Spring

Spring é um ecossistema de desenvolvimento para facilitar a criação de aplicações Java usando módulos de desenvolvimento independentes. Você pode criar:

- **Micro serviços:** para resumir, micro serviços são uma nova abordagem em que o código de uma aplicação é entregue em pequenas partes gerenciáveis e independentes.
  Reativos: é um novo paradigma que permite a criação de aplicações assíncronas e sem bloqueio que ligam o controle de fluxo.

Nota: observe que uma aplicação Spring Reativo para web é diferente do Spring MVC padrão em alguns aspectos abordados mais a diante.

- **Orientados a Eventos:** é o micro serviço orientado a um streaming de eventos, que representam um fluxo constante de eventos. Exemplo: uma ação da bolsa de valores muda o seu valor, essa mudança gera um evento que é lançado.
- **Binders:** eles preenchem uma lacuna entre o sistema de mensagens e o código. As mensagens são dados recebidos (normalmente gerados pelos eventos).
- **Cloud:** serviços necessários para que sua aplicação seja executada em um sistema de computação em nuvem.
- **Aplicativos Web:** modelo moderno, HTML ao lado servidor, API's REST e sistemas bidirecionais baseados em eventos.
- **Serveless:** é um recurso de abstração da computação em nuvem que permite você focar somente na construção da aplicação e a camada de negócio dela, sem se preocupar com a infraestrutura do projeto.
- **Batch (lote):** são processamentos de dados finitos que não requer interação ou interrupção externa. Fornecidos a nível de produção.

Estes e diversos outros serviços, você pode criar utilizando os módulos do Spring. Abordaremos todos os módulos nesta documentação.

Alguns dos códigos que não serão mostrados em imagens, estarão disponíveis no GitHub para estudos.

Spring Boot

Modulo que facilita a criação de aplicativos independentes baseados em Spring. Vamos esquecer o Spring Initilizr por hora.

O Spring Boot, permite a criação de aplicativos autônomos, incorporados a servidores como TomCat, sem sequer configura-los. Fornece dependências iniciais opcionais, configuração automática de bibliotecas, recursos prontos para produção, e não há requisitos de arquivos XML para configuração.

O Spring utiliza dos chamados Starters para fazer a maior parte de toda sua automação. Como vimos ele mesmo cuida da configuração do servidor, dependências e bibliotecas que queremos utilizar em nossa aplicação.

Requisitos de Sistema: para utilizar dos Spring Boot, deveremos ter em nossa máquina:

1. Java 8 ou superior
2. Maven ou Gradle

Containers Servlet: são servidores que você pode utilizar com Spring Boot para trabalhar com os containers Servlet.

- TomCat
- Jetty
- Undertown

Instalação do Spring Boot

Antes de mais nada temos que verificar a existência do Java no seu computador. No terminal digite o comando:

java –version

As informações sobre o Java instalado apareceram. Podemos continuar.

A instalação do Spring Boot pode ser através do site principal pegando as jars bibliotecas e adicionando no seu classPath ou utilizando a CLI do Spring Boot para isso.

Entretanto, vamos também instalar o Maven junto a CLI do Spring Boot. Usaremos o **SDKMAN** que é um gerenciador de versões para Java, Maven, Gradle, Spring CLI, entre outros.

Siga as instruções do site https://sdkman.io/ para fazer a instalação do SDKMAN, e assim podermos continuar.

Instalando

Com SDKMAN instalado na sua máquina, utilize o comando no terminal para instalarmos 1º o Maven – Maven é um gerenciador de projetos para Java, é o que vamos usar em toda documentação.

sdk install maven

Ele começará a instalar a última versão disponível do Maven. Ótimo, com Maven instalado, vamos instalar a CLI do Spring Boot que nos auxiliará em muitas coisas.

No terminal, digite o comando:

sdk install springboot

E pronto, ele instalará tudo que precisaremos para rodar nossos projetos Spring, independente do modulo, e também ajudará a controlar versões futuras ou anteriores, caso precisemos.

Desenvolvendo o 1º Aplicativo com Spring Boot

Vamos desenvolver um simples "Hello Word", aplicativo da web que rapidamente rodaremos e veremos o resultado.

Antes de tudo, verifique as instalações que fizemos anteriormente utilizando do terminal.

java –version

mvn –v

spring –v

Digite cada um desses comandos um de cada vez e verificará a instalação deles feita anteriormente.

Tudo certo? Continuemos. Crie uma pasta separada com um nome bem simples – *HelloWordSpringBoot* – e dentro dela é que ficara nossos arquivos com o código para rodar.

Dentro da pasta, crie um arquivo com nome *pom* e extensão *xml,* ficando *pom.xml*.

Esse pom.xml é nosso arquivo de maior importância, porque o Maven utiliza ele para toda a configuração do Projeto.

Abra o POM com alguma IDE instalada na sua máquina, pode ser um VSCODE ou até mesmo um InteliJ.

Dentro do POM coloque o código que está localizado neste site – *https://github.com/paddoncreative/spring_doc/blob/main/HelloWordSpringBoot/pom.xml* – que conterá o código do POM que precisamos.

Você pode usar o Maven para conferir, se está tudo certo, usando um comando no

terminal dentro de nossa pasta, no diretório raiz.

mvn package

Isso criara uma target pasta que conterá um jar executável do nosso projeto, mas ainda não terá nada para executar. Vamos lá criar então.

Adicione dependências ao classPath, que é basicamente "uma pasta que conterá todas as bibliotecas usadas no projeto". Para adicionar dependências (lib. Ou bibliotecas que baixaram com a ajuda do maven) você precisa ir ao pom que criamos e adicionar abaixo:

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

Agora que adicionamos esse starter, como pode ver no nome, podemos trabalhar com a integração de servidor, e web.

Um starter é como o nome já diz, a ideia vem que ele traga para nos bibliotecas ou servidores para uma determinada área, note a web, palavra, logo após a palavra starter. Significa, traga tudo que preciso (dependências, bibliotecas, servidor) para iniciar um projeto web.

Agora você pode utilizar do comando:

mvn dependency:tree

Para ver as dependências que esse starter nos trouxe e as quais trabalharemos para criar um projeto web.

Continuando, crie agora, um único arquivo Java, dentro do seguinte sistema de pasta – *src/main/java* – aqui você adicionara um *Hello.java*.

Entenda que por padrão o maven compila fontes de src > main > java por padrão.

Dentro do arquivo cole o código que está também no meu GitHub, neste link: *https://github.com/paddoncreative/spring_doc/blob/main/HelloWordSpringBoot/src/main/java/Hello.java*

Agora podem rodar o comando no terminal, lembrando que deve estar na pasta raiz (onde está o pom), usem:

mvn spring-boot:run

Prontinho, abram o navegador e naveguem para *localhost:8080*. La você verá o retorno de "Hello Word".

Explicando o Hello.java

Para entendermos, saibam que o Spring, assim como Java, usa anotações para dizer a uma classe o que ela significa. Então veja.

@RestController diz ao Spring que aquela classe é uma classe do tipo controller, ou seja, recebe e envia solicitações para o cliente, o comum da web.

@RequestMapping diz ao Spring que aquele método vai tratar da requisição na rota "/" ou raiz. Então ele executa o método quando navegamos para a rota raiz.

@EnableAutoConfiguration diz ao Spring que ele pode configurar o projeto como bem entender, claro, ele não fará nada de ruim. Mas ele entende que a configuração deve ser automatizada e então ele cuida disso para você.

Finalizando assim, temos o método *main* que como todos nós sabemos, é o método principal e em toda as aplicações é por ele que se dá início a execução do projeto.

Spring não é diferente, o método *main* ali descrito, está fazendo exatamente isso. Pedindo ao *SpringApplication* que corra ou execute nossa classe Hello.java.

Criando Jar Executável

Mesmo sendo uma aplicação simples, vamos criar um jar totalmente independente e executável para rodar na máquina de qualquer um ou em qualquer lugar que tenha Java instalado como requisito.

Para isso, adicione ao POM um simples código a mais:

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

Apenas com esse código e com o comando, no terminal:

mvn package

Você cria um jar que fica na pasta *target* e esse jar pode ser executado em qualquer lugar que tenha o nosso Java instalado.

Para rodar o jar, simplesmente, no terminal, use o comando:

java –jar target/myproject-0.0.1-SNAPSHOT.jar

E pronto, ele executa e você pode novamente ir ao navegador e olhar que "Hello Word", estará novamente no *localhost:8080.*