

03

Subindo nossa aplicação web

Transcrição

Com o *Spring Boot* já instalado em nossas dependências, precisamos configurar a aplicação. Sendo esta uma aplicação web, precisaremos de um container que a execute, sendo este o *Tomcat*, *Jetty*, *Wildfly* ou qualquer um.

A vantagem de usar *Spring Boot* é que não precisamos mais nos preocupar com a instalação e configuração do projeto em um destes containers. Precisamos apenas configurar o *Spring Boot* para que inicie um container automaticamente e gerencie todos os nossos *Beans*. Faremos isso por meio das classes Java.

Até um tempo atrás, as configurações não eram feitas usando as classes Java, mas sim por arquivos XML, o que não é mais necessário hoje. O único XML da nossa aplicação será o de dependências do Maven.

Criaremos então a classe `Configuracao` que irá configurar o *Spring* em nosso projeto. Esta será criada dentro do pacote `br.com.alura.listavip`.

```
package br.com.alura.listavip;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Configuracao {

}
```

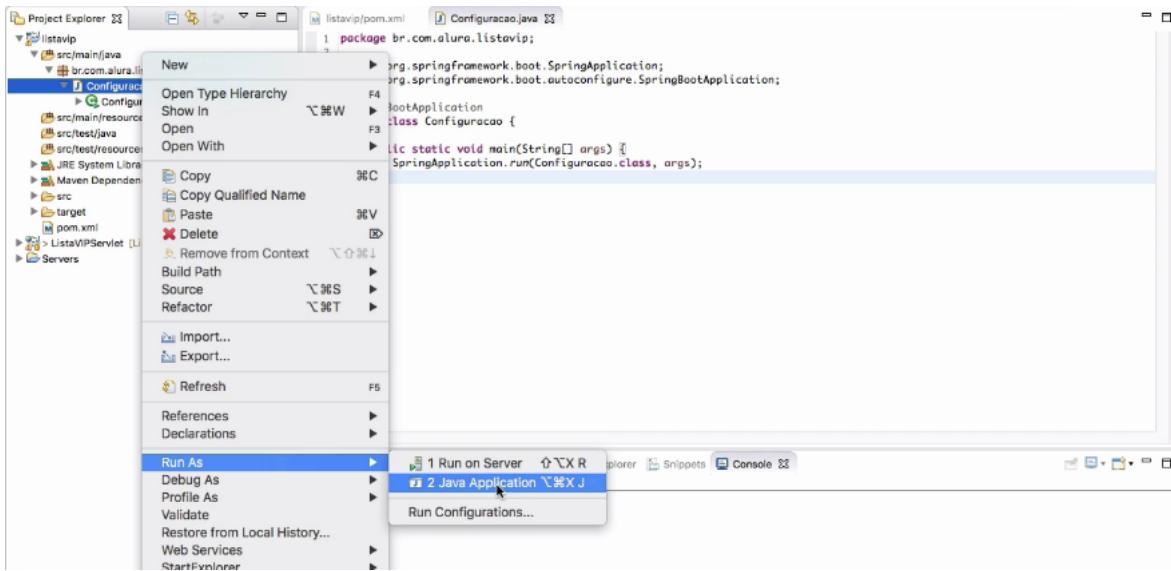
Note a presença da anotação `@SpringBootApplication`, ela é a responsável por configurar nossa aplicação *Spring*. Ela é o resultado de um aglomerado de outras configurações, como por exemplo, o diretório onde o *Spring* deve procurar todos os *Beans* da aplicação, entre outras configurações.

Agora temos nossa aplicação inicial pronta e já podemos executá-la. Mas como podemos executar uma aplicação Java com uma classe praticamente em branco? Não podemos! Toda e qualquer aplicação básica com Java precisa do método `main`. Criaremos um então.

Dentro do método `main`, usaremos uma classe do *Spring Boot* para que este execute todas as configurações da nossa aplicação e a deixe disponível para podermos acessá-la, que receberá o nome de `SpringApplication`. Ela terá o método `run`, responsável por executar a classe de configuração da aplicação. Vejamos:

```
public static void main(String[] args){
    SpringApplication.run(Configuracao.class, args);
}
```

Como estamos na própria classe de configuração, passamos apenas o nome da classe. E como não temos nenhum parâmetro adicional para estas configurações, usamos apenas o `args` recebido pelo método `main`. Assim, já podemos executar nossa aplicação. Execute-a como uma aplicação Java normal.



Logo veremos algumas informações sendo impressas no console.

```
Configuracao [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_73.jdk/Contents/Home/bin/java (21 de jul de 2016 15:51:50)
:: Spring Boot ::   (v1.3.6.RELEASE)

2016-07-21 15:51:53.552 INFO 2373 --- [main] br.com.alura.listavip.Configuracao      : Starting Configuracao on MacBook-de-Renaldo.local with PID 2373 (/Volumes
2016-07-21 15:51:53.620 INFO 2373 --- [main] br.com.alura.listavip.Configuracao      : No active profile set, falling back to default profiles: default
2016-07-21 15:51:53.899 INFO 2373 --- [main] o.s.boot.SpringApplication               : Refreshing org.springframework.boot.context.embedded.AnnotationConfig
2016-07-21 15:51:58.084 INFO 2373 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2016-07-21 15:51:58.114 INFO 2373 --- [main] o.apache.catalina.core.StandardService : Starting service Tomcat
2016-07-21 15:51:58.117 INFO 2373 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.0.36
2016-07-21 15:51:58.742 INFO 2373 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]    : Initializing Spring embedded WebApplicationContext
2016-07-21 15:51:59.042 INFO 2373 --- [ost-startStop-1] o.s.web.context.ContextLoader       : Root WebApplicationContext: initialization completed in 4947 ms
2016-07-21 15:51:59.042 INFO 2373 --- [ost-startStop-1] o.s.b.c.e.ServletRegistrationBean     : Mapping servlet: 'dispatcherServlet' to [/]
2016-07-21 15:51:59.064 INFO 2373 --- [ost-startStop-1] o.s.b.c.e.FilterRegistrationBean       : Mapping filter: 'characterEncodingFilter' to: [//*]
2016-07-21 15:51:59.069 INFO 2373 --- [ost-startStop-1] o.s.b.c.e.FilterRegistrationBean       : Mapping filter: 'hiddenHttpMethodFilter' to: [//*]
2016-07-21 15:51:59.074 INFO 2373 --- [ost-startStop-1] o.s.b.c.e.FilterRegistrationBean       : Mapping filter: 'httpPutFormContentFilter' to: [//*]
2016-07-21 15:51:59.074 INFO 2373 --- [ost-startStop-1] o.s.b.c.e.FilterRegistrationBean       : Mapping filter: 'requestContextFilter' to: [//*]
2016-07-21 15:52:01.163 INFO 2373 --- [main] s.w.s.m.m.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedde
2016-07-21 15:52:01.554 INFO 2373 --- [main] s.w.s.m.m.RequestMappingHandlerMapping : Mapped "[error]" onto public org.springframework.http.ResponseEntity<
2016-07-21 15:52:01.564 INFO 2373 --- [main] s.w.s.m.m.RequestMappingHandlerMapping : Mapped "[error].produces=[text/html]" onto public org.springframework
2016-07-21 15:52:01.781 INFO 2373 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springfr
2016-07-21 15:52:01.792 INFO 2373 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.spring
2016-07-21 15:52:02.243 INFO 2373 --- [main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2016-07-21 15:52:02.589 INFO 2373 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2016-07-21 15:52:02.620 INFO 2373 --- [main] br.com.alura.listavip.Configuracao      : Started Configuracao in 10.828 seconds (JVM running for 12.332)
```

E então, podemos acessar nossa aplicação no navegador.

Mas convenhamos que mostrar uma página de erro logo no início da aplicação não é uma boa ideia para mostrar para o chefe ou amigo.

Para exibirmos uma mensagem mais interessante, transformaremos nossa classe de configuração em um `Controller`, e mapearemos o endereço `/` para um método que retornará uma `String` com uma mensagem de olá.

```
@SpringBootApplication
@Controller
public class Configuracao {

    @RequestMapping("/")
    @ResponseBody
    public String ola(){
        return "Olá, Bem vindo ao sistema Lista VIPs";
    }
}
```

```
public static void main(String[] args){  
    SpringApplication.run(Configuracao.class, args);  
}  
  
}
```

As anotações `@Controller` , `@ResponseBody` , `@RequestMapping` , são específicas do *Spring MVC*. Caso não tenha conhecimentos sobre o framework, recomendamos que faça estes cursos:

- [Spring MVC I](https://cursos.alura.com.br/course/spring-mvc-1-criando-aplicacoes-web) (<https://cursos.alura.com.br/course/spring-mvc-1-criando-aplicacoes-web>).
- [Spring MVC II](https://cursos.alura.com.br/course/springmvc-2-integracao-cache-seguranca-e-templates) (<https://cursos.alura.com.br/course/springmvc-2-integracao-cache-seguranca-e-templates>).

Ao iniciarmos a aplicação novamente, teremos.



Lembre-se de sempre separar seus `controller`s em outras classes. Neste exemplo, apenas usamos a classe de configuração para demonstração da praticidade de trabalhar com o *Spring Boot*.