

## Tutorial para criação de componentes JSF Facelets

Por Érico GR – 07/08/2007

Nível: Intermediário

Este tutorial tem como objetivo criar um simples componente que “renderiza” uma tag *label* utilizando recursos do framework facelets, mas nada impede que componentes mais complexos sejam criados. Tentei simplificar ao máximo o exemplo, criando até um mapa de relacionamento entre os arquivos de configuração e o código Java. Pressuponho que o leitor que deseja seguir este tutorial já tenha conhecimento básico sobre java e facelets. Para mais informações, consulte as referências no final deste documento.

Neste exemplo vamos criar um componente chamado FirstJsfTest que renderiza um simples *label*.

Exemplo:

```
<label>Teste 123</label>
```

Premissas

1. JDK 1.5 ou superior
2. Biblioteca JSF 1.2 (aqui utilizamos JSF1.2\_04)
3. Seu projeto já deve estar configurado para utilizar a combinação de JSF + Facelets

Criação do componente

1. Criar uma classe que estenda *UIComponent*
2. Programar métodos para salvar e restaurar o estado do componente
3. Criar método que retorne a família do componente
4. Criar uma classe que estenda *UIComponentELTag*
5. Configurar as propriedades que o componente JSF utilizará
6. Programar métodos para retornar o tipo de componente e renderizador utilizado
7. Programar um renderizador
8. Criar um arquivo *.tld* (Descritor de biblioteca de tags)
9. Registre o componente no *faces-config.xml*
10. Criar um arquivo de configuração de biblioteca facelets e registrá-lo
11. Usando o componente
12. Relacionamento entre código java e os descritivos XML
13. Referências

### 1. Crie uma classe que estenda *UIComponent*

Para começar, vamos criar a classe *UIFirstJsfComponent* que estende a classe *UIOutput* (que é subclasse de *UIComponent*) e criaremos também os atributos do componente.

```
[UIFirstJsfComponent.java]
```

```
package net.sf.quizOnline.tags;
```

```
import javax.faces.component.UIOutput;
import javax.faces.context.FacesContext;

public class UIFirstJsfComponent extends UIOutput {
    private String m_Name;

    public UIFirstJsfComponent() {
    }

    public String getName() {
        return m_Name;
    }

    public void setName(String name) {
        m_Name = name;
    }
    ...
}
```

## 2. Programar métodos para salvar e restaurar o estado do componente

Abaixo está o trecho de código que fica responsável por salvar e restaurar o estado do componente. Salvamos e restauramos os atributo do nosso componente através dos métodos `saveState` e `restoreState` da classe `UIOutput`.

```
[UIFirstJsfComponent.java]

...
@Override
public Object saveState(FacesContext facesContext) {
    Object[] values = new Object[2];

    values[0] = super.saveState(facesContext);
    values[1] = getName();

    return (Object)values;
}

@Override
public void restoreState(FacesContext facesContext,
    Object state)
{
    Object[] values = (Object[])state;

    super.restoreState(facesContext, values[0]);
    setName((String)values[1]);
}
...
}
```

## 3. Retornar a família do componente

Precisamos do método `getFamily()` para retornar a família do componente. O retorno deste método será usado posteriormente para registrar o renderizador no arquivo de configuração do Jsf. Podemos retornar `COMPONENT_FAMILY` que é uma constante com o valor `javax.faces.Output`.

[UIFirstJsfComponent.java]

```
...  
    public String getFamily() {  
        return COMPONENT_FAMILY;  
    }  
...
```

#### 4. Criar uma classe que estenda *UIComponentELTag*

Nos passos anteriores, o componente foi criado. Agora precisamos definir a tag (marcação que ficará na página que estamos criando) que fará uma ponte entre o Jsp e o Jsf. No trecho de código abaixo, está também a propriedade *name* que será utilizada em nossa tag para passar valor ao componente com propriedade de mesmo nome. Depois mapearemos a tag através do arquivo de configuração de extensão .tld (tag library descriptor).

[FirstJsfTag.java]

```
package net.sf.quizOnline.tags;  
  
import javax.faces.webapp.UIComponentELTag;  
import javax.faces.component.UIComponent;  
  
public class FirstJsfTag extends UIComponentELTag {  
    private String m_Name;  
  
    public FirstJsfTag() {  
    }  
  
    public String getName() {  
        return m_Name;  
    }  
  
    public void setName(String name) {  
        this.m_Name = name;  
    }  
    ...  
}
```

#### 5. Configurar as propriedades que o componente JSF utilizará

Vamos armazenar o valor configurado para a tag no componente que criamos como mostra o código abaixo:

[FirstJsfTag.java]

```
...  
    @Override  
    protected void setProperties(UIComponent component) {  
        super.setProperties(component);  
        ((UIFirstJsfComponent) component).setName(getName());  
    }  
...
```

Como podemos ver, *component* é uma instância da classe *UIFirstJsfComponent* onde configuramos seus atributos pelos atributos da tag *FirstJsfTag*.

## 6. Programar métodos para retornar o tipo de componente e renderizador utilizado

O retorno deste método será utilizado nos arquivos de configuração do Jsf e Facelets.

[FirstJsfTag.java]

```
...  
    public String getComponentType() {  
        return "net.sf.quizOnline.HtmlFirstJsfTag";  
    }  
  
    public String getRendererType() {  
        return "net.sf.quizOnline.FirstJsfTag";  
    }  
...
```

## 7. Programar um renderizador

O renderizador escreverá a saída, que no nosso caso será codificação em HTML. Esta classe estende `Renderer` e provê uma série de métodos interessantes que podem ser explorados pelo programador, mas este não será o foco.

Este renderizador apenas escreve na página um label com o valor definido na propriedade `name`. A saída é esta: `<label>valor qualquer</label>`

[FirstJsfRenderer.java]

```
package net.sf.quizOnline.tags;  
  
import java.io.IOException;  
  
import javax.faces.component.UIComponent;  
import javax.faces.context.FacesContext;  
import javax.faces.context.ResponseWriter;  
import javax.faces.render.Renderer;  
  
public class FirstJsfRenderer extends Renderer {  
  
    @Override  
    public void encodeBegin(FacesContext context,  
        UIComponent component) throws IOException  
    {  
        UIFirstJsfComponent firstJsfComponent =  
            (UIFirstJsfComponent) component;  
        ResponseWriter writer = context.getResponseWriter();  
        writer.startElement("label", component);  
        writer.write(firstJsfComponent.getName());  
        writer.endElement("label");  
        writer.flush();  
    }  
}
```

## 8. Criar um arquivo .tld

Depois de ter criado o componente, temos que criar seu descritor, o arquivo tld (Tag Library Descriptor) chamado firstJsfTag.tld (para este exemplo). Normalmente este arquivo fica na pasta da sua aplicação web em /WEB-INF/taglib.

Exemplo: /WEB-INF/taglib/firstJsfTag.tld

[firstJsfTag.tld]

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag
Library 1.2//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_2.dtd">

<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>2.0</jsp-version>
  <short-name>firstJsfTag</short-name>
  <uri>http://net.sf.quizOnline/component/test/tags</uri>
  <description>Teste de taglibs jsf</description>

  <tag>
    <name>firstJsfTag</name>
    <tag-class>net.sf.quizOnline.tags.FirstJsfTag</tag-class>
    <attribute>
      <name>name</name>
      <required>true</required>
    </attribute>
  </tag>
</taglib>
```

## 9. Registre o componente no faces-config.xml

É necessário registrar o componente no arquivo de configuração do Jsf faces-config.xml. Normalmente este arquivo fica na pasta da sua aplicação web em /WEB-INF. Exemplo: /WEB-INF/faces-config.xml

[faces-config.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="1.2" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">
  <!-- Custom Tags -->

  <!-- Componente de UI do JSF para teste -->
  <component>
    <component-type>net.sf.quizOnline.HtmlFirstJsfTag</component-
type>
    <component-class>
      net.sf.quizOnline.tags.UIFirstJsfComponent
    </component-class>
  </component>
  <render-kit>
    <renderer>
```

```
<component-family>javax.faces.Output</component-family>
<renderer-type>net.sf.quizOnline.FirstJsfTag</renderer-
type>
  <renderer-class>
    net.sf.quizOnline.tags.FirstJsfRenderer
  </renderer-class>
</renderer>
</render-kit>
<!-- IMPORTANTE: foram ocultadas outras configuração não relevantes!
-->
</faces-config>
```

## 10. Criar um arquivo de configuração de biblioteca facelets e registrá-lo

Agora precisamos configurar o facelets para utilizar o componente Jsf criado. Para criar essa ponte entre os dois, declaramos em um arquivo de configuração o nome do componente, seu tipo e o tipo do renderizador. Normalmente este arquivo fica na pasta da sua aplicação web em /WEB-INF/taglib. Exemplo: /WEB-INF/taglib/firstJsfTag.taglib.xml

[firstJsfTag.taglib.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE facelet-taglib PUBLIC "-//Sun Microsystems, Inc.//DTD
Facelet Taglib 1.0//EN" "facelet-taglib_1_0.dtd" >
<facelet-taglib>
  <namespace>http://quizOnline.sf.net/firstJsfTag</namespace>

  <tag>
    <tag-name>firstJsfTag</tag-name>
    <component>
      <component-type>
        net.sf.quizOnline.HtmlFirstJsfTag
      </component-type>
      <renderer-type>
        net.sf.quizOnline.FirstJsfTag
      </renderer-type>
    </component>
  </tag>
</facelet-taglib>
```

Note que os tipos declarados neste arquivo são os mesmos retornados pelos métodos presentes na classe que estende `UIComponentELTag` (`getComponentType()` e `getRendererType()`).

É preciso também registrar este arquivo como uma biblioteca nos parâmetros de contexto do facelets. Caso exista mais de uma biblioteca, podemos separá-las com “;” ponto e vírgula. Normalmente este arquivo fica na pasta da sua aplicação web em /WEB-INF. Exemplo: /WEB-INF/web.xml

[web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
...
<context-param>
...
<param-name>facelets.LIBRARIES</param-name>
<param-value>
    /WEB-INF/taglib/firstJsfTag.taglib.xml
</param-value>
...
</context-param>
...
</web-app>
```

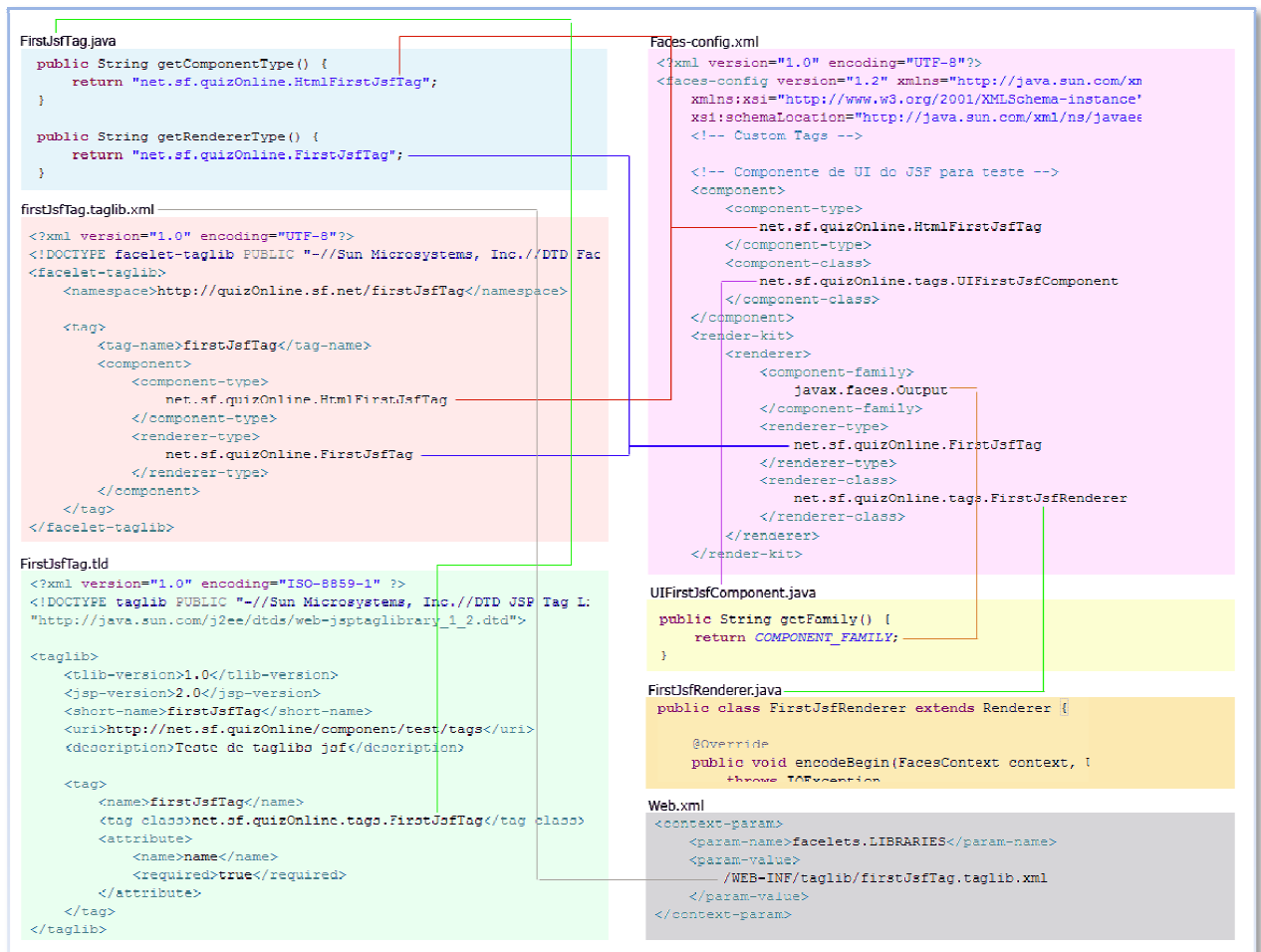
## 11. Usando o componente

Agora podemos usar o componente nas páginas xhtml. Abaixo segue o exemplo de uma página usando o componente.

```
[teste.xhtml]
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!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"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:c="http://java.sun.com/jstl/core"
    xmlns:fjt="http://quizOnline.sf.net/firstJsfTag">
<ui:composition template="/pages/public/template/default.xhtml">
    <ui:define name="pageTitle">
        <h:outputText value="Pagina de teste 3" />
    </ui:define>
    <ui:define name="corpo">
        <f:view>
            <fjt:firstJsfTag name="teste123"/>
        </f:view>
    </ui:define>
</ui:composition>
</html>
```

## 12. Relacionamento entre código Java e os descritivos XML

Abaixo temos uma figura que exibe trechos de código e de arquivos XML e como eles se relacionam para a composição do componente desenvolvido. Podemos acompanhar se esquecemos de criar algum arquivo ou configuração.



Mapa de referência de arquivos de configuração

Note que a maioria dos arquivos não estão completos, deixei somente a parte interessante para visualizarmos o relacionamento entre eles. Por exemplo, o arquivo FirstJsfTag.java não tem a definição de pacotes, as importações, a declaração de classes, etc.

### 13. Referências

Abaixo estão alguns sites que serviram de referência para montar este documento.

- Jdk <http://www.sun.com/java>
- facelets <http://facelets.java.net>
- Tomcat <http://tomcat.apache.org>
- JSF 1.2 (Sun) <https://glassfish.java.net>
- JSF Central <http://www.jsfcentral.com/>
- Introdução ao facelets [http://www.jsfcentral.com/articles/facelets\\_1.html](http://www.jsfcentral.com/articles/facelets_1.html)