

Spring Boot

Spring vs Spring Boot

- Spring: Framework java para a plataforma Java. Criado com o objetivo de facilitar o desenvolvimento de aplicações. É adicionado a um projeto e pode ser executado dentro de um servidor de aplicação como JBoss/Wildfly ou até Weblogic.
- Spring Boot: Framework que permite ter uma aplicação rodando em produção rapidamente através de configuração facilitada ou às vezes automática. Não necessita instalação de servidor, pois contém servidor embutido (*Tomcat*).

Mais info em: <https://github.com/targettrustbr/imersao-java-19-03/blob/master/aulas/spring.MD>)

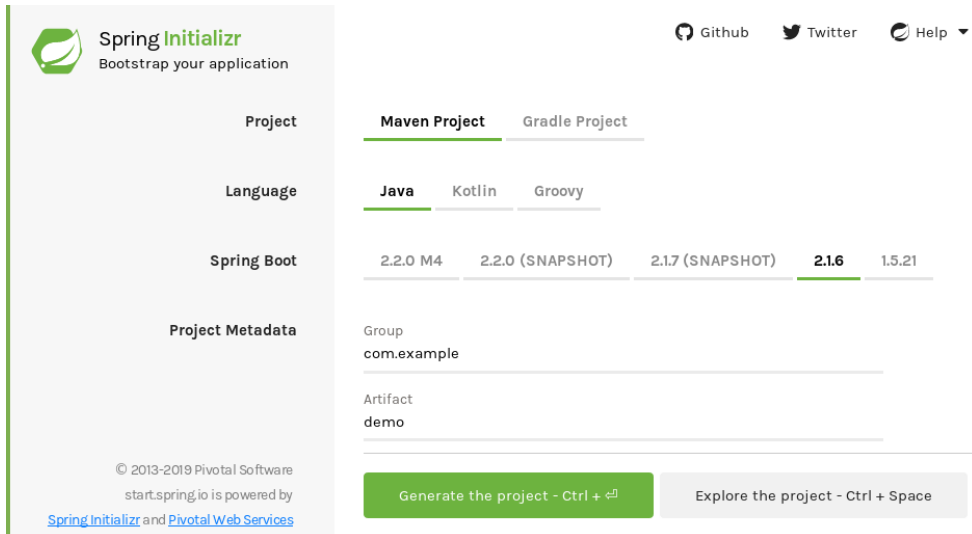
Spring vs Spring Boot

Como o Spring Boot usa Spring, todos os recursos do Spring podem ser usados no Boot, como:

- Injeção de dependências, inversão de controle
- Programação orientada a aspectos
- Model View Controller (MVC)
- Gerenciamento de transações
- Spring-data
- E muito mais :D

Initializr - <https://start.spring.io/>

É a porta de entrada para a criação de um projeto Spring Boot:



The screenshot shows the Spring Initializr web application. On the left is a sidebar with the logo and navigation links. The main area has tabs for 'Maven Project' and 'Gradle Project', with 'Maven Project' selected. Below these are tabs for 'Java', 'Kotlin', and 'Groovy', with 'Java' selected. A table of Spring Boot versions is shown, with '2.1.6' highlighted. Below the table are input fields for 'Group' (com.example) and 'Artifact' (demo). At the bottom are two buttons: 'Generate the project - Ctrl + ⌘' and 'Explore the project - Ctrl + Space'.

Spring Initializr
Bootstrap your application

Project

Language

Spring Boot

Project Metadata

© 2013-2019 Pivotal Software
start.spring.io is powered by
[Spring Initializr](#) and [Pivotal Web Services](#)

Github Twitter Help

Maven Project Gradle Project

Java Kotlin Groovy

2.2.0 M4	2.2.0 (SNAPSHOT)	2.1.7 (SNAPSHOT)	2.1.6	1.5.21
----------	------------------	------------------	-------	--------

Group
com.example

Artifact
demo

Generate the project - Ctrl + ⌘

Explore the project - Ctrl + Space

Exemplo em:

<https://github.com/targettrustbr/imersao-java-19-03/blob/master/aulas/spring-data-jpa.MD>

Initializr

Vamos fazer juntos: explorar um projeto no Initializr e verificar sua estrutura. Vamos analisar:

- Application
- pom.xml:
 - Parent Spring
 - Starters
 - Versão do Spring

Starters

São como plugins, que requerem mínima configuração e adicionam funcionalidades em nossa aplicação. Os mais comuns:

- data-jpa
- test
- web
- security
- actuator

Classe Application

Contém a anotação `@SpringBootApplication` responsável por fazer nosso projeto inicializar de fato. É como nossa classe MAIN.

Ele substitui as seguintes anotações:

- `@ComponentScan`: escaneamento de componentes
- `@EnableAutoConfiguration`: autoconfiguração geral do Spring
- `@Configuration`: Permite registrar beans nesta classe.

Vamos fazer juntos: Criar um projeto com os atributos:

- Group: br.com.tt
- Artifact: petshop
- Dependencies: web, thymeleaf

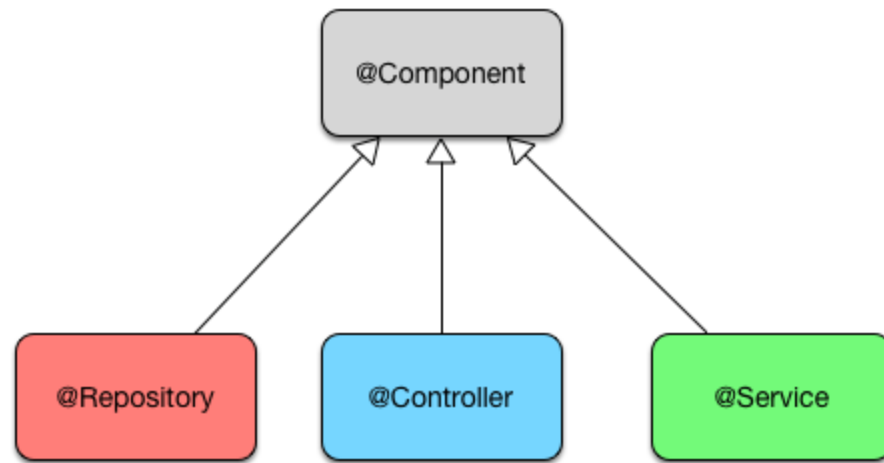
Executar o projeto. Abrir no navegador (localhost:8080).

Adicionar Actuator.

Spring Beans (alguém conhece os Java Beans?)

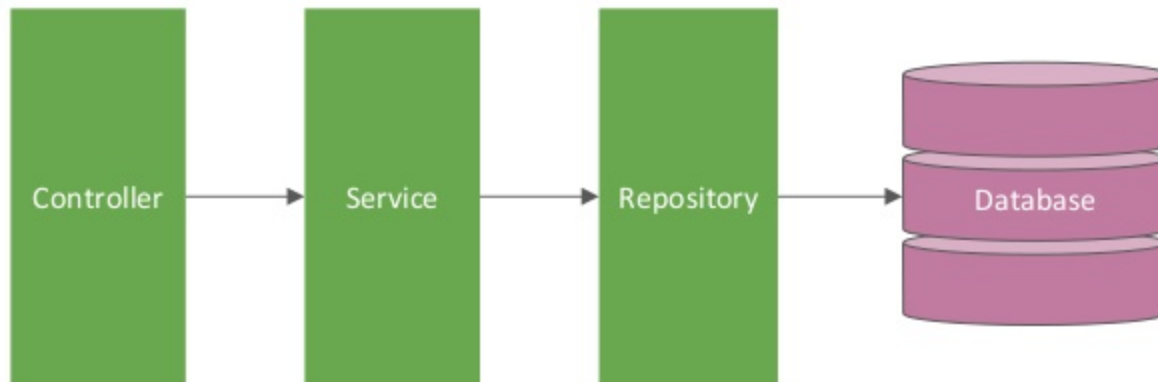
- São classes onde todo o ciclo de vida é gerenciado pelo Spring. Ou seja, não nos preocupamos em declarar, instanciar, destruir ou mesmo armazenar.
- Há duas formas de criar um Spring Bean:
 - Através de Stereotypes.
 - Através de @Bean.

Stereotypes Spring:



Camadas do MVC:

Classic way



Stereotypes Spring:

- @Controller: camada de apresentação.
 - @RestController: tipo especializado de Controller para utilização com REST.
- @Service: camada de aplicação e negócio
- @Repository: camada de persistência (DAO, Repositório)
- @Component - Utilizada em outros tipos de classes, que não se encaixam nas camadas acima, como classes utilitárias.

Spring MVC

- Aplicação Web com View integrada ao Spring.
- Usaremos um framework: Thymeleaf
 - Responsável por interpretar nossos templates
 - Utiliza variáveis do nosso Model
 - Utiliza `@Controller's` como definição de rotas

Spring MVC - Thymeleaf

Conteúdo dinâmico no template:

```
${...} : Variable expressions.  
*{...} : Selection expressions.  
#{...} : Message (i18n) expressions.  
@{...} : Link (URL) expressions.  
~{...} : Fragment expressions.
```

Inserindo no HTML:

```
<span th:text="${book.author.name}">  
<li th:each="book : ${books}">
```

Mais em: <https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>

Spring Beans

Vamos fazer juntos:

- Criar o ClienteController usando Stereotypes.
- Criar uma tela index com a lista de clientes.

Injeção de Dependência (DI) e Inversão de Controle (IC)

- A inversão de dependência é uma forma de tirar do nosso controle a responsabilidade de instanciar certas classes, passando para algum framework.
- Essa inversão pode ser feita utilizando a Injeção de Dependência. Isso é comum em diversos frameworks, incluindo Spring e também no Java EE.

Spring Beans

Vamos fazer juntos:

- Criar um service chamado ClienteService.
- Mover a criação do cliente para o service.
- "Injetar o service no Controller".