02

# A página de vendas

### Transcrição

Podemos criar a página `vendas.xhtml` agora. Para facilitar, podemos copiar a página `autor.xhtml` e remover tudo que estava dentro da tag `ui:define` :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:p="http://primefaces.org/ui"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <ui:composition template="_template.xhtml">

        <ui:define name="titulo">
                <p:outputPanel>Vendas</p:outputPanel>
        </ui:define>

        <ui:define name="conteudo">

        </ui:define>

    </ui:composition>

</html>
```

## O modelo do gráfico

Na [página][1] do *BarChart* já há um exemplo de como representar o gráfico no `xhtml` , vamos copiar a primeira linha (que representa o *BarChart* na vertical) e colocar no `vendas.xhtml` , dentro do `ui:define` . Mas vamos alterar o atributo `model` do exemplo, para ele utilizar o método `getVendasModel` , que ainda iremos criar na classe `VendasBean` :

```
<ui:define name="conteudo">
    <p:chart type="bar" model="#{vendasBean.vendasModel}" style="height:300px" />
</ui:define>
```

Para implementar o método `getVendasModel` , vamos observar o exemplo que tem no *ShowCase* da classe `ChartView.java` . Ela possui um método chamado `initBarModel` que iremos copiar o seu conteúdo e adaptá-lo a nossa aplicação:

```
public BarChartModel getVendasModel() {

    BarChartModel model = new BarChartModel();
```

```
        ChartSeries boys = new ChartSeries();
        boys.setLabel("Boys");
        boys.set("2004", 120);
        boys.set("2005", 100);
        boys.set("2006", 44);
        boys.set("2007", 150);
        boys.set("2008", 25);

        ChartSeries girls = new ChartSeries();
        girls.setLabel("Girls");
        girls.set("2004", 52);
        girls.set("2005", 60);
        girls.set("2006", 110);
        girls.set("2007", 135);
        girls.set("2008", 120);

        model.addSeries(boys);
        model.addSeries(girls);

        return model;
    }
```

O `model` representa os dados do nosso gráfico e o `series` são os dados específicos do nosso modelo, no nosso caso será o livro (seu título) e a quantidade de vendas.

Você já pode testar e verificar que já funciona, apesar de ainda não termos colocados os dados das vendas. Mas antes, adicione a página de vendas no nosso menu. Após isso, visualize-a:

```
<p:menuitem value="Vendas" action="vendas?faces-redirect=true" icon="fa fa-fw fa-bar-chart" />
```

A primeira `chartSerie`, no exemplo `boys`, é representada em azul, e `girls` em amarelo. Mas vamos representar as vendas adaptando o exemplo para a nossa aplicação. Vamos remover o `chartSerie` `girls` e mudar o `boys`, que passará a ser chamado de `vendaSerie` e terá **Vendas 2016** como *label*:

```
public BarChartModel getVendasModel() {

    BarChartModel model = new BarChartModel();

    ChartSeries vendaSerie = new ChartSeries();
    vendaSerie.setLabel("Vendas 2016");

    model.addSeries(vendaSerie);

    return model;
}
```

Mas ainda não colocamos as vendas no gráfico, pois ainda não temos acesso a elas, para isso vamos criar uma variável `vendas`, que chama o método `getVendas` e iterar por cada venda através de um `foreach`. Para cada venda, adicionamos na série o título do seu livro e a quantidade, através do método `set`:

```
public BarChartModel getVendasModel() {

    BarChartModel model = new BarChartModel();

    ChartSeries vendaSerie = new ChartSeries();
    vendaSerie.setLabel("Vendas 2016");

    List<Venda> vendas = getVendas();

    for (Venda venda : vendas) {
        vendaSerie.set(venda.getLivro().getTitulo(), venda.getQuantidade());
    }

    model.addSeries(vendaSerie);

    return model;
}
```

Agora já vemos um gráfico de exemplo para as vendas dos nossos livros.