precisar do JDK 8 ou maior. Antes de começar deve-se verificar se a instalação do java está correta. Basta abrir o terminal de comando e digitar java -version.

O Spring Boot é compatível com o Apache Maven 3.3 ou superior. Se você não tem o Maven instalado veja o vídeo.

Vídeos: https://www.youtube.com/results?search_query=instalar+maven&sp=CAI%253D

O Spring Boot utiliza o org.springframework.boot e o groupId para as suas dependências. O Maven lê o arquivo POM do projeto onde estão declaradas as dependências.

Aqui está um típico arquivo pom.xml

Baixe o exemplo: https://micheladrianomedeiros.com.br/blog/wp-content/uploads/2020/06/exemploDePom.zip

Vídeos: https://www.youtube.com/results?search_query=arquivo+pom.xml&sp=CAI%253D

Outras Maneiras de Trabalhar com Spring Boot

Spring Boot é compatível com o Gradle 4.4 ou superior.

Vídeos: https://www.youtube.com/results?search_query=instalar+Gradle&sp=CAI%253D

Spring Boot CLI (Command Line Interface) é uma ferramenta que pode ser utilizada para criar rápido protótipos com Spring. Pode executar scripts Groovy.

Você não precisa utilizar o CLI para trabalhar com Spring Boot, mas é o jeito mais rápido de começar a construir uma aplicação.

Vídeos: https://www.youtube.com/results?search_query=instalar+Spring+CLI

Instalação Manual

Você pode fazer o download do Spring CLI direto do repositório:

https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/2.1.7.RELEASE/spring-boot-cli-2.1.7.RELEASE-bin.zip

https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/2.1.7.RELEASE/spring-boot-cli-2.1.7.RELEASE-bin.tar.gz

As outras versões estão aqui https://repo.spring.io/snapshot/org/springframework/boot/spring-boot-cli/

Depois do download é só seguir as instruções do arquivo INSTALL.txt. Você pode executar o spring.bat para Windows que está na pasta bin. Você pode também utilizar o comando java -jar no arquivo do Spring que está na pasta lib.

Instalação com SDKMAN

SDKMAN! (The Software Development Kit Manager) pode ser utilizado para manipular múltiplas versões, incluindo Groovy e o Spring Boot CLI. Faça o download do SDKMAN! https://sdkman.io/. A instalação para o Windows achei bem confusa e não consegui fazer.

Vídeos: https://www.youtube.com/results?search_query=Instala%C3%A7%C3%A3o+com+SDKMAN&sp=CAI%253D

OSX Homebrew Installation

Para o MAC https://docs.spring.io/spring-boot/docs/2.1.7.RELEASE/reference/html/getting-started-installing-spring-boot.html#getting-started-homebrew-cli-installation

Vídeos: https://www.youtube.com/results?search_query=OSX+Homebrew+Installation&sp=CAI%253D

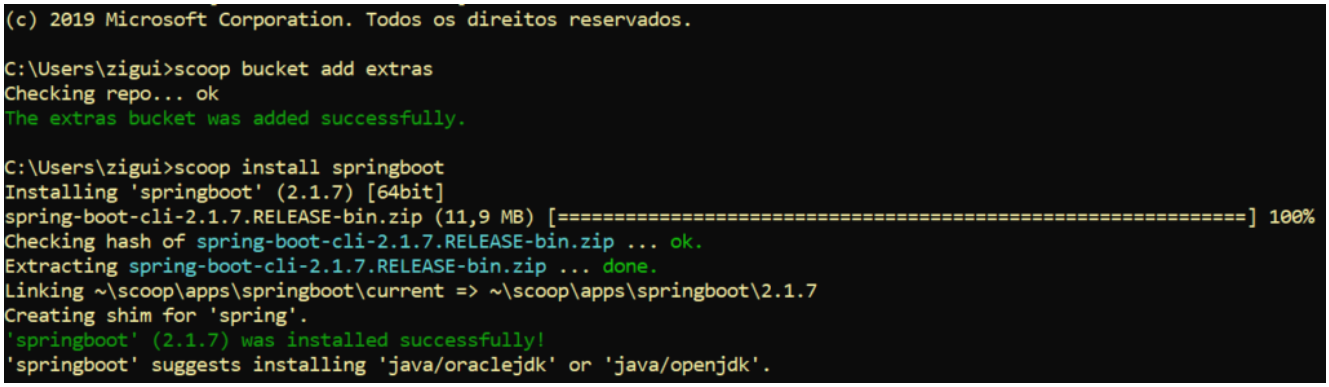
Instalação com o Windows Scoop

Você pode instalar o Spring Boot CLI utilizando o Scoop. Para isto acesse https://scoop.sh/, no site tem um vídeo mostrando como deve ser instalado o Scoop.

Agora faça os comandos:

scoop bucket add extras

scoop install springboot



O Scoop é instalado na pasta do seu usuário. No meu caso C:\Users\zigui\scoop.

Um Exemplo Rápido com o Spring CLI

Vamos criar um arquivo com o nome de app.groovy. Dentro do arquivo coloque o seguinte.

```
@RestController
class ThisWillActuallyRun {
  @RequestMapping("/")
  String home() {
    "Hello World!"
  }
}
```

Agora execute o comando: spring run app.groovy

A primeira vez que executar vai demorar porque tem que verificar todas as dependências, fazer alguns downloads, procedimentos comuns do Spring.

Se deu tudo certo é só acessar http://localhost:8080/.

É para aparecer a mensagem: Hello World!

Desenvolvendo Sua Primeira Aplicação Spring Boot

Antes de começarmos, abra um terminal e vamos verificar a versão do Java e do Maven.

C:\Users\zigui>java -version

java version “1.8.0_241”

Java(TM) SE Runtime Environment (build 1.8.0_241-b07)

Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)

C:\Users\zigui>mvn -v

Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T16:00:29-03:00)

Maven home: C:\NovosProgramas\apache-maven-3.6.1\bin\..

Java version: 13.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-13.0.1

Default locale: pt_BR, platform encoding: Cp1252
OS name: “windows 10”, version: “10.0”, arch: “amd64”, family: “windows”

Criando o POM

Vamos criar um arquivo chamado pom.xml. O pom.xml é a receita que é utilizada para construir o projeto. Abra um editor de texto e digite o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>myproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.4.RELEASE</version>
  </parent>
  <description/>
  <developers>
    <developer/>
  </developers>
  <licenses>
    <license/>
  </licenses>
  <scm>
    <url/>
  </scm>
  <url/>
  <!-- Additional lines to be added here... -->
</project>
```

Criando o POM

Para testar você pode executar o comando mvn package. Em uma IDE não há necessidade de fazer o comando.

Adicionando as Dependências Classpath

Spring Boot fornece um número de “Iniciadores” que permite que você adicione jars para o seu classpath.

Nossas aplicações para teste de fumaça utiliza o spring-boot-starter-parent dentro da seção parent do POM.

O spring-boot-starter-parent é um iniciador especial que fornece configurações padrões do Maven.

Também fornece uma seção dependency-management que você pode omitir a tag version.

Outros “Iniciadores” fornecem dependências que você precisará quando desenvolver um tipo específico de aplicação.

Se formos desenvolver uma aplicação web, nós precisaremos adicionar a dependência spring-boot-starter-web.

Antes disso execute o comando: mvn dependency:tree

Adicionando as Dependências Classpath

```
[INFO]
[INFO] ---------< com.example:myproject >---------
[INFO] Building myproject 0.0.1-SNAPSHOT
[INFO] ----------[ jar ]----------
[INFO]
[INFO] --- maven-dependency-plugin:3.1.1:tree (default-cli) @ myproject ---
[INFO] com.example:myproject:jar:0.0.1-SNAPSHOT
[INFO] ------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------
[INFO] Total time:  12.521 s
[INFO] Finished at: 2020-02-04T09:35:01-03:00
[INFO] ------------------------------------
```

O comando mvn dependency:tree imprime a árvore de dependência do seu projeto. Você pode ver as dependências que o spring-boot-starter-parent fornece.

Para adicionar as dependências necessária, edite o seu pom.xml e adicione a dependência spring-boot-starter-web abaixo da seção parent.

Adicionando as Dependências Classpath

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

Execute novamente o mvn dependency:tree e verá que será adicionada mais dependências no projeto como o Tomcat web server e o Spring Boot.

Escrevendo o Código

Para terminar nossa aplicação, nós precisamos criar um arquivo java. Por padrão, o Maven compila o código fonte da pasta src/main/java, então nós precisamos criar essa estrutura e depois criar um arquivo java.

No fim o caminho desse arquivo ficará assim: src/main/java/Example.java. Vamos criar o arquivo Example com o seguinte código:

```
import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.web.bind.annotation.*;
@RestController
@EnableAutoConfiguration
public class Example {
  @RequestMapping("/")
  String home() {
    return "Hello World!";
  }
  public static void main(String[] args) {
    SpringApplication.run(Example.class, args);
  }
}
```

Apesar de ter pouco código, há muita coisa acontecendo. Iremos ver as partes principais.

As Anotações @RestController e @RequestMapping

A primeira anotação da nossa classe Example é a @RestController. Isso é conhecido como uma anotação estereotipa (stereotype annotation).

Ela fornece dicas para as pessoas que estão lendo o código e para o Spring é fornecido uma regra específica.

Nesse caso, nossa classe é uma web @Controller, o Spring entende que essa classe irá manipular requisições vinda da web.

A anotação @RequestMapping fornece informações sobre o “roteamento”. No Spring qualquer requisição HTTP com o caminho / é mapeado pelo método home.

A anotação @RestController diz ao Spring para renderizar o resultado e retornar para o requisitor.

As anotações @RestController e @RequestMapping são anotações do Spring MVC e não do Spring Boot.

A Anotação @EnableAutoConfiguration

O segundo nível de anotação é o @EnableAutoConfiguration. Essa anotação diz ao Spring Boot “se virar” com a configuração do Spring, baseados nas dependências que você adicionou.

O spring-boot-starter-web adiciona o Tomcat e o Spring MVC, a autoconfiguração assume que você é um desenvolvedor web e configura o Spring de acordo com esse cenário.

Iniciadores e Autoconfiguração

Autoconfiguração é projetado para trabalhar bem com os “Iniciadores”, mas os dois conceitos não estão ligados.

Você é livre para escolher dependências externas dos iniciadores. Ainda assim o Spring Boot fará a melhor configuração para a sua aplicação.

O Método “main”

A parte final da nossa aplicação é o método main. Esse é apenas o método padrão do Java para inicializar uma aplicação.

Nosso método main é controlado pela classe SpringApplication do Spring Boot que executa o run.

SpringApplication sobe a nossa aplicação, iniciando o Spring, o qual inicia a autoconfiguração do Tomcat web server.

Nós precisamos passar Example.class como argumento para o método run para dizer ao SpringApplication que é o primeiro componente Spring.

O args array é também passado para expor quaisquer argumentos da linha de comando.

Revendo os Arquivos

Vamos rever os arquivos que criamos:

Arquivo pom.xml - faça o download: https://micheladrianomedeiros.com.br/blog/wp-content/uploads/2020/06/pom.zip

Revendo os Arquivos

O arquivo Example.java deve ficar na pasta src/main/java.Example.java.

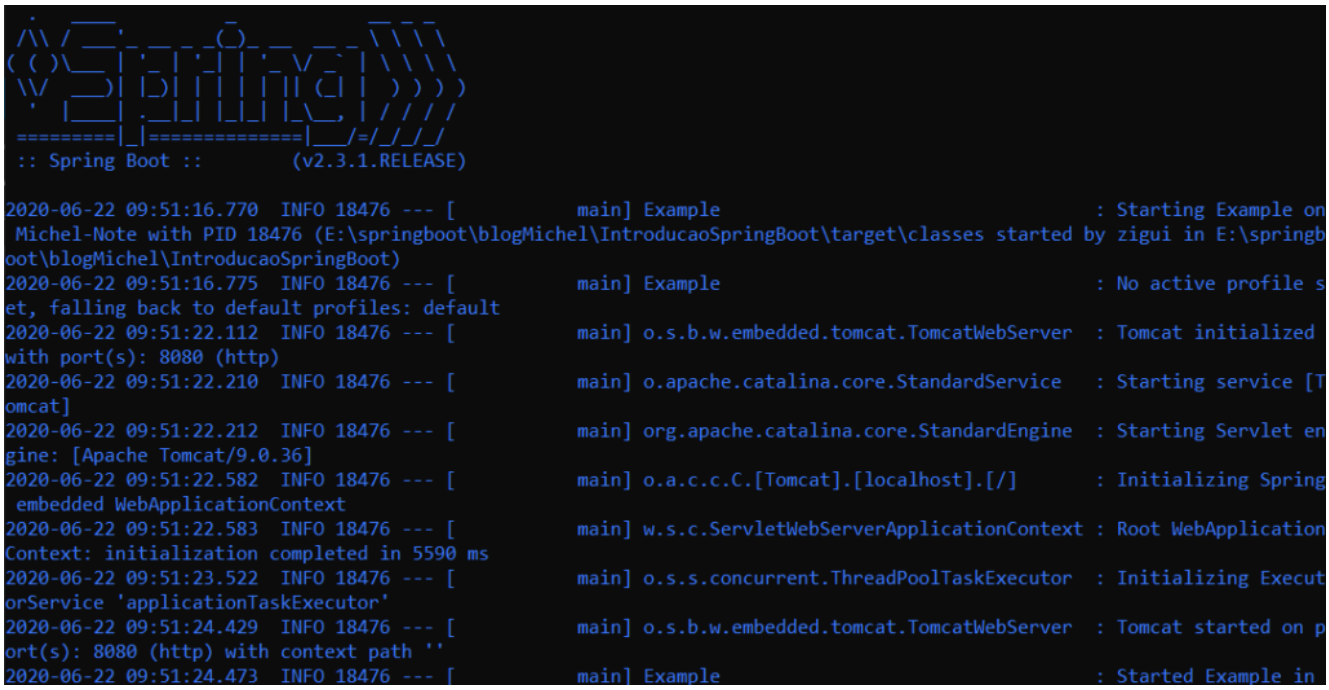
Example.java

```
import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.web.bind.annotation.*;
@RestController
@EnableAutoConfiguration
public class Example {
  @RequestMapping("/")
  String home() {
    return "Hello World!";
  }
  public static void main(String[] args) {
    SpringApplication.run(Example.class, args);
  }
}
```

Nesse ponto a aplicação deve estar funcionando. Desde que você usou o spring-boot-starter-parent POM, você tem que usar o run para executar a aplicação.

Execute o comando mvn spring-boot:run e você deve ver algo do tipo:

Revendo os Arquivos



Se você abrir o seu navegador no endereço http://localhost:8080/ verá a mensagem:

Hello World!

Para parar o aplicativo pressione as teclas: CTRL+C.

Criando e Executando um JAR

Para terminar o exemplo que criamos, temos que criar um executável com a extensão jar. Executáveis jars(algumas vezes chamado de “fat jars”) são arquivos que contém classes compiladas com dependências que precisam para o seu código ser executado.

Executáveis jars e Java

Java não fornece um jeito padrão para carregar arquivos jar aninhados(arquivos jar que contém dentro dele um jar). Isso pode ser um problema se você quer distribuir uma aplicação que seja independente.

Para resolver esse problema, muitos desenvolvedores utilizam “uber” jars. Um pacote uber jar contém todas as classes para sua aplicação ter um simples arquivo.

O problema com essa abordagem é que se torna difícil ver quais bibliotecas estão dentro da sua aplicação. Isso pode ser um problema se o arquivo com o mesmo nome, for utilizado(em diferentes projetos) em múltiplos jars.

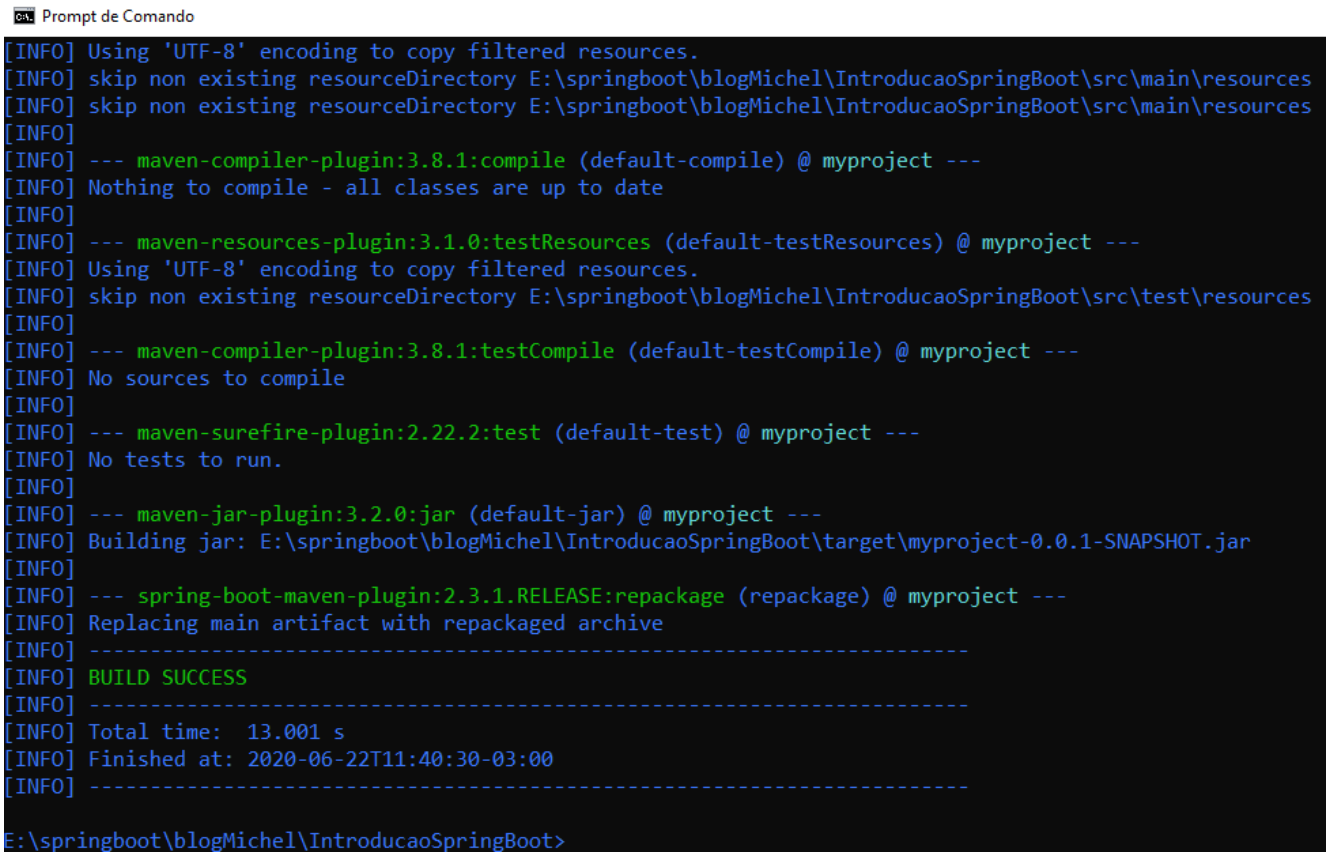
Para criar um executável jar, precisamos adicionar o spring-boot-maven-plugin ao nosso pom.xml.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

O spring-boot-starter-parent POM inclui configurações (<executions>) para ligar ao repackage. Se você não usar o parent POM, você precisa declarar essa configuração.

Executáveis jars e Java

Execute o comando mvn package:



Se você for na pasta target do seu projeto, você verá o arquivo myproject-0.0.1-SNAPSHOT.jar. O arquivo deve ter um tamanho mais ou menos de 16 MB.

Se quiser ver o interior do arquivo faça o comando:

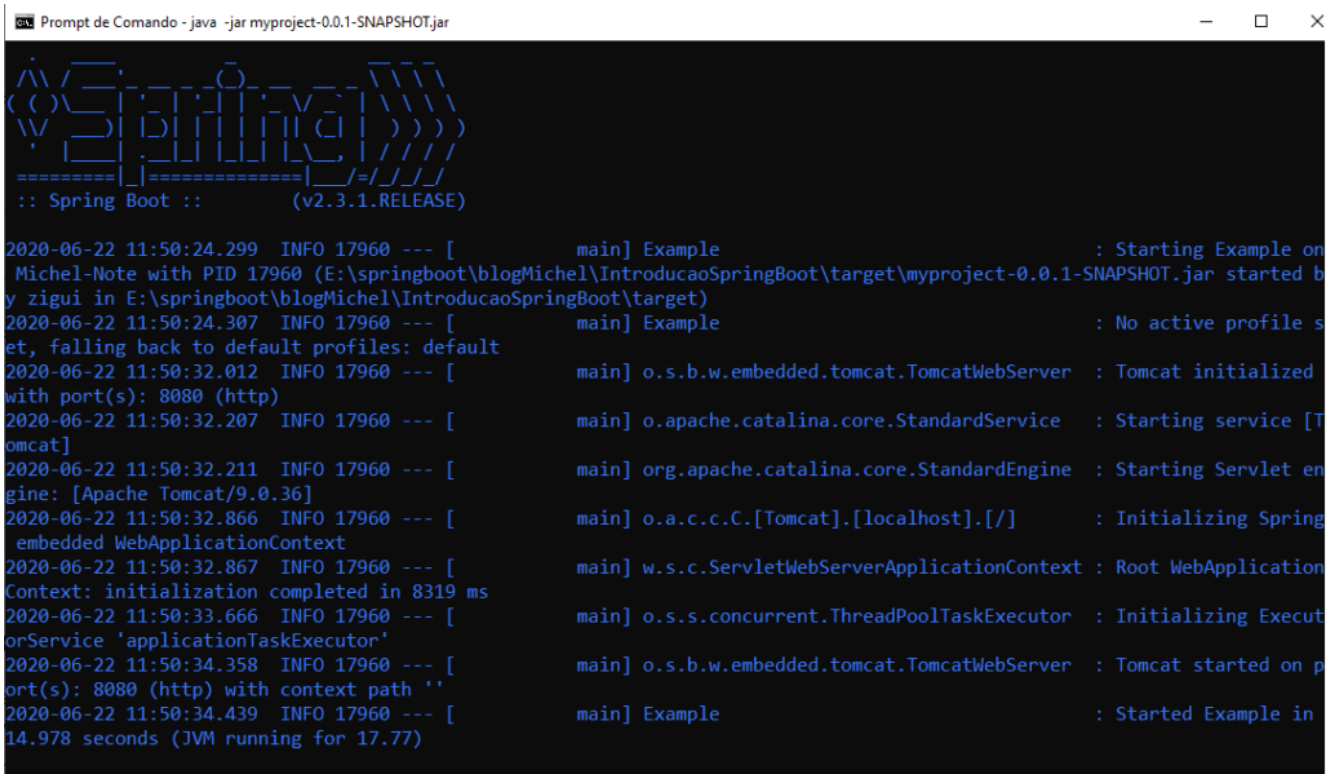
jar tvf myproject-0.0.1-SNAPSHOT.jar

Executáveis jars e Java

Há um outro arquivo chamado myproject-0.0.1-SNAPSHOT.jar.original que está na pasta target. Isso é um arquivo jar original que o Maven cria antes ser reempacotado pelo Spring Boot.

Para executar o arquivo criado execute o comando:

java -jar myproject-0.0.1-SNAPSHOT.jar



Para parar a aplicação pressione as teclas CTRL+C.

Acesse o site: https://cursojavanow.com.br/