07

Mãos à obra: Gráficos para deixar nossa aplicação ainda melhor

Chegou a hora de interagirmos com gráficos. Segue o passo a passo para que você não se perca durante a implementação:

1 - Apesar de já termos as classes `Autor` e `Livro`, para que possamos ver um exemplo mais completo do uso de gráficos com Primefaces criaremos uma nova classe chamada `Venda`. Faz todo sentido, porque podemos mostrar em gráficos, por exemplo, as vendas de um livro. Para sua comodidade, segue a classe `Vendas`:

```
public class Venda {

    private Livro livro;
    private Integer quantidade;

    public Venda(Livro livro, Integer quantidade) {
        this.livro = livro;
        this.quantidade = quantidade;
    }

    public Livro getLivro() {
        return livro;
    }
    public void setLivro(Livro livro) {
        this.livro = livro;
    }
    public Integer getQuantidade() {
        return quantidade;
    }
    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }
}
```

2 - Além da classe `Venda`, vamos criar a classe `VendasBean`, aquela que servirá de suporte para nossa view. Ela terá o método `getVendas` que retorna uma lista do tipo `Venda`. Para cada `Livro` que temos cadastrado em nosso sistema, adicionaremos uma venda aleatória para ele em nossa lista de vendas.

```
@ManagedBean
@ViewScoped
public class VendasBean {

   public List<Venda> getVendas(long seed) {

       List<Livro> livros = new DAO<Livro>(Livro.class).listaTodos();
       List<Venda> vendas = new ArrayList<Venda>();

       Random random = new Random(seed);

       for (Livro livro : livros) {
           Integer quantidade = random.nextInt(500);
           vendas.add(new Venda(livro, quantidade));
```

```
        }

        return vendas;
    }
}
```

Obs: O método `getVendas` é apenas um método auxiliar e poderia ser `private`. A nossa página `xthml` não vai usar esse método diretamente.

3 - Crie `vendas.xhtml` e herdando de `_template.xhtml`. Usando o componente `p:chart` crie o gráfico do tipo `bar`:

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

    </ui:composition>
</html>
```

Veja que a propriedade `model` aponta para `vendasBean.vendasModel` que ainda não existe em `VendasBean`. Lembre-se que não podemos associar diretamente a lista de vendas diretamente com o componente, precisamos encapsulá-la em uma classe especial (`BarCharModel`) do Primefaces.

4 - Crie o método `getVendasModel` em `VendasBean`. O método deve retornar um `BarChartModel`. Como sempre em Java, não esqueça de realizar o import da classe `BarChartModel`:

```
public BarChartModel getVendasModel() {

    BarChartModel model = new BarChartModel();

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

    List<Venda> vendas = getVendas(1234);

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

```
        model.addSeries(vendaSerie);

        ChartSeries vendaSerie2015 = new ChartSeries();
        vendaSerie2015.setLabel("Vendas 2015");

        vendas = getVendas(4321);

        for (Venda venda : vendas) {
            vendaSerie2015.set(venda.getLivro().getTitulo(), venda.getQuantidade());
        }

        model.addSeries(vendaSerie2015);

        return model;
    }

    public List<Venda> getVendas(long seed) {

        List<Livro> livros = new DAO<Livro>(Livro.class).listaTodos();
        List<Venda> vendas = new ArrayList<Venda>();

        Random random = new Random(seed);

        for (Livro livro : livros) {
            Integer quantidade = random.nextInt(500);
            vendas.add(new Venda(livro, quantidade));
        }

        return vendas;
    }
```

5 - Para controlar a posição de legenda do gráfico podemos utilizar o método `setLegendPosition("posicao")` do `BarChartModel` (ou qualquer `ChartModel`). As posições são: "nw" para noroeste, "ne" para nordeste, "sw" para sudoeste, "se" para sudeste, "n" para norte, "e" leste, "s" para sul e finalmente "w" para oeste.

```
        // código omitido...
        model.setTitle("Vendas"); // setando o título do gráfico
        model.setLegendPosition("ne"); // nordeste

        // pegando o eixo X do gráfico e setando o título do mesmo
        Axis xAxis = model.getAxis(AxisType.X);
        xAxis.setLabel("Título");

        // pegando o eixo Y do gráfico e setando o título do mesmo
        Axis yAxis = model.getAxis(AxisType.Y);
        yAxis.setLabel("Quantidade");

        return model;
    }
```

6 - Ainda lembra como se realiza uma exibição animada do gráfico? Experimente!