Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

22 - INTEGRANDO JSF COM SPRING BOOT

O Spring Boot foi concebido, inicialmente, para aplicações baseadas em microservices. Mas, como ele se destacou como ponto de partida para aplicativos baseados na estrutura Spring, muitos começaram a se perguntar como integrar o JavaServer Faces (JSF) ao Spring Boot. Nesta aula, vamos unir todas as peças e criar um pequeno programa que permite aos usuários listar e persistir produtos em um banco de dados.

1. Abertura do projeto

1.1. Acesse o menu ***File*** ➔ ***Import...***

1.2. Selecione ***Maven*** ➔ ***Existing Maven Projects*** e clique em ***Next***.

1.3. Clique no botão ***Browse***, localize a pasta **D:\aula1204** e o projeto **product**.

1.4. Clique em ***Finish*** para abrir o projeto.

2. Configuração de Dependências

2.1. Localize o arquivo `pom.xml` com Ctrl + Shift + R.

2.2. Acrescente as bibliotecas da especificação da interface JSF (**api**) e a implementação (**impl**) abaixo na seção `dependencies` do `pom.xml`:

```
<dependency>
	<groupId>javax.faces</groupId>
	<artifactId>jsf-api</artifactId>
	<version>2.0</version>
</dependency>

<dependency>
	<groupId>com.sun.faces</groupId>
	<artifactId>jsf-impl</artifactId>
	<version>2.0.2</version>
</dependency>
```

2.3. Acrescente, também, a dependência do *Primefaces* na sequência:

```
<dependency>
	<groupId>org.primefaces</groupId>
	<artifactId>primefaces</artifactId>
	<version>6.0</version>
</dependency>
```

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

2.4. A quarta dependência (`tomcat-embed-jasper`) é necessária para que a JVM possa analisar e executar a visualização JSF no tempo de execução.

```
<dependency>
	<groupId>org.apache.tomcat.embed</groupId>
	<artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

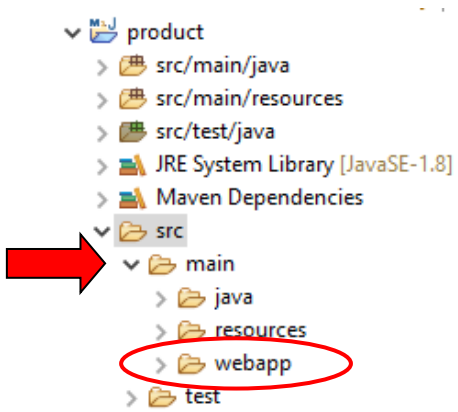
2.5. Por fim, adicione a dependência do driver MySQL.

```
<dependency>
	<groupId>mysql</groupId>
	<artifactId>mysql-connector-java</artifactId>
</dependency>
```

2.6. Salve o arquivo.

3. Configuração do JSF

3.1. Copie a pasta **webapp** de **\\[rede]\Professor\PCII\modelos**, acesse o Eclipse e copie para a pasta **src/main**.



3.2. Localize o arquivo `web.xml` com Ctrl + Shift + R.

3.3. Configure o servlet `FacesServlet`:

```
<servlet>
	<servlet-name>Faces Servlet</servlet-name>
	<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
	<load-on-startup>1</load-on-startup>
</servlet>
```

3.4. Salve o arquivo.

Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

3.5. Abra o arquivo `faces-config.xml` com Ctrl + Shift + R.

3.6. Vamos delegar a criação dos *ManagedBeans* para o *framework*, registrando um `el-resolver`. Copie o texto abaixo e cole no `faces-config.xml`.

```
<application>
        <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>
</application>
```

3.7. Salve o arquivo.

3.8. Como último passo para configurar o JSF com Spring Boot, abra a classe **ProductApplication** do nosso projeto para criar o bean:

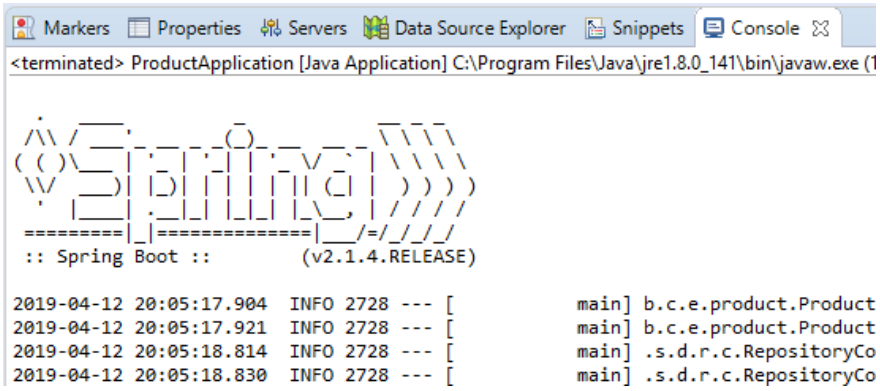
```
@SpringBootApplication
public class ProductApplication {

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

	@Bean
	public ServletRegistrationBean<FacesServlet> servletRegistrationBean() {
		return new ServletRegistrationBean<FacesServlet>(new FacesServlet(), "*.xhtml");
	}
}
```

3.9. Salve a classe.

4. Testando a aplicação

4.1. Para executar o código, clique com o botão direito do mouse na classe **ProductApplication** e selecione as opções ***Run As*** ➔ ***Java Application***.

4.2. Observe a inicialização do *framework* Spring:



Instruções: *Todos os programas devem ser resolvidos utilizando os conceitos de Programação Orientada a Objetos, a linguagem Java™ e os conceitos de Java Persistence API.*

4.3. A inicialização falhará:

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.

Reason: Failed to determine a suitable driver class
```

** O problema ocorreu devido à falta de configuração com o banco de dados. Faremos a configuração posteriormente.

5. GitHub

5.1. Abra o **Prompt de Comando** e acesse o diretório **product**.

5.2. Verifique a situação do arquivo no repositório Git.

5.3. Faça com que os arquivos sejam rastreados pelo Git.

5.4. Verifique a situação do arquivo no repositório Git novamente.

5.5. Execute o comando para gravar as mudanças no repositório com a mensagem **"Integrando JSF com Spring Boot"**.

5.6. Verifique a situação do arquivo no repositório Git novamente.

5.7. Envie os dados para o Github via **`git push`**.