



DIGITAL  
INNOVATION  
ONE

Spring Boot

# Profiles e Configurações

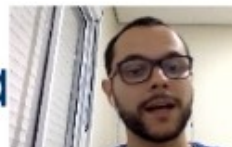
Rodrigo Peleias

Desenvolvedor Java Senior na ClickBus



# Objetivos da Aula

- 1.** Importância do uso de Profiles.
- 2.** Configurações com arquivos properties.
- 3.** Configurações com arquivos YAML.
- 4.** Configurações com command line.
- 5.** Configurações com variáveis de a



# Requisitos Básicos

- ✓ Java 8 ou versões superiores.
- ✓ Maven 3.5.2 ou versões superiores.
- ✓ IntelliJ IDEA Community Edition ou sua IDE favorita.
- ✓ Muita vontade de aprender e compartilhar conhecimento :)





DIGITAL  
INNOVATION  
ONE

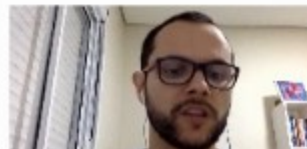
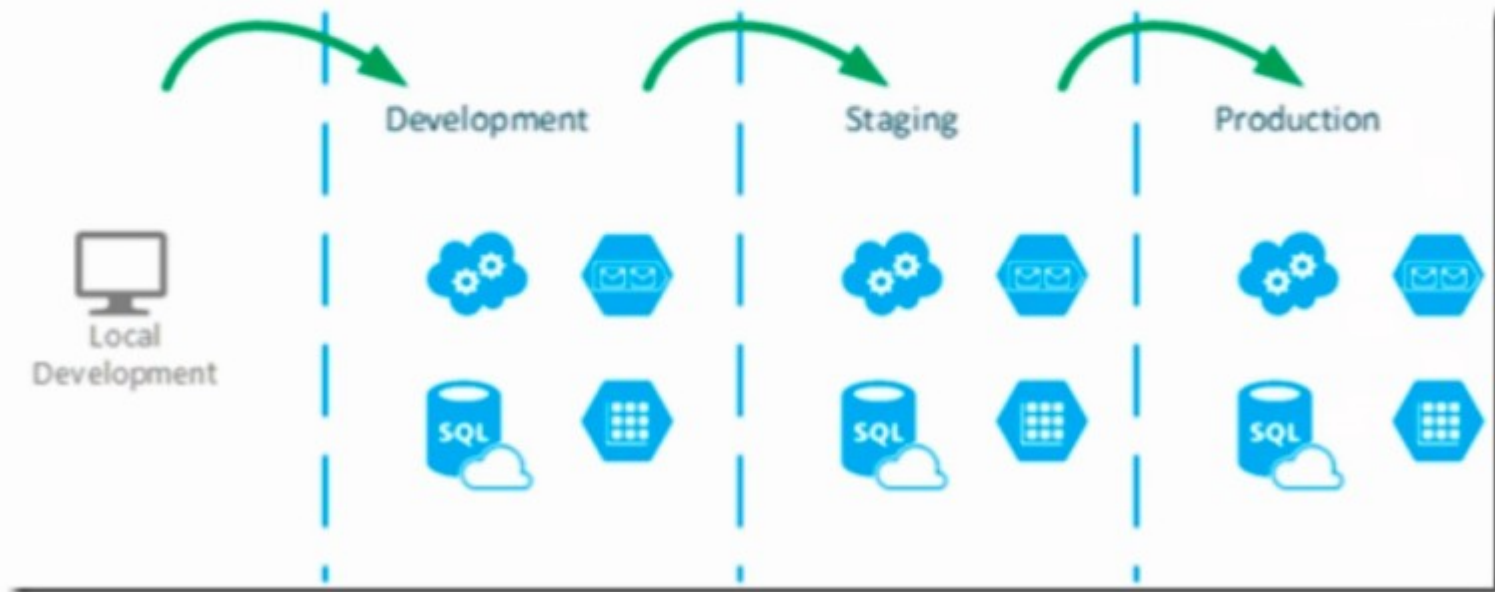
# Parte 1: Importância dos Profiles

## Spring Boot



# Múltiplos ambientes

- Ambientes para desenvolvimento, teste e produção.





# Múltiplos ambientes

- Bancos de dados para cada ambiente.
- Execução de testes unitários em ambiente local.
- Suíte de testes completas em ambiente de teste.
- Simulação do ambiente real em staging.
- Deploy simplificado em produção.

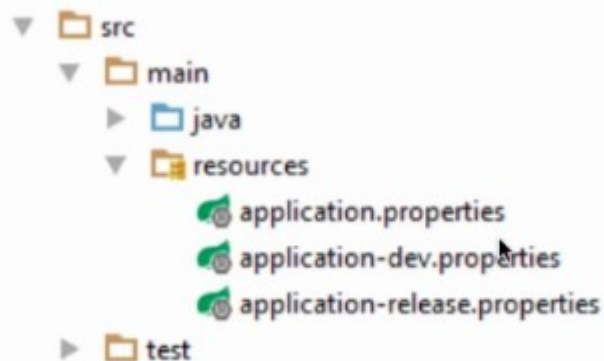






# Spring Boot Profiles

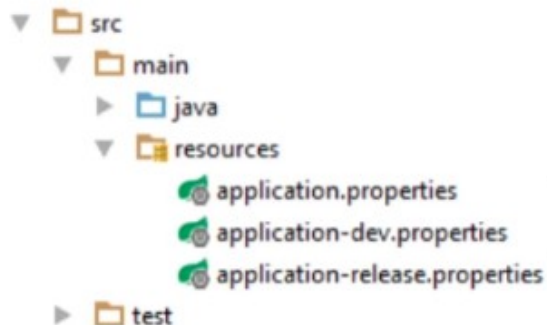
- Configurações próprias para cada ambiente.
- Ambientes com sua configuração: **dev, production.**





# Spring Boot Profiles

- Configurações próprias para cada ambiente.
- Ambientes com sua configuração: **dev, production.**



## #DEV ENVIRONEMNT SETTING#

`app.message= This is the property file for the ${spring.`

```
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:mem:db;DB_CLOSE_DELAY=-1
spring.datasource.username=sa
spring.datasource.password=sa
```

## #PROD ENVIRONEMNT SETTING#

`app.message= This is the property file for the ${spring.appli`

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc
spring.datasource.url=jdbc:mysql-instance100.sovannm
spring.datasource.username=<USERNAME_PROD>
spring.datasource.password=<SECRET_PASS>
```







DIGITAL  
INNOVATION  
ONE

# Parte 2: Configurações com Properties

Spring Boot



# Spring Boot Properties

```
@Configuration
@ConfigurationProperties("spring.datasource")
@SuppressWarnings("unused")
public class DBConfiguration {

    private String driverClassName;
    private String url;
    private String username;
    private String password;
```

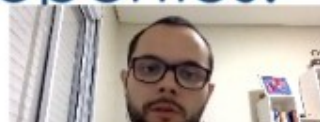
```
@Profile("dev")
@Bean
public String devDatabaseConnection() {
    System.out.println("DB connection for DEV - H2");
    System.out.println(driverClassName);
    System.out.println(url);
    return "DB connection for DEV - H2";
}
```

```
@Profile("prod")
@Bean
public String prodDatabaseConnection() {
    System.out.println("DB Connection to RDS_PROD - High Performance");
    System.out.println(driverClassName);
    System.out.println(url);
    return "DB Connection to RDS_PROD - High Performance Instance";
}
```



# Exercício juntos

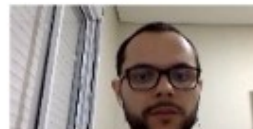
- Projeto com spring.initializr e importar na IDE.
- Arquivos application.properties para dev e prod.
- Classe de configuração de BD e anotar com @Configuration.
- Mapear propriedades com @ConfigurationProperties.





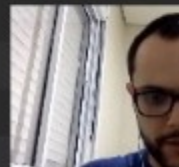
# Exercício juntos

- Criar métodos para instanciar Beans de cada env.
- Criar classe anotada com `@RestController`.
- Injetar propriedade `appMessage` com `@Value`.
- Criar método que retorna a mensagem acima.
- Executar projeto no browser.



```
public class DBConfiguration {  
  
    private String driverClassName;  
    private String url;  
    private String username;  
    private String password;  
  
    @Profile("dev")  
    @Bean  
    public String testDatabaseConnection() {  
        System.out.println("DB connection for DEV - H2");  
        System.out.println(driverClassName);  
        System.out.println(url);  
        return "DB Connection to H2_TEST - Test instance";  
    }  
  
    public String productionDatabaseConnection() {  
        System.out.println("DB connection for Production - MySQL");  
        System.out.println(driverClassName);  
        System.out.println(url);  
        return "DB Connection to MYSQL_TEST - Test instance";  
    }  
}
```

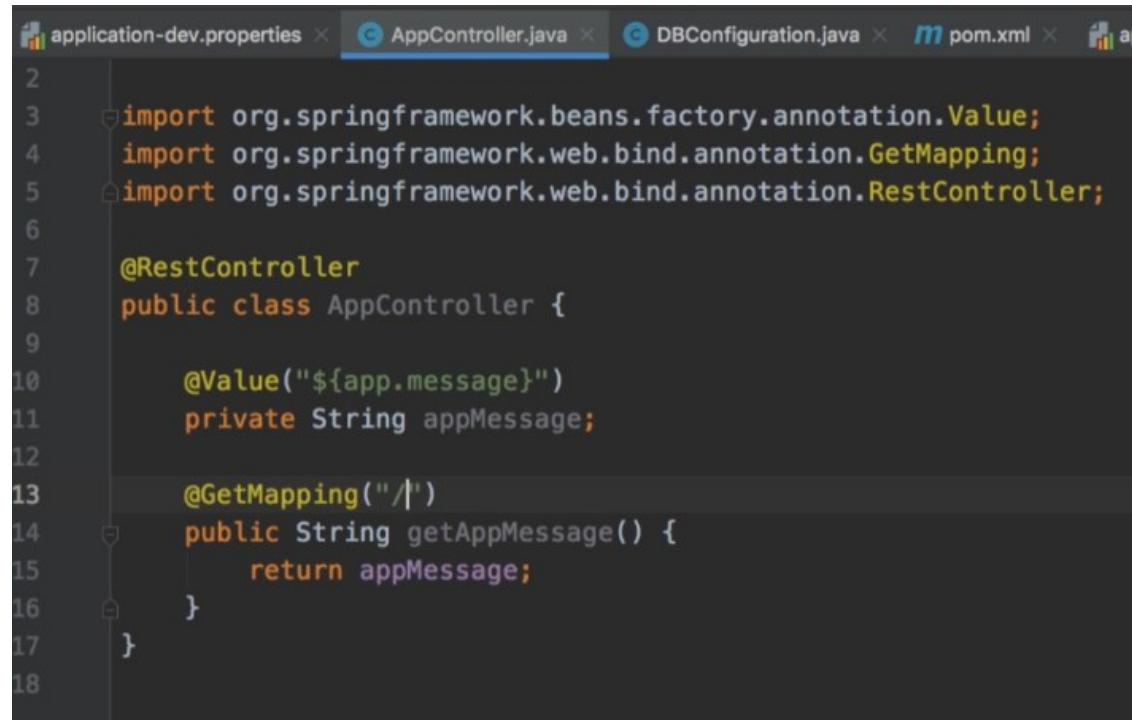
DBConfiguration > productionDatabaseConnection()





```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.10</version>
  <scope>provided</scope>
</dependency>
```

Lombok – com anotações  
Invocamos os Gatter and Setters



```
application-dev.properties × AppController.java × DBConfiguration.java × pom.xml ×
2
3 import org.springframework.beans.factory.annotation.Value;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 public class AppController {
9
10     @Value("${app.message}")
11     private String appMessage;
12
13     @GetMapping("/")
14     public String getAppMessage() {
15         return appMessage;
16     }
17 }
18
```



DIGITAL  
INNOVATION  
ONE

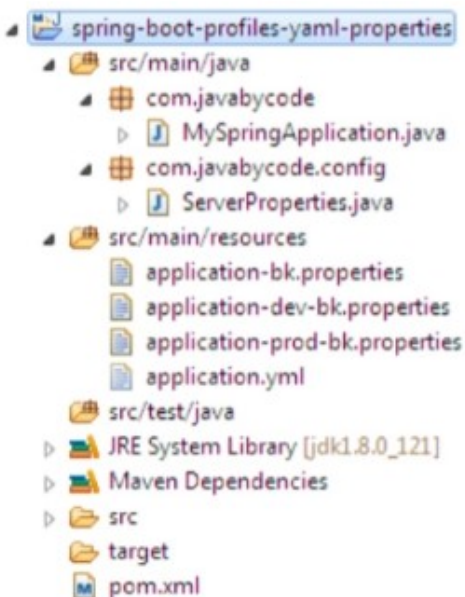
# Parte 3: Configurações com YAML

## Spring Boot



# Configurações com YAML

- Troca no formato de configurações: formato .YML.



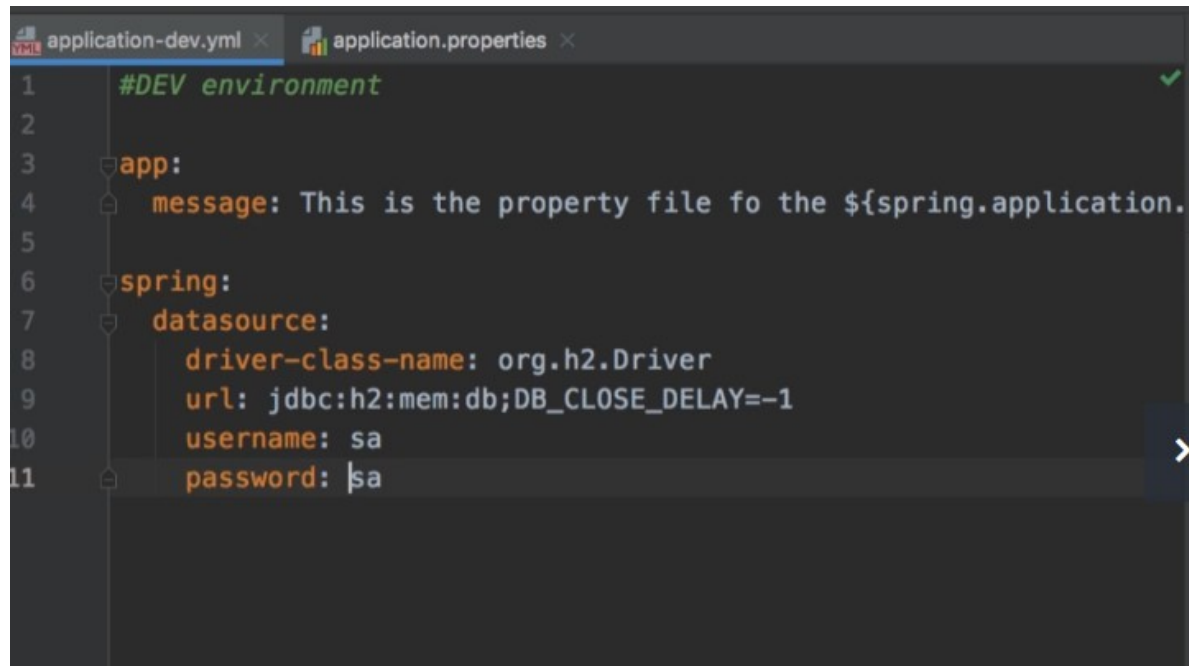
application.properties

```
basic.value: true  
basic.message: Dynamic Message  
basic.number: 100
```

application.yml

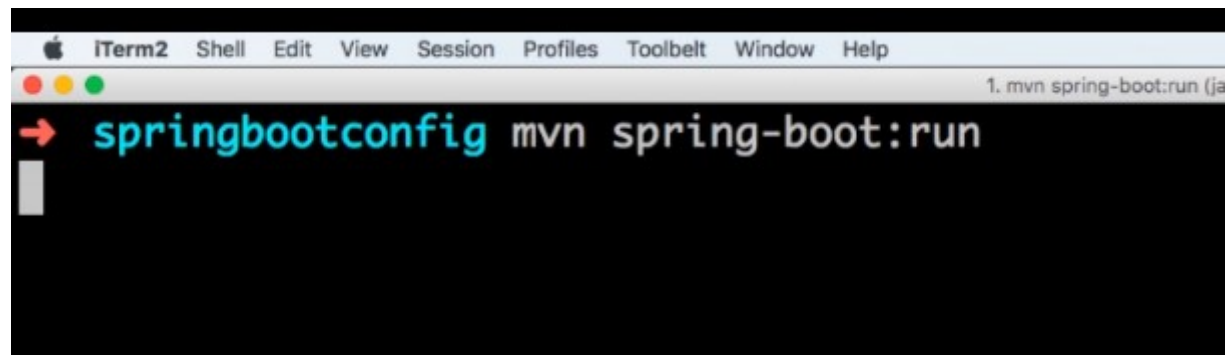
```
basic:  
  value: true  
  message: Dynamic Message YAML  
  number: 100
```





The image shows a code editor with two tabs: 'application-dev.yml' and 'application.properties'. The 'application-dev.yml' tab is active, displaying a YAML configuration for a development environment. The configuration includes a comment, an 'app' section with a message, and a 'spring' section with a 'datasource' configuration. The 'datasource' configuration specifies the driver class name, URL, username, and password. A green checkmark is visible in the top right corner of the editor.

```
1 #DEV environment ✓
2
3 app:
4   message: This is the property file fo the ${spring.application.
5
6 spring:
7   datasource:
8     driver-class-name: org.h2.Driver
9     url: jdbc:h2:mem:db;DB_CLOSE_DELAY=-1
10    username: sa
11    password: sa
```



The image shows an iTerm2 terminal window. The title bar includes the Apple logo, 'iTerm2', and menu items: 'Shell', 'Edit', 'View', 'Session', 'Profiles', 'Toolbelt', 'Window', and 'Help'. The terminal content shows a command prompt with a red arrow pointing to the command 'springbootconfig mvn spring-boot:run'. The command is highlighted in cyan. The terminal also shows the output of the command, which is '1. mvn spring-boot:run (ja'.

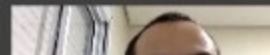
```
→ springbootconfig mvn spring-boot:run
1. mvn spring-boot:run (ja
```



DIGITAL  
INNOVATION  
ONE

# Parte 4: Configurações com command line

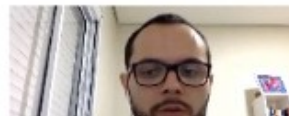
Spring Boot







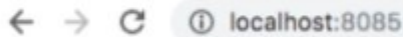
# Uso de command line

- Propriedades do arquivo de configuração na linha de comando.
- Sobrescreve as propriedades definidas no arquivo de configurações padrão.
- Valores passados como argumento na execução do projeto.



# Uso de command line

```
server.port=8081  
spring.application.name=SampleApp
```

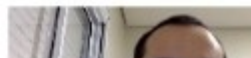
  
**mvn spring-boot:run -Dserver.port=8085**  


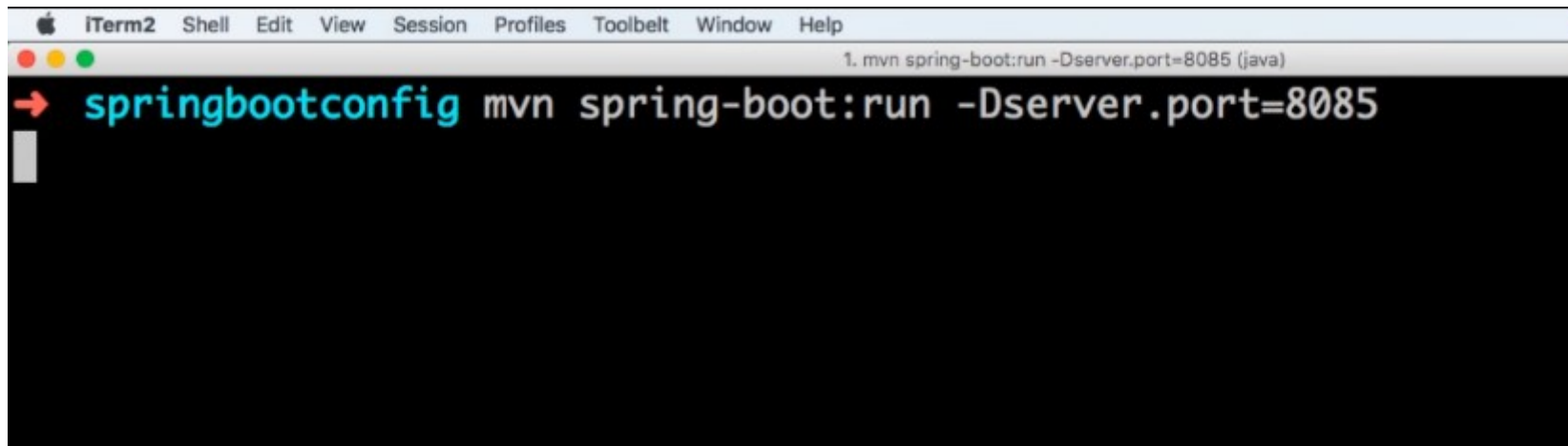
SampleApp



# Exercício juntos

- Passar como argumento a propriedade `server.port=8085`.
- Executar o projeto no terminal com o argumento.
- Abrir o browser no endereço `http://localhost:8085`.





The image shows a screenshot of an iTerm2 terminal window. The title bar at the top includes the Apple logo, the text "iTerm2", and a menu bar with "Shell", "Edit", "View", "Session", "Profiles", "Toolbelt", "Window", and "Help". Below the title bar, the window has three colored window control buttons (red, yellow, green) and a status bar that reads "1. mvn spring-boot:run -Dserver.port=8085 (java)". The main area of the terminal is black, and the command `→ springbootconfig mvn spring-boot:run -Dserver.port=8085` is entered in a light blue/cyan monospace font. A white cursor is positioned at the end of the command.

```
→ springbootconfig mvn spring-boot:run -Dserver.port=8085
```



DIGITAL  
INNOVATION  
ONE

# Parte 5: Configurações com variáveis de ambiente

Spring Boot



# Variáveis de ambiente

- Variável de ambiente pode ser injetada através da anotação @Value no projeto
- Linux e Mac: export comum de variável.

```
export ENV_DB_URL=jdbc:h2:mem:db;DB_CLOSE_DELAY=-1
```

- Windows: padrão de variável de ambiente.



# Variáveis de ambiente

- Injeção com anotação `@Value({NOME_VARIAVEL})`.
- Definição de valor default quando não há variável.

```
@Value("${ENV_DB_URL:NENHUMA}")  
private String dbEnvironmentVariable;
```



# Exercício juntos

- Injetar a variável com @Value em ApplicationController.
- Definição de valor default junto com a anotação @Value.
- Criar método para chamada do novo método e exibir o valor.
- Executar projeto no terminal e exibir no browser



```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AppController {

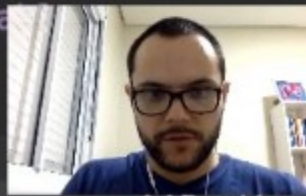
    @Value("${app.message}")
    private String appMessage;

    @Value("${ENV_DB_URL:NENHUMA}")
    private String dbEnvironmentVariable;

    @GetMapping("/")
    public String getAppMessage() {
        return appMessage;
    }

    @GetMapping("/envVar")
    public String getEnvironmentVariable() {
        return "A seguinte variável de ambiente foi passada: " + dbEnvironmentVariable;
    }
}
```

AppController > getEnvironmentVariable()





# Referências

- <https://www.baeldung.com/spring-boot-command-line-arguments>.
- <https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-profiles.html>.
- <https://dzone.com/articles/spring-boot-profiles-1>.
- <https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.





**Qual a anotação utilizada para injetar um profile de um ambiente na criação de um Bean?**

@Property

@BeanProfile

@Properties

@BeanPropertyProfile

@Profile



PRÓXIMA PERGUNTA

**A anotação @Profile deve ser combinada, a nível de classe, com quais das anotações abaixo?**

@Bean



@Inject

@Configuration



Todas alternativas estão corretas.

@yaml

PRÓXIMA PERGUNTA

**Como passamos, através da execução do projeto Spring Boot no terminal, o profile que irá subir com o projeto e sobrescreverá a mesma propriedade definida no arquivo application.properties?**

Nenhum dos comandos acima são válidos. O profile definido no arquivo application.properties tem prioridade maior que a command line.



-Dactive.profiles=dev

-Dspring.profiles.active=dev



Todos as alternativas estão corretas.

-Pactive.profiles=dev

PRÓXIMA PERGUNTA

**Qual anotação podemos usar para injetar o valor da propriedade `server.port` em um atributo do tipo inteiro de uma classe do nosso projeto Spring Boot ?**

@Bean("server.port")

@Property("server.port")

@Valuable("server.port")

@Value("server.port")



@Integer("server.port")

PRÓXIMA PERGUNTA

**No arquivo padrão `application.properties`, qual propriedade configuramos para indicar que desejamos utilizar o profile para o ambiente de desenvolvimento?**

`spring.profiles.activation=dev`

`profiles.active=dev`

`active.profile=dev`

`spring-boot.active.profile=dev`

`spring.profiles.active=dev`



PRÓXIMA PERGUNTA

**Qual o formato padrão do arquivo de configurações do Spring Boot, gerado automaticamente no setup do projeto?**

Properties



java

yaml

groovy

xml

PRÓXIMA PERGUNTA



**Quais são as formas de atribuir um valor de propriedade em um projeto Spring Boot, além do arquivo `application.properties`?**

Variáveis de ambiente.

Todas as alternativas estão corretas.



Dentro do arquivo `pom.xml`.

Através de command line.

Através de um arquivo `application.yml`.

PRÓXIMA PERGUNTA

**Qual a finalidade da anotação @ConfigurationProperties em uma classe de configuração Java?**

Serve para mapeamento de arquivos XML.

Serve apenas para mapear arquivos do tipo YAML.

Todas as estão corretas.



Mapear atributos do valor int em uma configuração.

Mapear propriedades que tem chaves em comum em um arquivo de configuração java.



PRÓXIMA PERGUNTA

**Em qual o diretório o `application.properties` é localizado por padrão em um projeto Spring Boot?**

`src/main/resources`



`src/test/resources`

`src/main/java`

na raiz do projeto

`src/resources`

PRÓXIMA PERGUNTA

**Em um projeto Spring Boot, é necessário adicionar a URL de conexão com o banco de dados através da variável de ambiente DB\_URL. Como podemos usar este valor no nosso arquivo application.properties?**

`datasource.url=${DB_URL}`

`Spring.datasource.url=${DB_URL}`



`Spring.datasource.url=<DB_URL>`

`Spring.datasource.url="DB_URL"`

`Spring.datasource.url=DB_URL`

FINALIZAR