Build do projeto

Course	Spring Boot 3
Confidence	Not Confident
Last Edited	@14 de agosto de 2023 10:55
Curso	Spring Boot 3: Documente teste e prepare uma API para o deploy

Tópicos

Build com Maven

Exemplo de comando que deve ser executado para executar o jar do projeto, passando variaveis de ambiente logo em seguida do comando `-D`

```
java "-Dspring.profiles.active=prod" -DDATASOURCE_URL=jdbc:mysql://localhost/vollmed_api -DDATASOURCE_USERNATE=root -DDATASOURCE_PASSO
```

Para saber mais: build com arquivo .war

Projetos que utilizam o Spring Boot geralmente utilizam o formato **jar** para o empacotamento da aplicação, conforme foi demonstrado ao longo desta aula. Entretanto, o Spring Boot fornece suporte para o empacotamento da aplicação via formato **war**, que era bastante utilizado em aplicações Java antigamente.

Caso você queira que o build do projeto empacote a aplicação em um arquivo no formato war, vai precisar realizar as seguintes alterações:

1) Adicionar a tag `<packaging>war</packaging>` no arquivo `pom.xml` do projeto, devendo essa tag ser filha da tag raiz `<project>`:

```
<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>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.0</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>med.voll</groupId>
  <artifactId>api</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <name>api</name>

  <packaging>war</packaging>
```

2) Ainda no arquivo `pom.xml`, adicionar a seguinte dependência:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-tomcat</artifactId>
  <scope>provided</scope>
</dependency>
```

3) Alterar a classe *main* do projeto (`ApiApplication`) para herdar da classe `SpringBootServletInitializer`, bem como sobrescrever o método `configure`:

```
@SpringBootApplication
public class ApiApplication extends SpringBootServletInitializer {

  @Override
  protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
    return application.sources(ApiApplication.class);
  }

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

}
```

Pronto! Agora, ao realizar o build do projeto, será gerado um arquivo com a extensão `.war` dentro do diretório `target`, ao invés do arquivo com a extensão `.jar`.

Varáveis de ambiente

Aprendemos ao longo desta aula como utilizar variáveis de ambiente nos arquivos properties da aplicação.

Escolha as alternativas que representam vantagens de se utilizar variáveis de ambiente nas configurações de uma aplicação.

- Flexibilizar as configurações da aplicação.
  - Com variáveis de ambiente, é possível modificar configurações na aplicação sem ter que realizar alterações no código dela.
- Evitar vulnerabilidades na aplicação.
  - Com variáveis de ambiente, é possível modificar configurações na aplicação sem ter que realizar alterações no código dela.
- Evitar a execução dos testes automatizados no processo de build da aplicação.
  - Os testes automatizados não serão ignorados por conta da utilização de variáveis de ambiente.
- Acelerar o tempo de build da aplicação.
  - Não necessariamente o tempo de build será menor ao se utilizar variáveis de ambiente.

O que aprendemos?

Nesta aula, você aprendeu como:

- Funciona o build de uma aplicação com Spring Boot;
- Utilizar arquivos de propriedades específicos para cada profile, alterando em cada arquivo as propriedades que precisam ser modificadas;

- Configurar informações *sensíveis* da aplicação, como dados de acesso ao banco de dados, via **variáveis de ambiente**;
- Realizar o build do projeto via Maven;
- Executar a aplicação via terminal, com o comando `java -jar`, passando as variáveis de ambiente como parâmetro.
