Desenvolvimento para Servidores-II

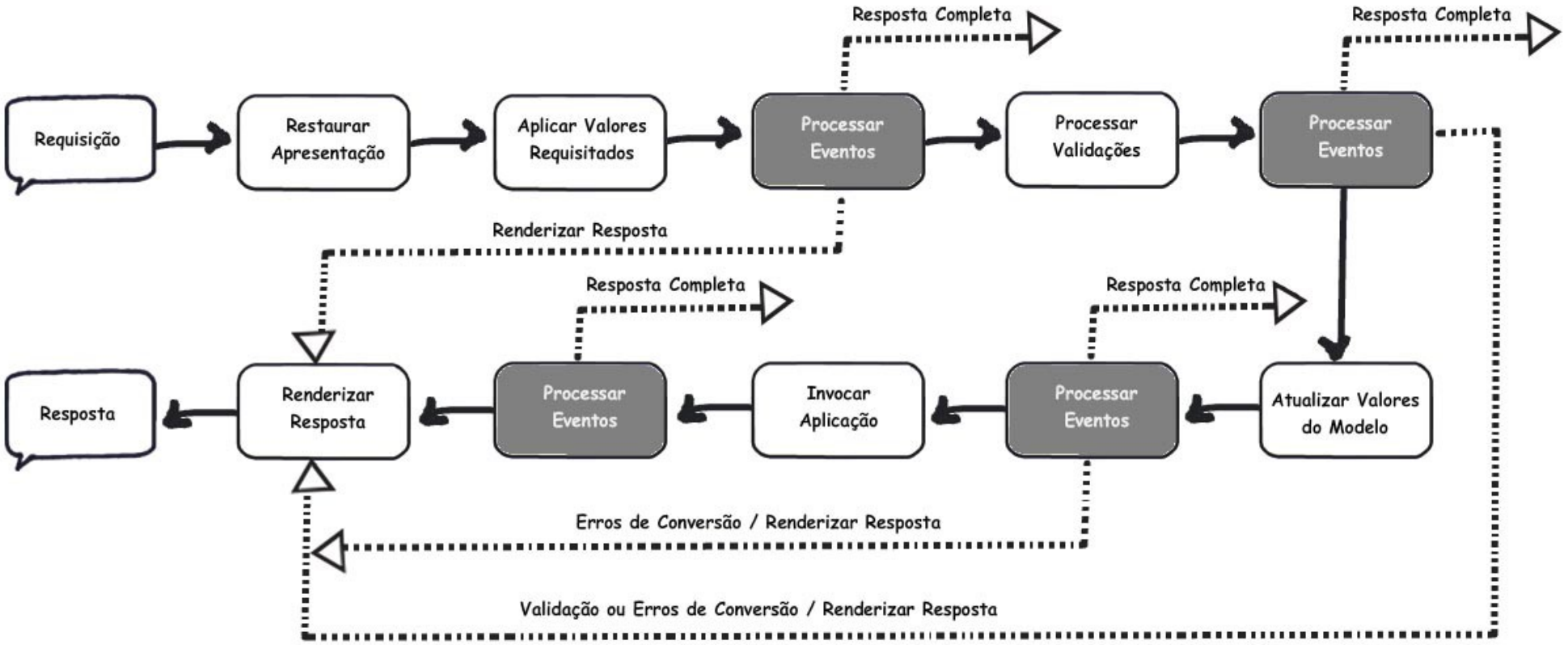
Tratamento de Eventos

Neste tópico abordaremos o tratamento de eventos gerados pelo usuário, tais como a seleção de um item de menu e o clique num botão.

Prof. Ciro Cirne Trindade

Ciclo de vida JSF (1/2)

- O ciclo de vida JSF é composto por 6 fases



Ciclo de vida JSF (2/2)

- Após a fase “Aplicar Valores Requisitados”, a implementação JSF pode gerar eventos
- Esses eventos podem ser tratados por *listeners*

Registrando *listeners* a componentes

- Listeners podem ser classes ou métodos de um *managed bean*
  - Se um *listener* é um **método** de um *managed bean*, a referência a este método pode ser feita tanto pelo **atributo** `valueChangeListener` ou pelo **atributo** `actionListener`
  - Se o *listener* é uma **classe** a referência pode ser feita tanto pela **tag** `valueChangeListener` ou **pela** tag `actionListener` que devem estar no corpo da tag do componente ao qual se quer associar o *listener*

Eventos de mudança de valor (1/2)

- Componentes em uma aplicação web geralmente dependem uns dos outros
- Por exemplo, a mudança no valor de um campo pode influenciar o conteúdo de outro
- Podemos usar o atributo `valueChangeListener` para associar um método à mudança do valor de um componente

Eventos de mudança de valor (2/2)

- O método associado a um evento de mudança de valor deve esperar um argumento do tipo `ValueChangeEvent`
- Através de um objeto do tipo `ValueChangeEvent` é possível recuperar o novo valor componente (`getNewValue()`) e o valor antigo dele (`getOldValue()`)

Exemplo: seleção do idioma

- Vamos implementar uma aplicação em que o usuário possa escolher entre o idioma português ou inglês
- Todas as mensagens e rótulos serão definidos em arquivos de propriedades
  - messages_pt_BR.properties (português)
  - messages_en.properties (inglês)

Arquivos de propriedades

- messages_pt_BR.properties

```
titulo=Título
nome=Nome
idioma=Idioma
botao=Enviar
```

- messages_en.properties

```
titulo=Title
nome=Name
idioma=Language
botao=Submit
```

Registrando as mensagens no faces-config.xml

```
<faces-config version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-
facesconfig_2_2.xsd">
    <application>
        <locale-config>
            <default-locale>pt-br</default-locale>
            <supported-locale>en</supported-locale>
        </locale-config>
        <resource-bundle>
            <base-name>messages.messages</base-name>
            <var>msgs</var>
        </resource-bundle>
    </application>
</faces-config>
```

Managed bean: Form.java (1/2)

```
package beans;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ValueChangeEvent;
import javax.faces.model.SelectItem;

@ManagedBean
@RequestScoped
public class Form {
    private String nome;
    private Locale localizacao;
    private static final Locale[] countries =
          { Locale.forLanguageTag("pt-br"), Locale.ENGLISH };

    public Form() { }

    public String getNome() { return nome; }
```

Managed bean: Form.java (2/2)

```
    public void setNome(String nome) {
        this.nome = nome;
    }

    public Locale getLocalizacao() { return localizacao; }

    public void setLocalizacao(Locale localizacao) {
        this.localizacao = localizacao;
    }

    public void mudouIdioma(ValueChangeEvent event) {
        FacesContext.getCurrentInstance().getViewRoot().
             setLocale((Locale)event.getNewValue());
    }

    public List<SelectItem> getIdiomasSuportados() {
        List<SelectItem> idiomas = new ArrayList<>();
        for (Locale loc : countries)
            idiomas.add(new SelectItem(loc,
                        loc.getDisplayLanguage()));
        return idiomas;
    }
}
```

index.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://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
   <h:head><title>#{msgs.titulo}</title></h:head>
   <h:body>
      <h:form>
         <h:panelGrid columns="2">
            #{msgs.idioma}:
            <h:selectOneMenu onchange="submit()"
               value="#{form.localizacao}"
               valueChangeListener="#{form.mudouIdioma}"
               converter="converter.LocalizacaoConverter">
                <f:selectItems value="#{form.idiomasSuportados}" />
            </h:selectOneMenu>
            #{msgs.nome}:
            <h:inputText value="#{form.nome}"/>
         </h:panelGrid>
         <h:commandButton action="/index" value="#{msgs.botao}"/>
      </h:form>
   </h:body>
</html>
```

Força a submissão do formulário para que a página seja atualizada (poderia ser substituída por uma requisição AJax)

Conversor que converte `String` para `Locale` e vice-versa

LocalizacaoConverter.java

```
package converter;

import java.util.Locale;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;

@FacesConverter("converter.LocalizacaoConverter")
public class LocalizacaoConverter implements Converter {

    @Override
    public Object getAsObject(FacesContext context,
                              UIComponent component, String value) {
        return Locale.forLanguageTag(value);
    }

    @Override
    public String getAsString(FacesContext context,
                              UIComponent component, Object value) {
        return ((Locale)value).toLanguageTag();
    }
}
```

Exemplo: seleção do idioma

- Para definir o idioma da aplicação como um todo e não só de uma página, utilize o atribute `locale` da tag `<f:view>`, preferencialmente no *template* das páginas

Eventos de ação (1/3)

- Eventos de ação são disparados através de botões e links
- Eventos de ação são invocados na fase “Invocar Aplicação”, perto do fim do ciclo de vida
- Para adicionar um *action listener* a um botão ou link pode-se usar o atributo **`actionListener`**

Eventos de ação (2/3)

- É importante distinguir *action listeners* de *actions*
  - *actions* são projetados para lógica de negócios e participam do controle de navegação
  - *action listeners* tipicamente executam lógica de interface e não participam do controle de navegação
- *Action listeners* as vezes trabalham em conjunto com *actions* quando uma ação precisa de informações a respeito da interface

Eventos de ação (3/3)

- O método que implementa um *action listener* espera um argumento do tipo `ActionEvent` que representa a ativação de um componente de interface com o usuário
- A classe `ActionEvent` define métodos tais como o `getComponent()` que devolve o componente (`UIComponent`) que gerou o evento

Exemplo: clique em uma imagem (1/2)

- Vamos implementar uma aplicação que utiliza um *action* e um *action listener* para reagir a cliques do mouse em uma imagem encaminhando a requisição a uma página JSF

Exemplo: clique em uma imagem (2/2)

Managed bean: Form.java (1/2)

```
package beans;

import java.awt.Point;
import java.awt.Rectangle;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

@ManagedBean
@RequestScoped
public class Form {
    private String outcome;
    private static final Rectangle APPLE =
                                new Rectangle(0, 0, 270, 340);
    private static final Rectangle MICROSOFT =
                                new Rectangle(280, 0, 270, 340);

    public Form() {
    }
```

Managed bean: Form.java (2/2)

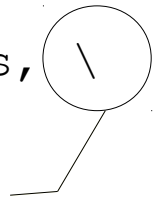
```
    public void trataCliqueMouse(ActionEvent e) {
        FacesContext context =
                FacesContext.getCurrentInstance();
        String clientId =
                e.getComponent().getClientId(context);
        Map<String, String> requestParams =
      context.getExternalContext().getRequestParameterMap();
        int x = new Integer((String)
                    requestParams.get(clientId + ".x"));
        int y = new Integer((String)
                    requestParams.get(clientId + ".y"));
        outcome = null;
        if (APPLE.contains(new Point(x, y)))
            outcome = "apple";
        else if (MICROSOFT.contains(new Point(x, y)))
            outcome = "microsoft";
    }

    public String go() {
        return outcome;
    }
}
```

messages.properties

```
indexTitle=Apple x Microsoft
indexHeader=Para informações sobre as empresas, \
            click nos seus logos
appleTitle=Apple
appleHeader=Apple
microsoftTitle=Microsoft
microsoftHeader=Microsoft
linkBack=Voltar

appleText=A Apple fundada em 1976 por Steve Jobs e Steve \
          Wozniak. O primeiro computador da Apple foi o \
          Apple-I, um Computador artesanal. O primeiro \
          computador da Apple produzido em escala \
          comercial foi o Apple-II (1977).
microsoftText=A Microsoft foi fundada em 1974 por dois \
              estudantes de Havard, Bill Gates e Paul \
              Allen. Em 1975 a Microsoft comercializou um \
              interpretador do Basic para o Mits Altair. \
              Em 1981 licenciou o Sistema Operacional \
              MS-DOS para o IBM PC.
```



Indica que a linha continua abaixo

_template.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://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <meta http-equiv="Content-Type"
              content="text/html; charset=UTF-8" />
        <h:outputStylesheet name="./css/default.css"/>
        <h:outputStylesheet name="./css/cssLayout.css"/>
        <title><ui:insert name="title">título </ui:insert></title>
    </h:head>
    <h:body>
        <div id="top"><ui:insert name="top">Top</ui:insert></div>

        <div id="content" class="center_content">
            <ui:insert name="content">Content</ui:insert>
        </div>

        <div id="bottom">
            <ui:insert name="bottom">Bottom</ui:insert>
        </div>
    </h:body>
</html>
```

index.xhtml

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

   <ui:define name="title">#{msgs.indexTitle}</ui:define>
   <ui:define name="top"><h1>#{msgs.indexHeader}</h1>
   </ui:define>
   <ui:define name="content">
       <h:form>
           <h:commandButton
                image="/resources/img/apple_x_microsoft.jpg"
                actionListener="#{form.trataCliqueMouse}"
                action="#{form.go}"/>
       </h:form>
   </ui:define>
   <ui:define name="bottom"></ui:define>

</ui:composition>
```

apple.xhtml

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

    <ui:define name="title">#{msgs.appleTitle}</ui:define>
    <ui:define name="top"><h1>#{msgs.appleHeader}</h1>
    </ui:define>
    <ui:define name="content">
        <h:panelGrid columns="2">
            <h:graphicImage name="apple_logo.jpg"
                            library="img"/>
            <p>#{msgs.appleText}</p>
        </h:panelGrid>
    </ui:define>
    <ui:define name="bottom">
        <h:link value="#{msgs.linkBack}" outcome="/index"/>
    </ui:define>
</ui:composition>
```

microsoft.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<ui:composition xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
                template="./_template.xhtml"
                xmlns:h="http://xmlns.jcp.org/jsf/html">
    <ui:define name="title">#{msgs.microsoftTitle}
    </ui:define>
    <ui:define name="top"><h1>#{msgs.microsoftHeader}</h1>
    </ui:define>
    <ui:define name="content">
        <h:panelGrid columns="2">
            <h:graphicImage name="microsoft_logo.jpg"
                            library="img"/>
            <p>#{msgs.microsoftText}</p>
        </h:panelGrid>
    </ui:define>
    <ui:define name="bottom">
        <h:link value="#{msgs.linkBack}" outcome="index"/>
    </ui:define>
</ui:composition>
```

Event listener tags (1/2)

- Até agora, adicionamos *listeners* de ações e mudanças de valores a componentes através dos atributos `actionListener` e `valueChangeListener`
- Entretanto, também é possível fazer o mesmo através das tags:
  - `f:actionListener`
  - `f:valueChangeListener`

Event listener tags (2/2)

- As tags `f:valueChangeListener` e `f:actionListener` são análogas aos atributos `valueChangeListener` e `actionListener`, respectivamente
- Mas associamos classes e não métodos a essas tags
- A vantagem das tags sobre os atributos é que podemos associar mais de um *listener* a um componente

A tag f:valueChangeListener (1/3)

- Para associar um *listener* a um evento de mudança de valor através do atributo `valueChangeListener` fizemos:

```
<h:selectOneMenu onchange="submit()"
      valueChangeListener="#{form.mudouIdioma}">
      ...
</h:selectOneMenu>
```

A tag f:valueChangeListener (2/3)

- Para fazer a mesma coisa através da tag `f:valueChangeListener` faríamos:

```
<h:selectOneMenu onchange="submit()">
    <f:valueChangeListener
                type="listener.IdiomaListener"/>
    ...
</h:selectOneMenu>
```

O atributo `type` indica a classe que implementa o *listener*

A tag f:valueChangeListener (3/3)

- A classe associada a um `f:valueChangeListener` deve implementar a interface `ValueChangeListener`
- Esta interface define um único método:
  - `void processValueChange(ValueChangeEvent)`

Classe IdiomaListener.java

```
package listener;

import java.util.Locale;
import javax.faces.context.FacesContext;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ValueChangeEvent;
import javax.faces.event.ValueChangeListener;

public class IdiomaListener implements ValueChangeListener {

    @Override
    public void processValueChange(ValueChangeEvent event)
                                throws AbortProcessingException {
        FacesContext.getCurrentInstance().getViewRoot().
            setLocale((Locale)event.getNewValue());
    }

}
```

Tag `f:actionListener` (1/4)

- A tag `f:actionListener` é análoga a `f:valueChangeListener`
- Ela também possui um atributo `type` que especifica o nome da classe
- Esta classe deve implementar o interface `ActionListener`
- Esta interface define um único método:
  - `void processAction(ActionEvent)`

Tag f:actionListener (2/4)

- Para associar um *listener* a um evento de ação através do atributo `actionListener` fizemos:

```
<h:commandButton
    image="/resources/img/apple_x_microsoft.jpg"
    actionListener="#{form.trataCliqueMouse}"
    action="#{form.go}"/>
```

Tag f:actionListener (3/4)

- Para fazer a mesma coisa através da tag f:actionListener faríamos:

```
<h:commandButton
    image="/resources/img/apple_x_microsoft.jpg"
    action="#{form.go}">
    <f:actionListener
        type="listener.CliqueListener"/>
</h:commandButton>
```

Tag f:actionListener (4/4)

- É possível especificar múltiplos *listeners* para um componente através de múltiplas tags `f:actionListener` ou `f:valueChangeListener`
- Por exemplo, poderíamos adionar outro *action listener* ao nosso exemplo anterior dessa forma:

```
<h:commandButton action="#{form.go}"
   image="/resources/img/apple_x_microsoft.jpg">
   <f:actionListener
      type="listener.CliqueListener"/>
   <f:actionListener
      type="listener.LoggerListener"/>
</h:commandButton>
```

Referências

- GEARY, David; HORSTMANN, Cay. *Core JavaServer Faces*. 3. ed., Prentice-Hall, 2010.
- ORACLE Corporation. *The Java EE 7 Tutorial*. Disponível em: https://docs.oracle.com/javaee/7/JEETT.pdf, 2014.