

Spring Boot





Spring Boot



Primeiro projeto



Spring boot



Spring Boot

O Spring Boot é um framework Java open source que tem como objetivo facilitar esse processo em aplicações Java. Consequentemente, ele traz mais agilidade para o processo de desenvolvimento, uma vez que devs conseguem reduzir o tempo gasto com as configurações iniciais.

O Spring Boot ajuda você a criar aplicativos autônomos baseados em Spring de nível de produção que podem ser executados.



Principais objetivos

- Fornece uma experiência inicial radicalmente mais rápida e amplamente acessível para todo o desenvolvimento do Spring.
- Seja opinativo fora da caixa, mas saia do caminho rapidamente, pois os requisitos começam a divergir dos padrões.
- Fornece uma variedade de recursos não funcionais que são comuns a grandes classes de projetos (como servidores incorporados, segurança, métricas, verificações de saúde e configuração externalizada).
- Absolutamente nenhuma geração de código e nenhum requisito para configuração XML.

Inversão de Controle

Inversão de controle (Inversion of Control ou IoC, em inglês) trata-se da interrupção do fluxo de execução de um código retirando, de certa forma, o controle sobre ele e delegando-o para uma dependência ou container. O principal propósito é minificar o acoplamento do código.

No Java, falamos mesmo em desacoplamento das classes. Isso permite um ganho enorme em manutenibilidade, além da facilidade de trocar ou acrescentar comportamentos ao sistema, se necessário. Também diminui a possibilidade de ocorrência bugs em cascata.

No Spring Framework, as instâncias das classes são fracamente acopladas, ou seja, a interdependência entre os objetos é mínima.

Injeção de Dependência

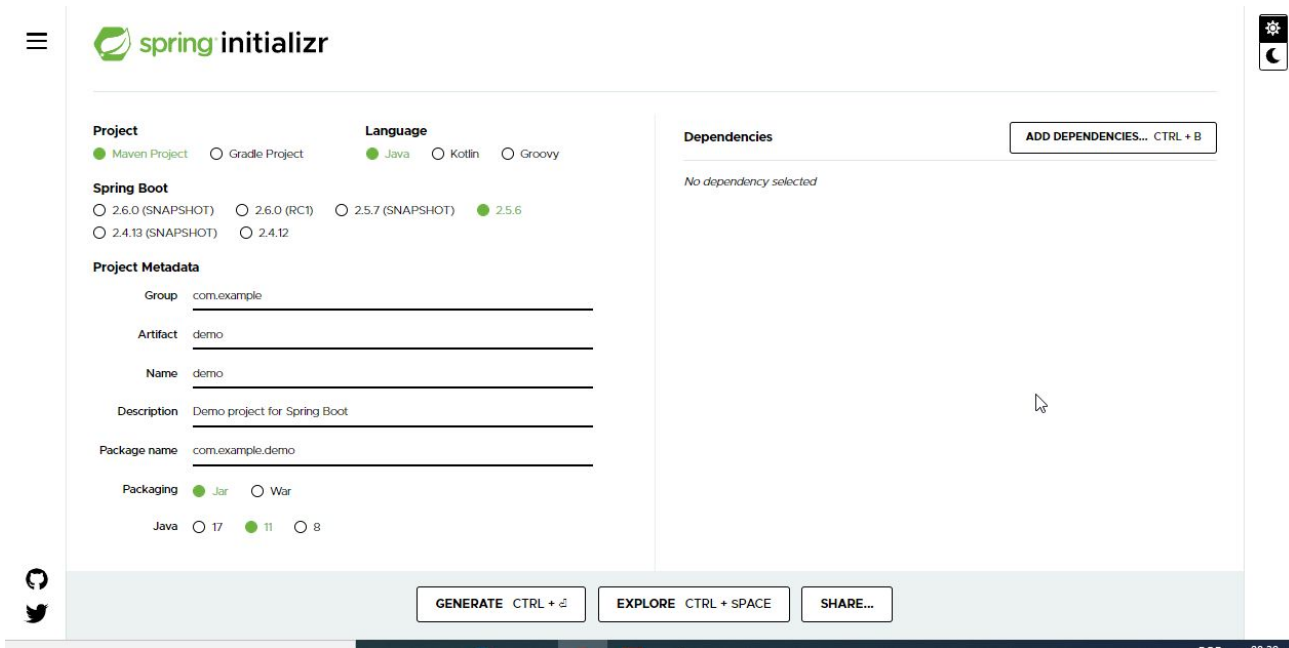
A injeção de dependência tem o propósito de evitar o acoplamento de código numa aplicação. Em outras palavras, é a proveniência de instâncias de classes que um objeto precisa sem que este instancie por si mesmo. Podemos dizer que a injeção de dependência é uma forma de aplicar a inversão de controle.

No Spring Framework, podemos fazer a injeção de dependência da seguinte forma:

- anotação `@Autowired`;
- utilizando o construtor da classe (Constructor Injection);
- utilizando o método setter (Setter Injection).

Nosso primeiro programa

Vamos criar uma aplicação web mais simples possível e analisar o seu resultado.
Para criar um novo projeto acessamos : [Spring Initializr](#)



The screenshot shows the Spring Initializr web application interface. The header includes the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Includes radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "2.6.0 (SNAPSHOT)", "2.6.0 (RC1)", "2.5.7 (SNAPSHOT)", "2.5.6" (selected), "2.4.13 (SNAPSHOT)", and "2.4.12".
- Project Metadata:** Includes input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Packaging:** Includes radio buttons for ".jar" (selected) and "War".
- Java:** Includes radio buttons for "17", "11" (selected), and "8".
- Dependencies:** Includes a button "ADD DEPENDENCIES... CTRL + B" and the text "No dependency selected".

At the bottom of the interface, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

Nosso primeiro programa

Nessa parte, temos o diferencial: escolhemos as dependências de que o nosso projeto precisa para funcionar. Ao digitarmos *web* no campo **Dependencies**, são exibidas todas as opções relacionadas. Vamos selecionar apenas a opção **Spring Web**, depois vamos digitar *dev* e selecionar **Spring Boot DevTools**.

Dependências

ADICIONAR DEPENDÊNCIAS... CTRL + B

Spring Web

REDE

Crie aplicativos da web, incluindo RESTful, usando Spring MVC. Usa Apache Tomcat como o contêiner integrado padrão.

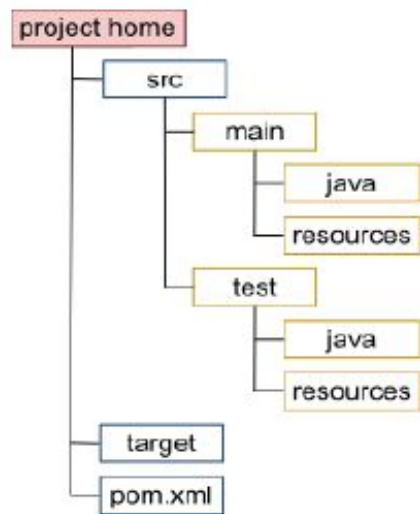
Spring Boot DevTools

FERRAMENTAS DE DESENVOLVIMENTO

Fornece reinicializações rápidas de aplicativos, LiveReload e configurações para uma experiência de desenvolvimento aprimorada.

Maven

O Maven padroniza a criação de projetos. As estruturas dos projetos Maven sempre são iguais o que permite que independente da IDE que o programador esteja utilizando, ele pode importar um projeto Maven.



src/main/java - Arquivos .java do projeto (serão entregues ao cliente)

src/main/resources - Recursos do projeto (propriedades, xml, etc).

src/test/java - Arquivos .java dos testes (não vai para o cliente)

src/test/resources - Recursos de testes do projeto (propriedades, xml, etc).

target - Binários e documentações.

pom.xml - Project Object Model (pom.xml) (Veremos na sequência)

Maven

pom.xml

Descreve todas as informações do projeto, como o group id, artifact id, version, que colocamos ao criar um projeto Além disto, o pom xml armazena informações das dependências (nossos jars), plugins, servidores e muitas coisas que podemos precisar no projeto.

Dependências

O Maven gerencia as dependências de todo o projeto no arquivo pom.xml. Esse gerenciamento beneficia o programador com a padronização das configurações das dependências, sendo assim, uma vez familiarizado com as configurações Maven, o programador conseguirá interpretar qualquer arquivo de configurações POM, de qualquer projeto.

Na estrutura de nosso projeto, o Maven criará uma pasta chamada 'Maven Dependencies' onde armazenará todas as dependências.

Dependências

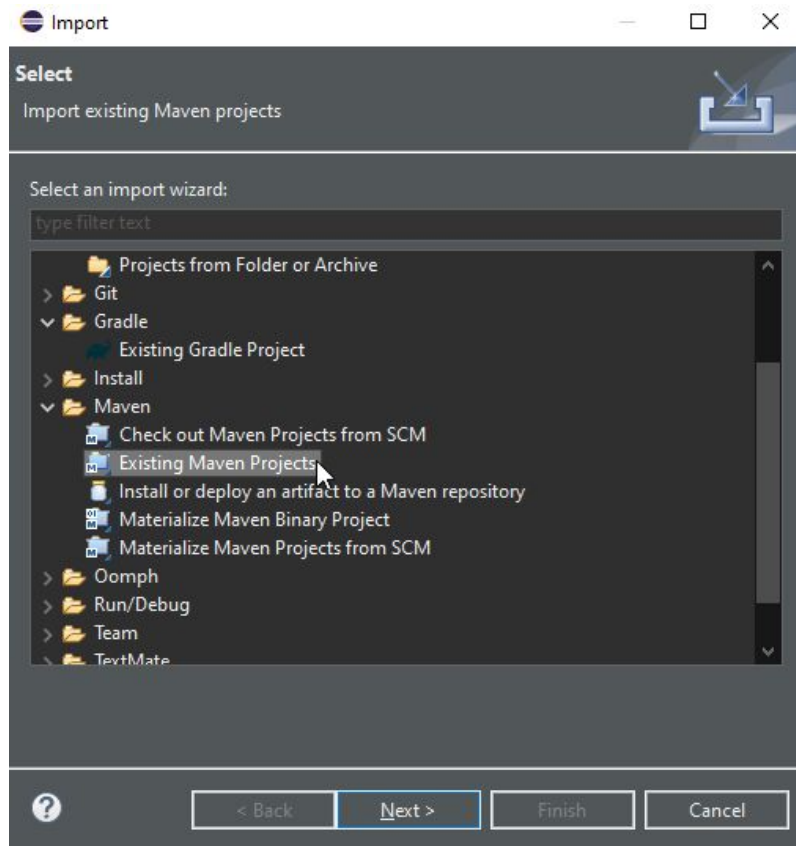
Spring Boot Devtools é um conjunto de funcionalidades que ajuda o trabalho de qualquer dev. Como, por exemplo, restart automático da aplicação quando ocorre alguma mudança no código.

O **Spring Web** é utilizado para criar aplicativos Web, incluindo RESTful, utilizando o Spring MVC. Indispensável para criação aplicações web baseadas em Spring Framework.

Importar projeto

Após gerar o projeto no Spring Initializr e descompacta-lo devemos importar nosso projeto no Eclipse:

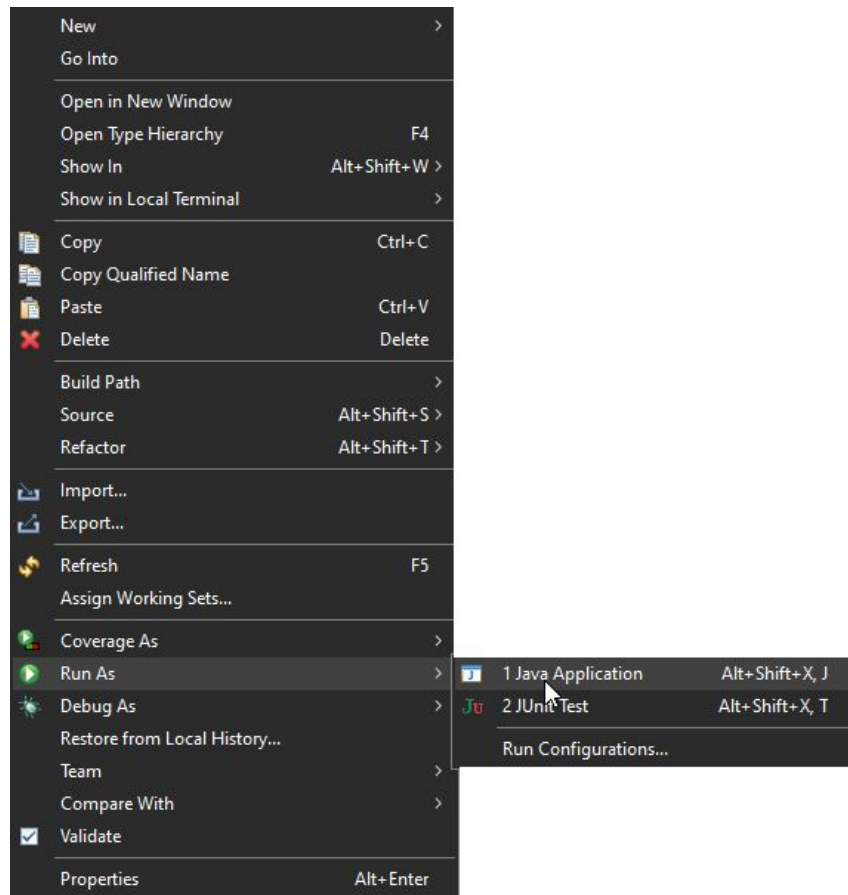
file -> import -> maven -> Existing Maven Projects



Iniciar projeto

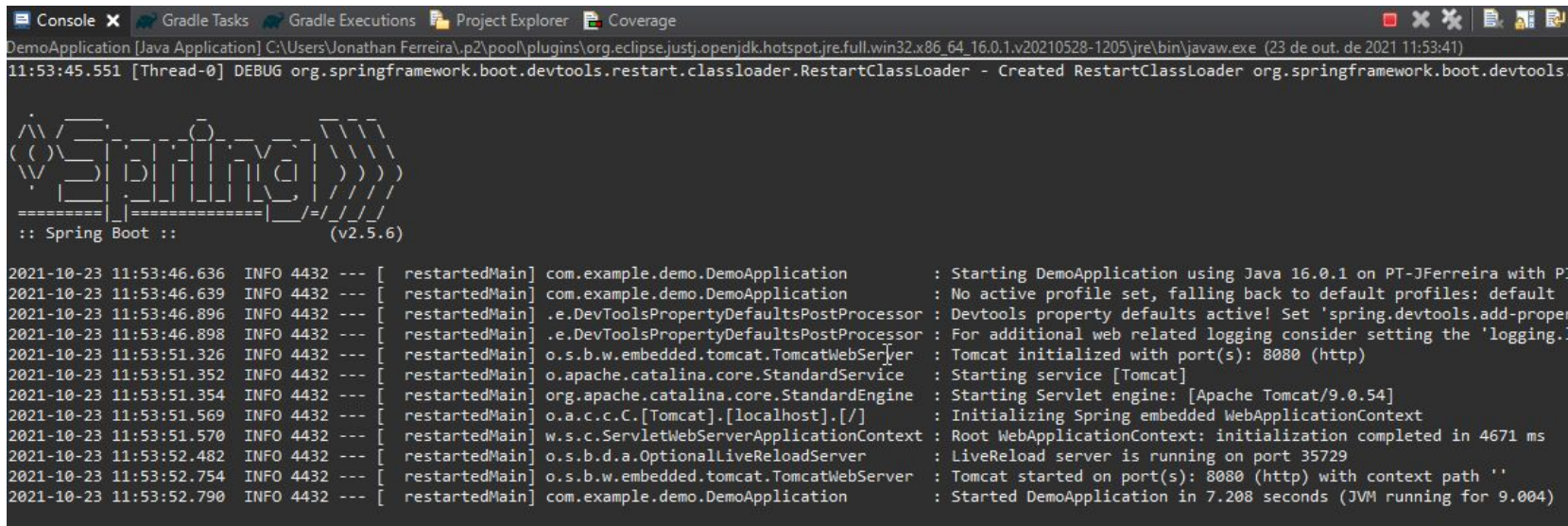
Para testar o funcionamento da nossa aplicação, clicamos com o botão direito sobre o projeto:

Run as -> Java Application(vide imagem ao lado)



Iniciar projeto

Note que alguns informações aparecem em nosso Console, entre elas a informação: Tomcat initialized with port(s): 8080 (http), que pode ser visualizado ao acessar: *localhost:8080*, em qualquer navegador.



```
Console x Gradle Tasks Gradle Executions Project Explorer Coverage
DemoApplication [Java Application] C:\Users\Jonathan Ferreira\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (23 de out. de 2021 11:53:41)
11:53:45.551 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools

:: Spring Boot ::
(v2.5.6)

2021-10-23 11:53:46.636 INFO 4432 --- [ restartedMain] com.example.demo.DemoApplication : Starting DemoApplication using Java 16.0.1 on PT-JFerreira with P
2021-10-23 11:53:46.639 INFO 4432 --- [ restartedMain] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2021-10-23 11:53:46.896 INFO 4432 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'true' to enable Spring DevTools to auto-refresh the browser on each build method change.
2021-10-23 11:53:46.898 INFO 4432 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.org.springframework.boot.devtools.restart' to 'DEBUG'
2021-10-23 11:53:51.326 INFO 4432 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-10-23 11:53:51.352 INFO 4432 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-10-23 11:53:51.354 INFO 4432 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.54]
2021-10-23 11:53:51.569 INFO 4432 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-10-23 11:53:51.570 INFO 4432 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4671 ms
2021-10-23 11:53:52.482 INFO 4432 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2021-10-23 11:53:52.754 INFO 4432 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-10-23 11:53:52.790 INFO 4432 --- [ restartedMain] com.example.demo.DemoApplication : Started DemoApplication in 7.208 seconds (JVM running for 9.004)
```


Anotações

`@Configuration` – define uma classe como fonte de definições de beans e é uma das anotações essenciais se você estiver usando a configuração baseada em Java.

`@Bean` – utilizada em métodos de uma classe, geralmente marcada com `@Configuration`, indicando ao Spring Framework que deve invocar aquele método e gerenciar o objeto retornado por ele, ou seja, agora este objeto pode ser injetado em qualquer ponto da sua aplicação (Lembra da injeção de dependência?).

`@Autowired` – já falada e exemplificada, define pontos de injeção de dependências dentro de uma classe.

Anotações

`@Scope` – define o tempo de vida de um objeto gerenciado pelo Spring Framework. Podemos combinar com outras anotações como `@Component` ou `@Bean`.

`@Primary` – é usada quando temos dois métodos anotados com `@Bean` que retornam o mesmo tipo de objeto. O Spring precisa saber qual deles será injetado por padrão quando for solicitado. Indica qual o deve ser o padrão. Obs.: Para alterar o padrão, utiliza-se a annotation `@Qualifier`..

`@Profile` – define para qual profile tal bean deve ser carregado. Comum quando existem classes que somente devem ser carregadas em ambiente de desenvolvimento ou de produção ou de teste.

Anotações

`@SpringBootApplication` – combina as `@Configuration`, `@EnableAutoConfiguration` e `@ComponentScan`. Desse modo, não precisamos instalar um servidor Web, pois o Spring Boot já vem com um servidor Tomcat incorporado.

Essa anotação ativa a configuração baseado em Java, bem como o recurso de varredura e configuração automática de componentes do Spring Boot.

`@EnableAutoConfiguration` – ativa o recurso de configuração automática.

`@EnableAsync` – quando precisa-se de ações no sistema em background (outra thread). Essa annotation deve ser colocada em classes anotadas com `@Configuration`, para que o Spring habilite o suporte de execução assíncrona.

Anotações

`@Async` – habilitado o uso de execução de métodos assíncronos com `@EnableAsync`, marca-se qualquer método de um bean gerenciado. Assim que tal método é invocado, o Spring vai garantir que a execução dele será em outra thread.

`@Component` – indica que uma classe vai ser gerenciada pelo container do Spring (Spring Container IoC).

`@ComponentScan` – utilizada em classes de configuração indicando quais pacotes ou classes devem ser escaneadas pelo Spring para que essa configuração funcione.

Anotações

`@Service` – define uma classe de serviço.

`@RestControllerAdvice` – combinação das annotations `@ControllerAdvice` e `@ResponseBody`. Indica ao Spring que se trata de um componente especializado em exceções e que o retorno dos métodos da mesma devem ser inseridos no corpo da resposta HTTP e convertidos para JSON.

`@ControllerAdvice` – lida com exceções num aplicativo Spring MVC. É usada para manipulação global de erros. Tem o controle total sobre o corpo da resposta e código de status.

VAMOS PRATICAR?