Java Facelets: Aprenda a c riar templates

Veja neste artigo como criar templates simple usando Facelets.

Com o uso de Facelets torna-se uma tarefa fácil criar templates para uso em nossos sistemas. Ele nos proporciona toda a estrutura necessária para tal tarefa e isso é o que veremos ao longo deste artigo aprendendo a criar e usar os templates através de Facelets.

Para este artigo usaremos 4 tags que são muito importantes, em suma, são elas que irão realizar a base estrutural de nosso template, são elas:

- **ui:insert** - Esta tag serve como um container para definir que área do template pode ser substituída na página que está carregando o template. Suponha, por exemplo, que você possua uma template onde apenas o conteúdo do meio mudará, os elementos de cabeçalho e rodapé serão os mesmos para todas as páginas. Nesse caso você definirá o ui:insert dentro da sua DIV que contém o conteúdo central.

- **ui:define** - Se você tem uma tag para dizer qual conteúdo pode ser sobreposto (ui:insert) você deve ter uma tag para dizer qual conteúdo irá ser inserido neste local, essa é a função da ui:define. Se no template você diz: "Eu quero que apenas dentro do ui:insert sejam colocado conteúdos diferentes do layout", na página que usará o layout você irá definir: "Eu quero que no local deste ui:insert definido pelo layout, fique o meu conteúdo, fazemos isso através do ui:define.
- **ui:include** - Para quem veio do PHP, é fácil perceber que a função desta tag é apenas incluir uma página dentro da outra. Com essa tag você consegue colocar uma página inteira dentro de outra página, dessa forma se você tiver uma página que contém muito código, poderá dividi-la em páginas menores.
- **ui:composition** - Essa tag "chama" o seu template na página que você deseja, ou seja, se você quiser usar o template A na página B, deverá dizer ao ui:composition explicitamente que deseja usar o template A. Porém ele tem algumas especificidades que você deve estar atento: Quando ele é utilizado todo conteúdo fora del é removido assim que sua página é renderizada para o navegado

Na prática

Dada as devidas explicações vamos a prática, e para tal utilizaremos exemplos bem simples que demonstrarão apenas a utilização das tags descritas acima. Na listagem 1 você verá como nosso layout será definido.

Listagem 1: Definição do layout.xhtml

```
<?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://java.sun.com/jsf/html"
```

```
      xmlns:ui="http://java.sun.com/jsf/facelets"
      >

    <h:head>
    </h:head>

    <h:body>

        <div id="page">

            <div id="header">
                <ui:insert name="header" >
                  <ui:include src="/template/basicheader.xhtml" />
                </ui:insert>
            </div>

            <div id="content">
                <ui:insert name="content" >
                  <ui:include src="/template/basiccontent.xhtml" />
                </ui:insert>
            </div>

            <div id="footer">
                <ui:insert name="footer" >
                  <ui:include src="/template/basicfooter.xhtml" />
                </ui:insert>
            </div>

        </div>

    </h:body>
</html>
```


Perceba que o ui:insert está definido para 3 áreas, são elas: cabeçalho, conteúdo do meio e rodapé. Isso significa que estamos permitindo a alteração nessas 3 áreas.

Listagem 2: basicheader.xhtml

```
<?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:ui="http://java.sun.com/jsf/facelets"
      >
    <body>
            <ui:composition>

                <h1>Header Padrão</h1>

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

Listagem 3: Basiccontent.xhtml

```
<?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:ui="http://java.sun.com/jsf/facelets"
      >
    <body>
            <ui:composition>

                <h1>Conteúdo do Meio</h1>

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


Listagem 4: Basicfooter.xhml

```
<?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:ui="http://java.sun.com/jsf/facelets"
      >
    <body>
            <ui:composition>
```

```
                <h1>Footer padrão</h1>

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

Note que no basicheader.xhtml usamos o ui:composition, o que significa que todo código que está fora desta tag não será renderizado, então veja como ficará nosso basicheader.xhtml depois de renderizado.

Listagem 5: Basicheader.xhtml depois de renderizado

```
<ui:composition>

        <h1>Header padrão</h1>

</ui:composition>
```

Enfim um layout básico foi criado e suas áreas foram definidas, mas e como fazemos para utilizá-lo em nossas páginas ? Vamos criar uma página chamada índex.xhtml e definir nosso layout nesta.

Listagem 6: index.xhtml usando layout.xhtml

```
<?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://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      >
    <h:body>

        <ui:composition template="template/layout.xhtml">

        </ui:composition>
```

```
    </h:body>

</html>
```

Quando carregamos o layout acima sem nenhum ui:define (que é o correspondente ao ui:insert) será carregado o conteúdo padrão definido no ui:insert do layout (se existir), no caso da listagem 6, é isso que ocorrerá, todo conteúdo padrão será carregado. Mas na listagem 7 vamos fazer algo mais robusto e inserir nosso próprio conteúdo especifico para a página cadastro.xhtml.

Listagem 7: Cadastro.xhtml

```
<?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://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      >
    <h:body>

        <ui:composition template="/template/layout.xhtml">

                <ui:define name="content">
                        formulario de cadastro ficará aqui
                </ui:define>

        </ui:composition>

    </h:body>

</html>
```

Perceba no código da listagem 7 que definimos apenas o ui:define para o ui:insert de nome “content” definido em nosso layout.xhtml. Dessa forma estamos sobrescrevendo todo conteúdo do ui:insert de nome “content”.

CONCLUSÃO

O uso do Facelets para gerenciamos da sua camada de visualização (view) faz-se necessário ao visto que este ajuda o desenvolvedor a focar no negócio e deixar o trabalho de design para outra área.

Veja também

- Facelets – Visão Geral Conceitos, Utilização e Criação de Componentes
- Facelets passo a passo no NetBeans

por Ronaldo Lanhellas

Expert em Java e programação Web