JavaServer Faces

Ciclo de vida de requisicoes em JSF

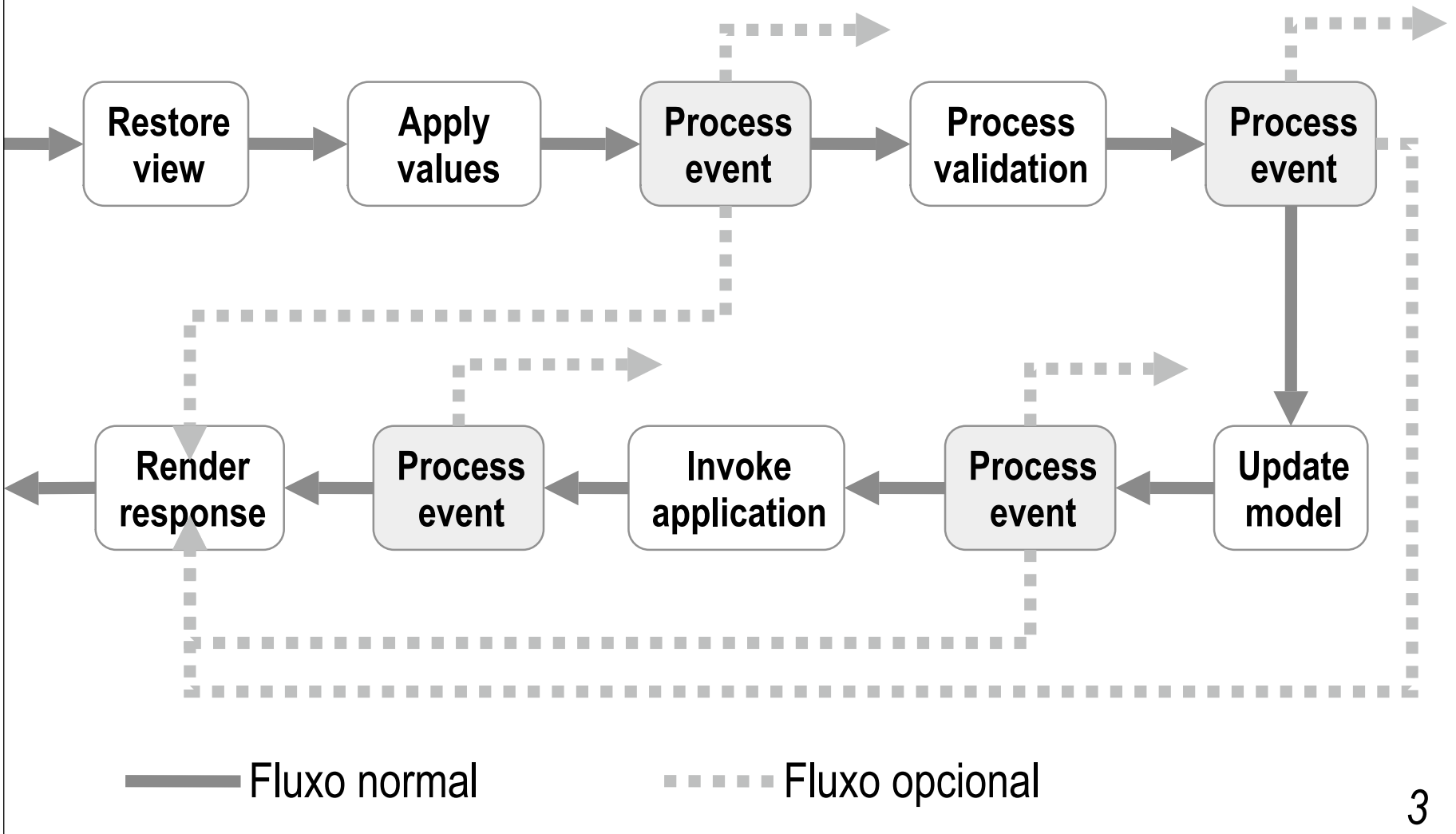
Alteracoes e atualizacoes: Gustavo Wagner © *2008*
gugawag@gmail.com

Slides originais gentilmente cedidos por: Frederico Costa Guedes Pereira © 2007
fredguedespereira@gmail.com

Ciclo de vida da requisição

- Uma requisição no JSF passa por várias fases
  - ***Restore view*** → ***Apply request values*** → ***Process validation*** → ***Update model values*** → ***Invoke application*** → ***Render response***
- JSF transforma os detalhes de uma requisição em uma visão (view) de mais alto nível, formada por **componentes** e **eventos**.
  - Na fase ***Invoke application*** haverá uma árvore de componentes montada e atualizada, todas as validações foram realizadas e os backing beans e objetos de negócio foram atualizados

Ciclo de vida da requisição



Ciclo de vida da requisição

- Fase ***Restore view***
  - Localiza ou cria uma árvore de componentes para a visão que enviou a requisição. Componentes de comando geram eventos de ação (action) nesta fase
  - Tipos de eventos lançados: Phase
- Fase ***Apply request values***
  - Atualiza os valores dos componentes com os valores recebidos na requisição (usando conversores)
  - Tipos de eventos lançados: Phase, Data model e Action

Ciclo de vida da requisição

- Fase ***Process validators***
  - Solicita a cada componente que se valide. Mensagens de erro de validação podem ser produzidas.
  - Tipos de eventos lançados: Phase Data model e Value-changed
- Fase ***Update model values***
  - Atualiza os *backing beans* e objetos do modelo associados aos componentes. Mensagens de erro de conversões podem ser produzidas
  - Tipos de eventos lançados: Phase e Data model

Ciclo de vida da requisição

- Fase ***Invoke application***
  - Executa os *action listeners* registrados e também o *action listener default* que determina a próxima página
  - Tipos de eventos lançados: Phase e Action
- Fase ***Render response***
  - Produz a visão utilizando a tecnologia corrente de produção de visões (JSP, por exemplo)
  - Tipos de eventos lançados: Phase

Fase 1: *Restore view*

- A ação mais importante desta fase é ***identificar a visão que gerou a requisição***
  - A árvore de componentes desta visão (na sessão) será recuperada ou construída.
  - A visão é armazenada no FacesContext
  - Os eventos gerados pelos componentes precisam ser identificados e associados aos componentes desta visão
  - Instancias de componentes de UI associados (com binding) a *bacink beans* são sincronizados nesta fase

Fase 1: *Restore view*

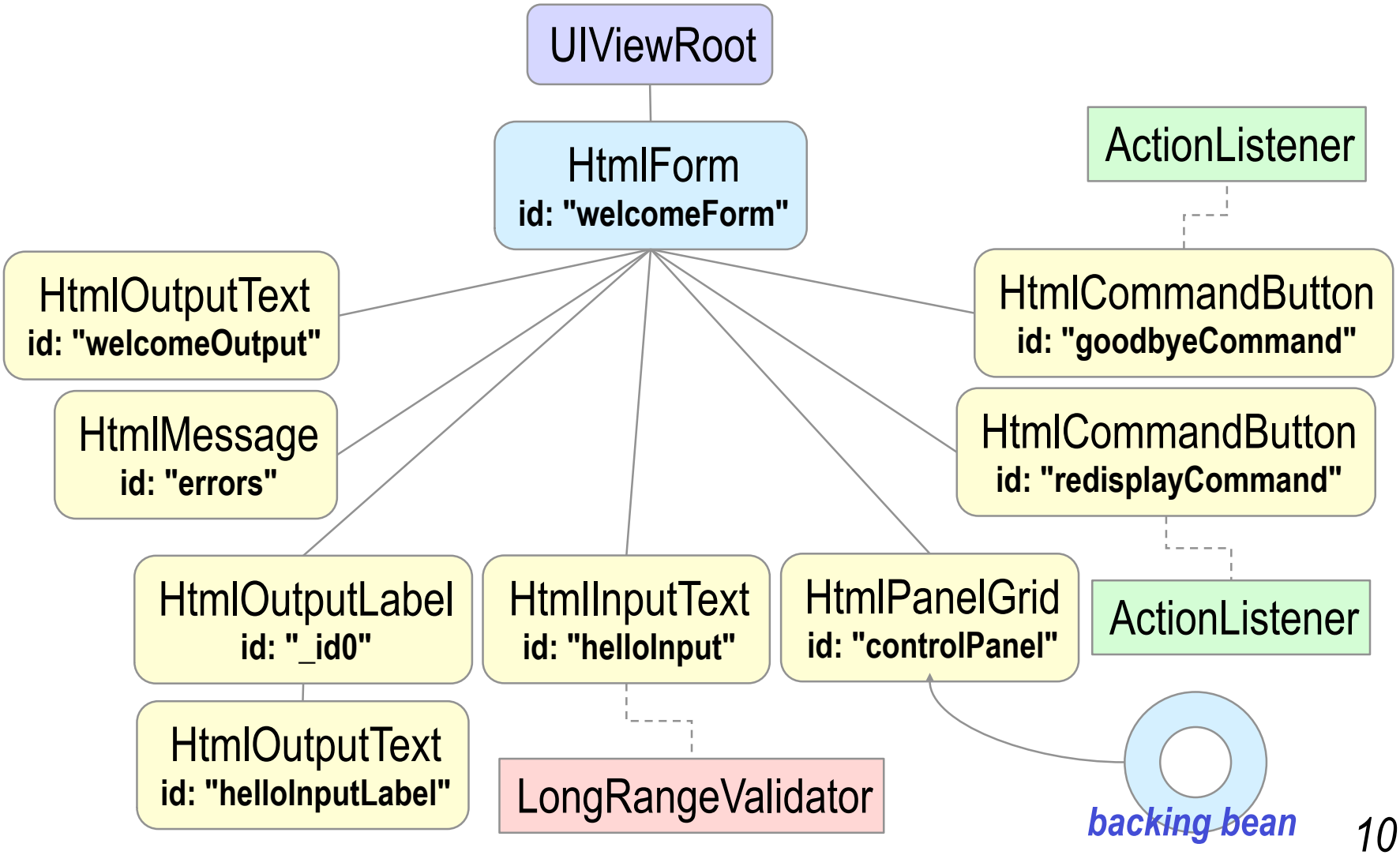
- Exemplo:

Fase 1: *Restore view*

- Dados da requisição:

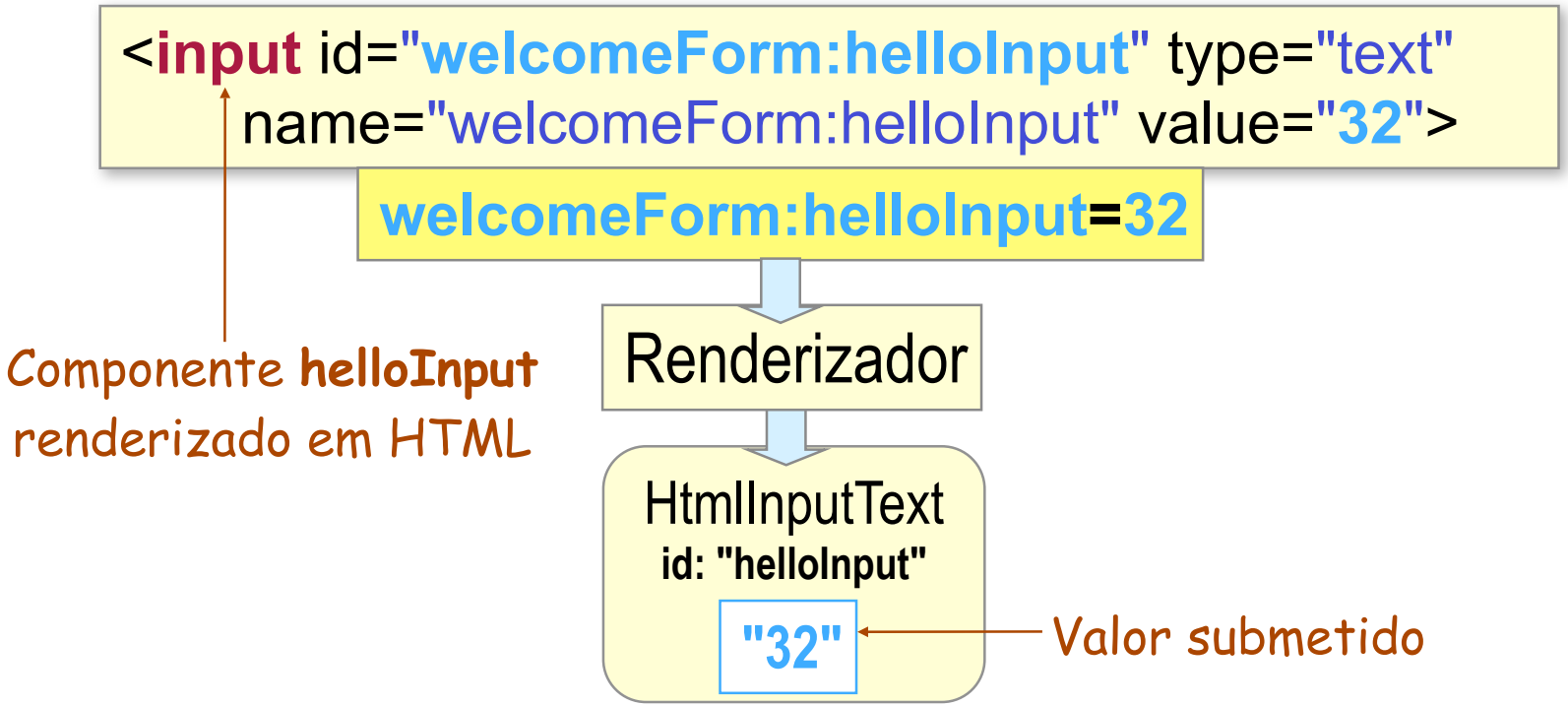
```
POST /jia-hello-world/faces/hello.jsp HTTP/2.1
Host: deadlock:8080
User-Agent: Mozilla/5.0Accept: text/xml,application/xml, text/html;
...
Keep-Alive: 300
Connection: keep-alive
Referer: http://deadlock:8080/jia-hello-world/faces/hello.jsp
Cookie: JSESSIONID=5832475003927 6F39E61932ABDE319DF
Content-Type: application/x-www-form-urlencoded
Content-Length: 92

welcomeForm:helloInput=32&
welcomeForm:redisplayCommand=Redisplay&
welcomeForm=welcomeForm
```

Fase 1: *Restore view*



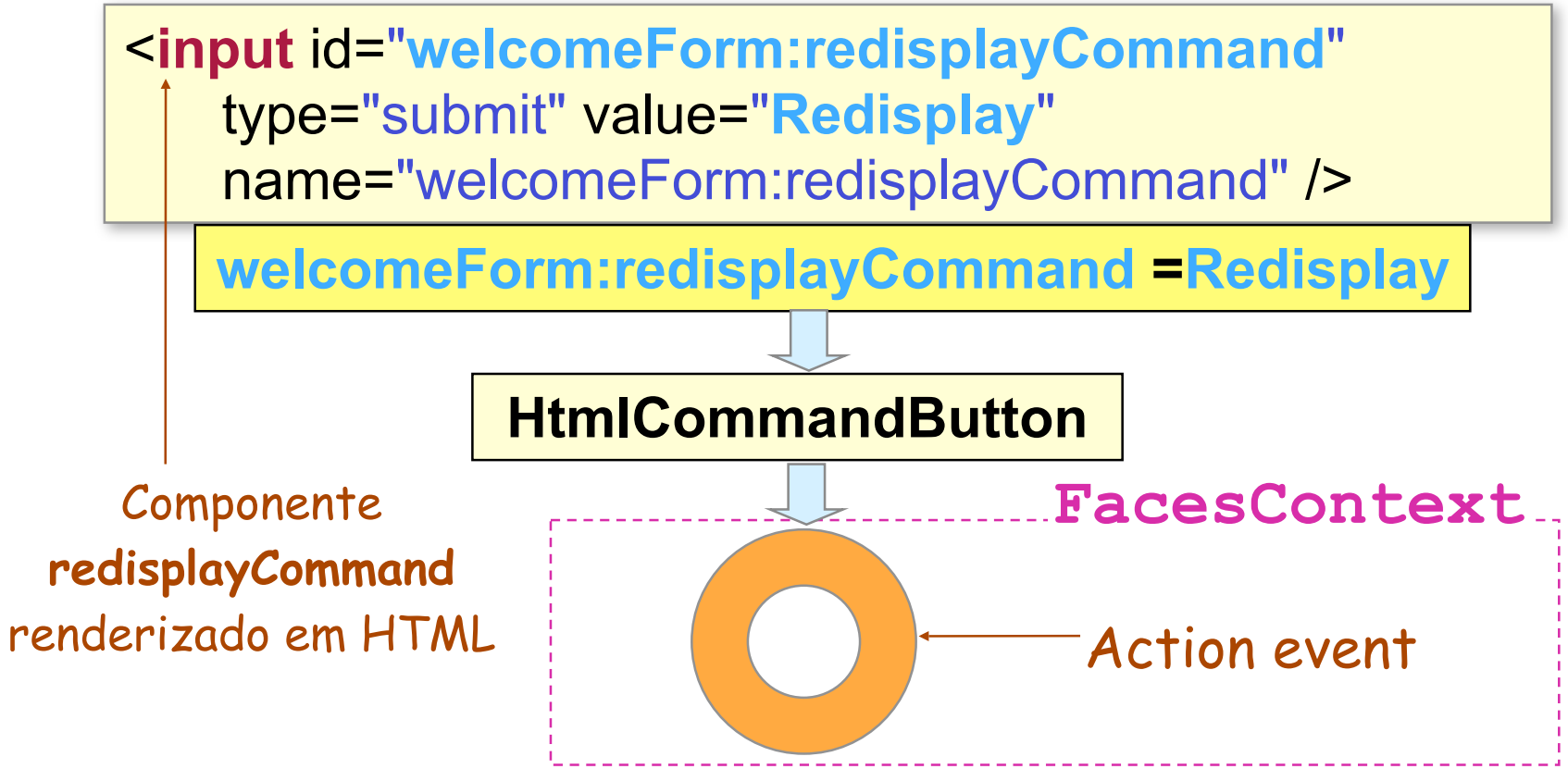
Fase 2: *Apply request values*

- Nesta fase ocorre a ***decodificação***, isto é, atualização das propriedades dos objetos de componentes com os valores fornecidos pelo usuário

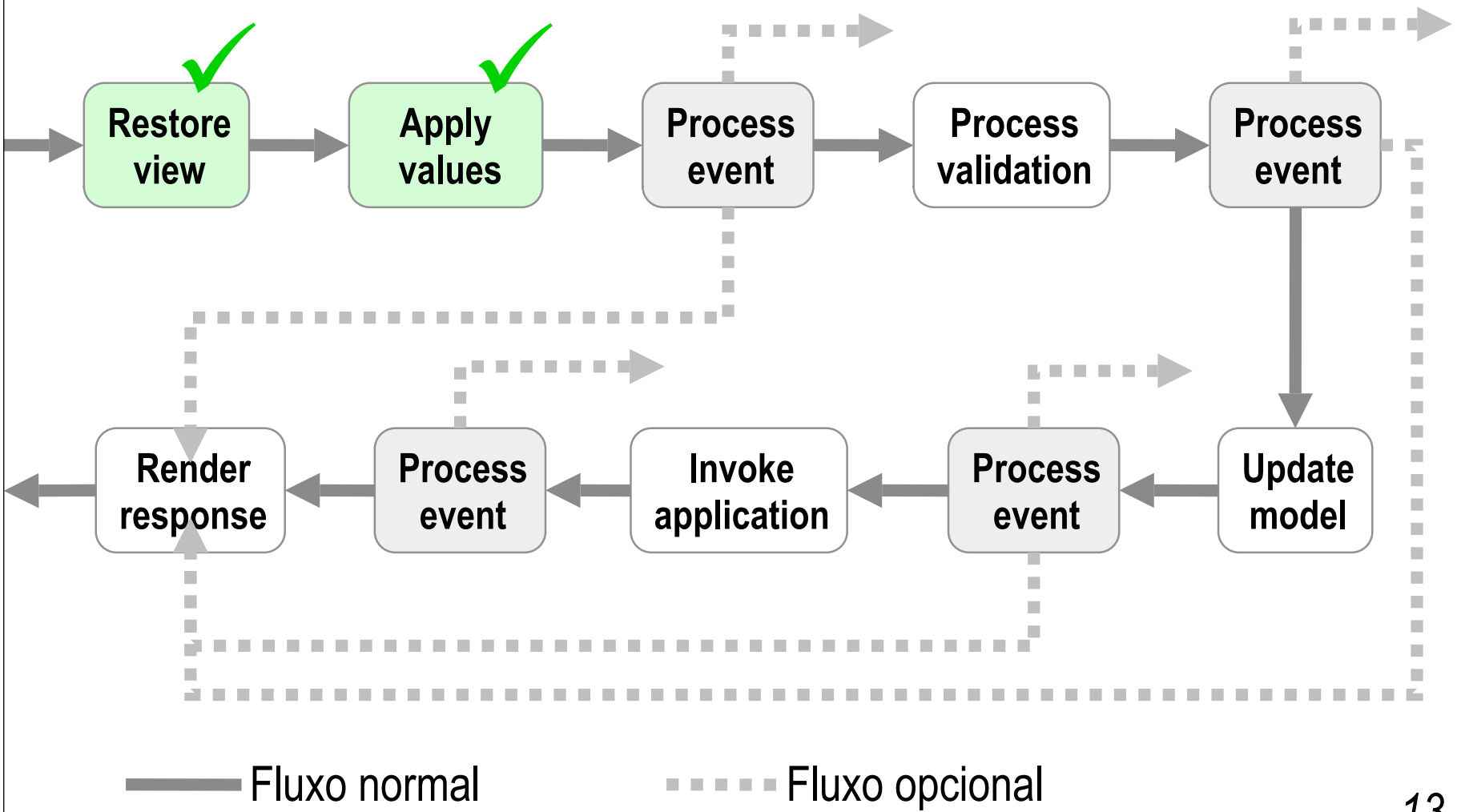


Fase 2: *Apply request values*

- Nesta fase também são criados os eventos de ação, a partir dos componentes de comando (botões, links,...)



Ciclo de vida da requisição



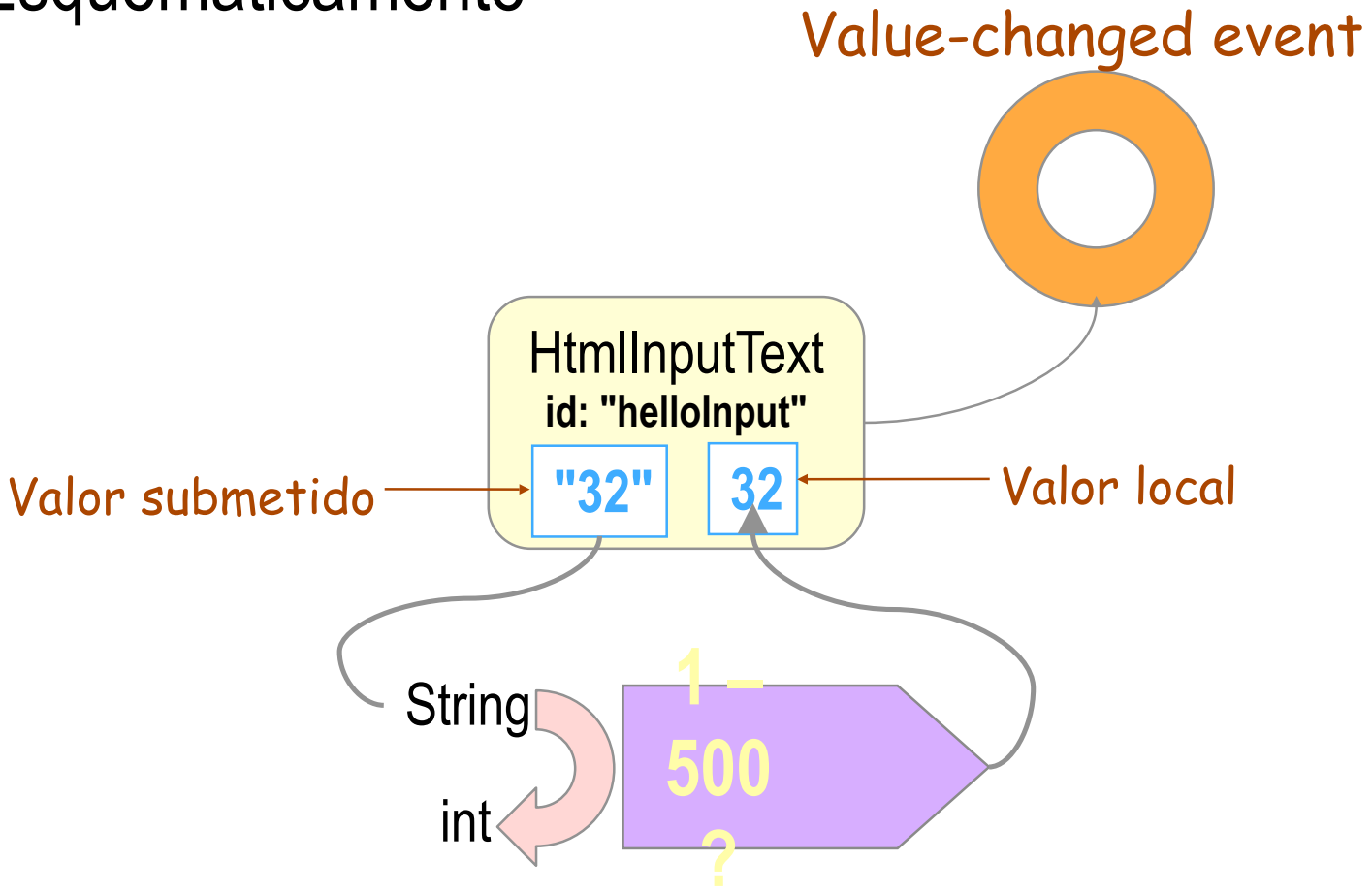
Fase 3: *Process validation*

- A árvore de componentes é percorrida e cada um deles é questionado quanto à ***validade*** valores submetidos, sendo válidos, são ***convertidos*** para o tipo da propriedade do *backing bean* associado

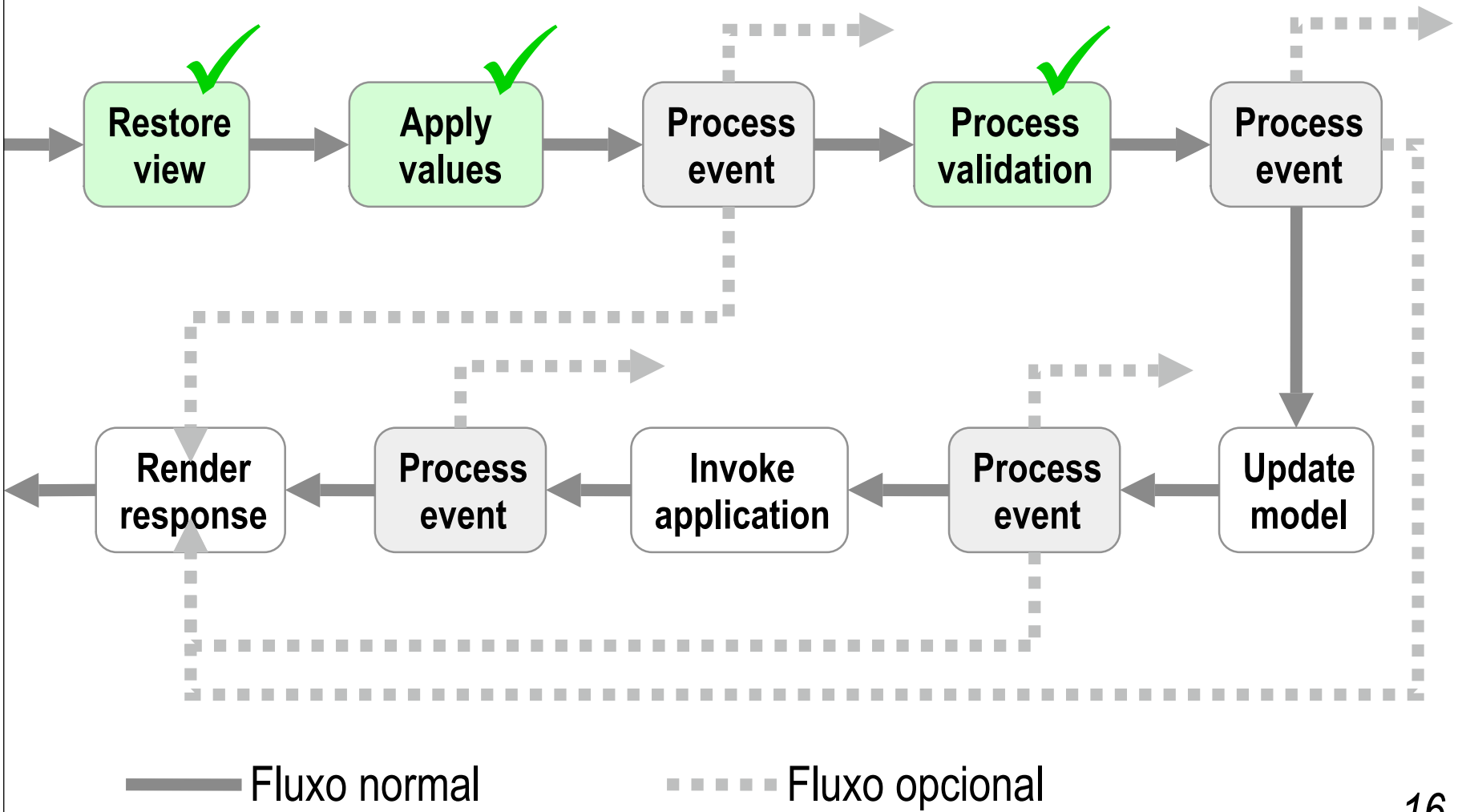
```
<h:inputText id="helloInput"
      value="#{helloBean.numControls}"
      required="true">
      <f:validateLongRange minimum="1"
          maximum="500"/>
</h:inputText>
```

Fase 3: *Process validation*

- Esquematicamente



Ciclo de vida da requisição



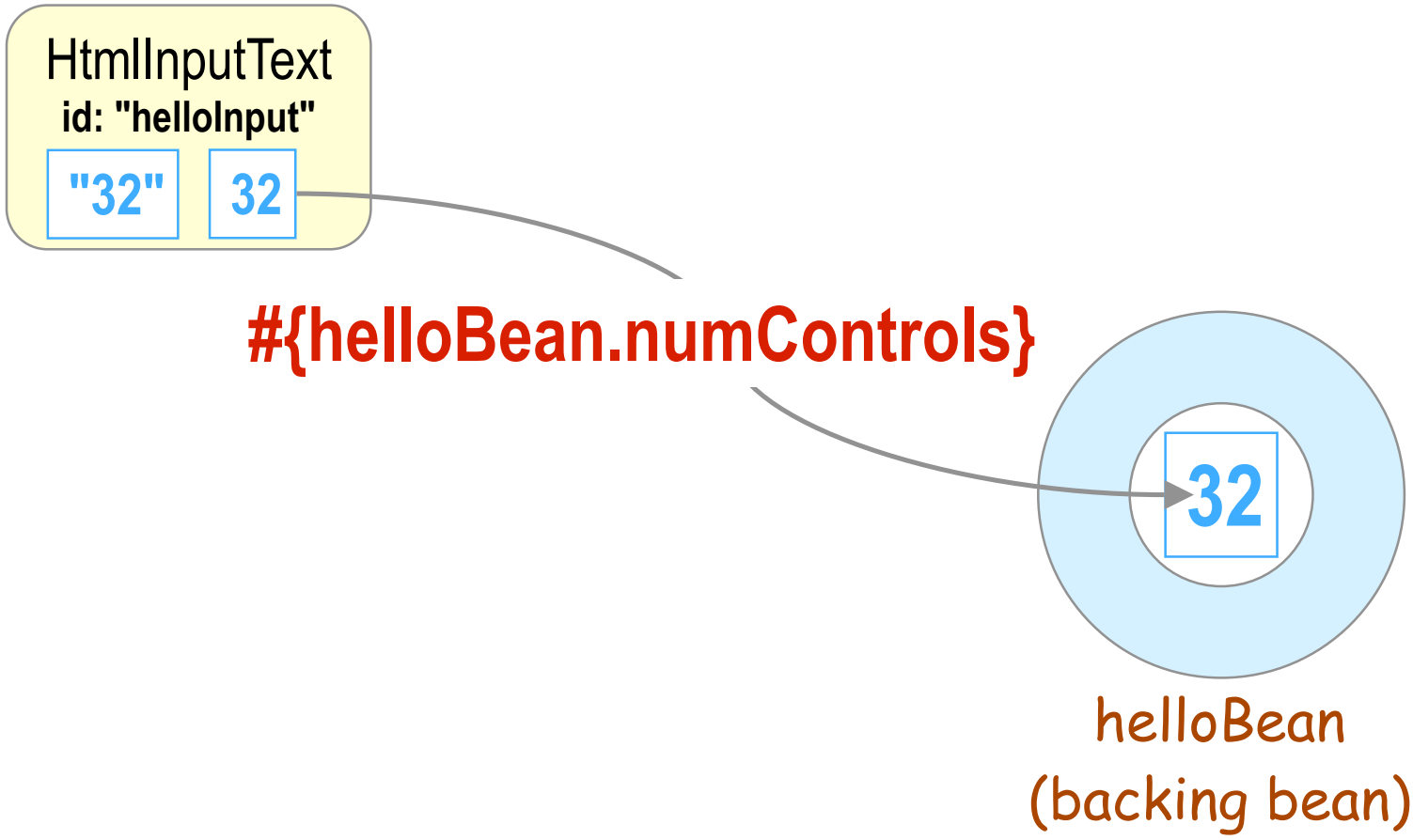
Fase 4: *Update model values*

- As propriedades dos objetos de modelo ou dos backing beans associados ao componente são ***atualizados*** com os novos valores

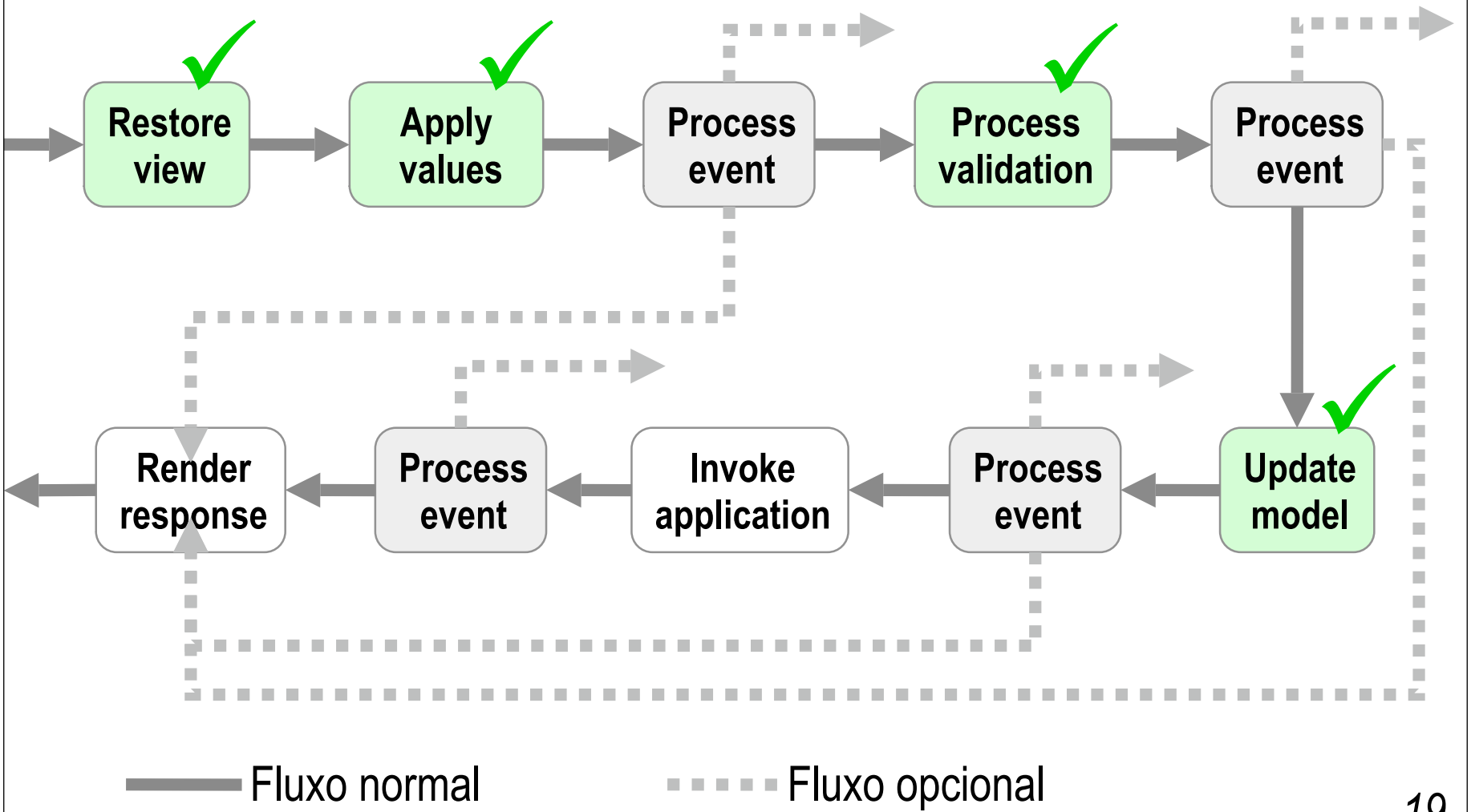
```
<h:inputText id="helloInput"
      value="#{helloBean.numControls}"
      required="true">
...
</h:inputText>
```

Fase 4: *Update model values*

- Esquematicamente



Ciclo de vida da requisição



Fase 5: *Invoke application*

- Os eventos criados nas fases anteriores são despachados para os tratadores registrados

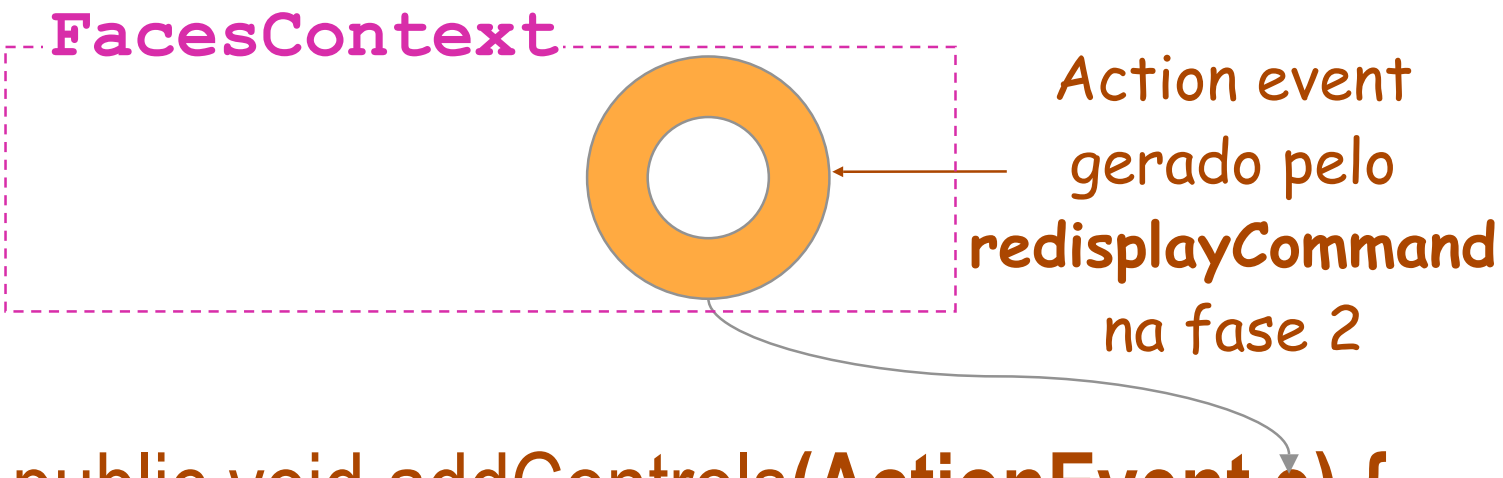
```
<h:commandButton id="redisplayCommand"
      type="submit" value="Redisplay"
      actionListener="#{helloBean.addControls}"/>
```

  - O método **addControls()** do helloBean é executado para tratar o evento de ação gerado pelo componente **redisplayCommand**

É nesta fase que vive a lógica da aplicação!

Fase 5: *Invoke application*

- Esquematicamente:



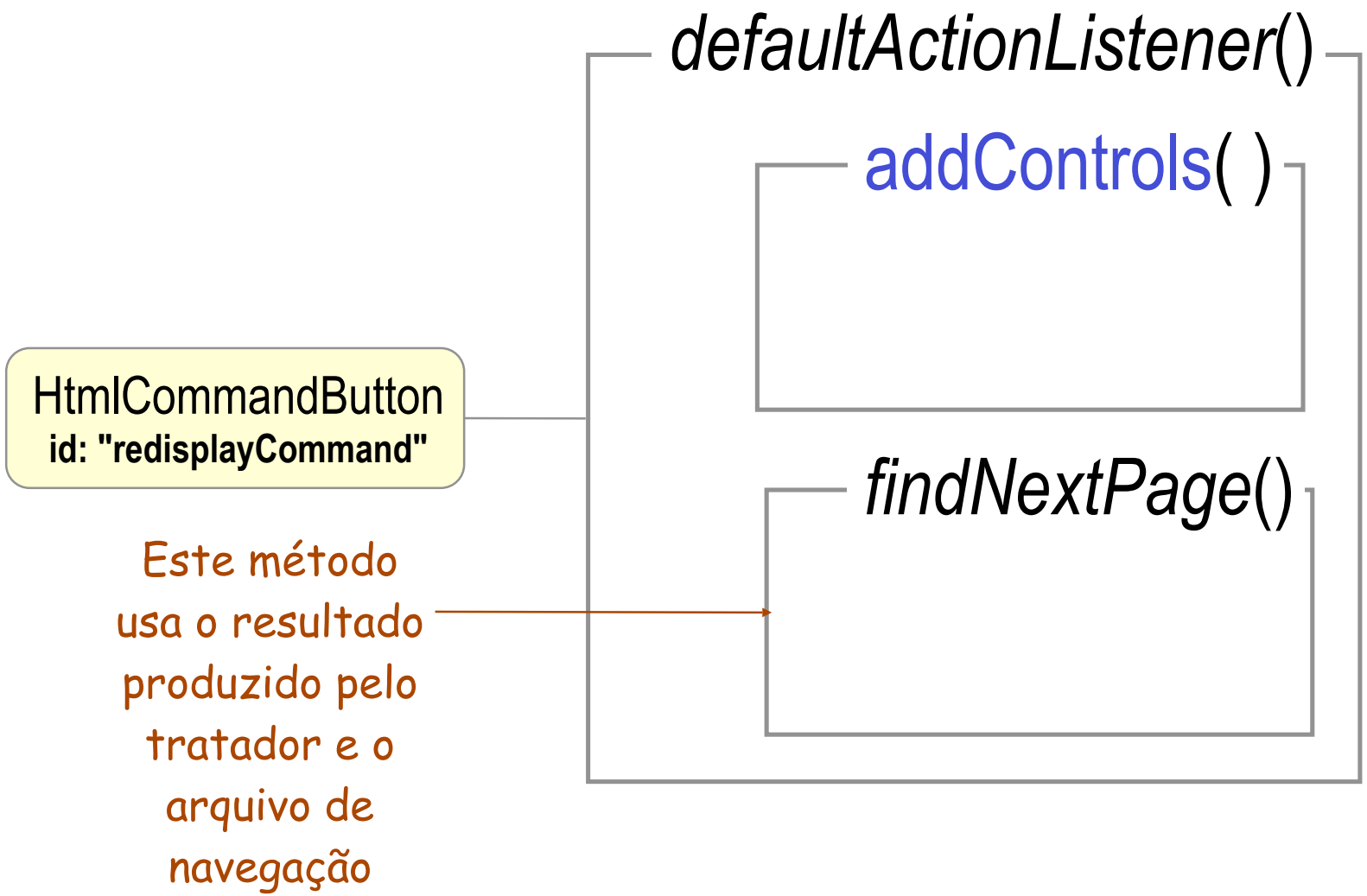

```
public void addControls(ActionEvent e) {
        ...
}
```

Fase 5: *Invoke application*

```
public void addControls(ActionEvent e) {
    Application app =
        FacesContext.getCurrentInstance().getApplication();
    List children = controlPanel.getChildren();
    children.clear();
    for (int count = 0; count < numControls; count++) {
        HtmlOutputText output = (HtmlOutputText) application.
            createComponent(
                HtmlOutputText.COMPONENT_TYPE);
        output.setValue(" " + count + " ");
        output.setStyle("color: blue");
        children.add(output);
    }
}
```

Componente associado ao BB com **binding**

Fase 5: *Invoke application*

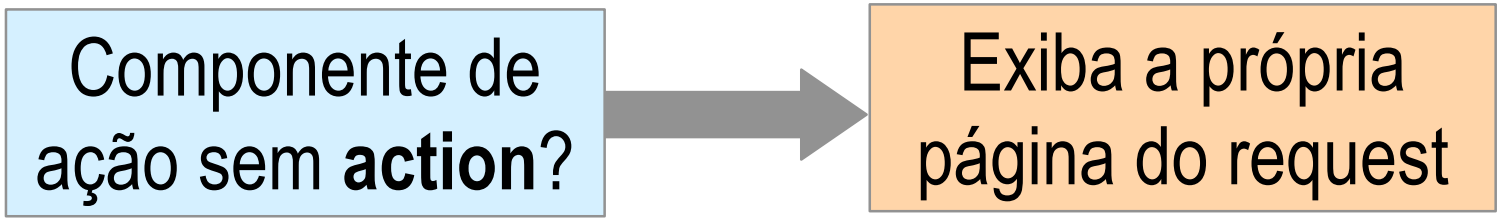


Fase 5: *Invoke application*

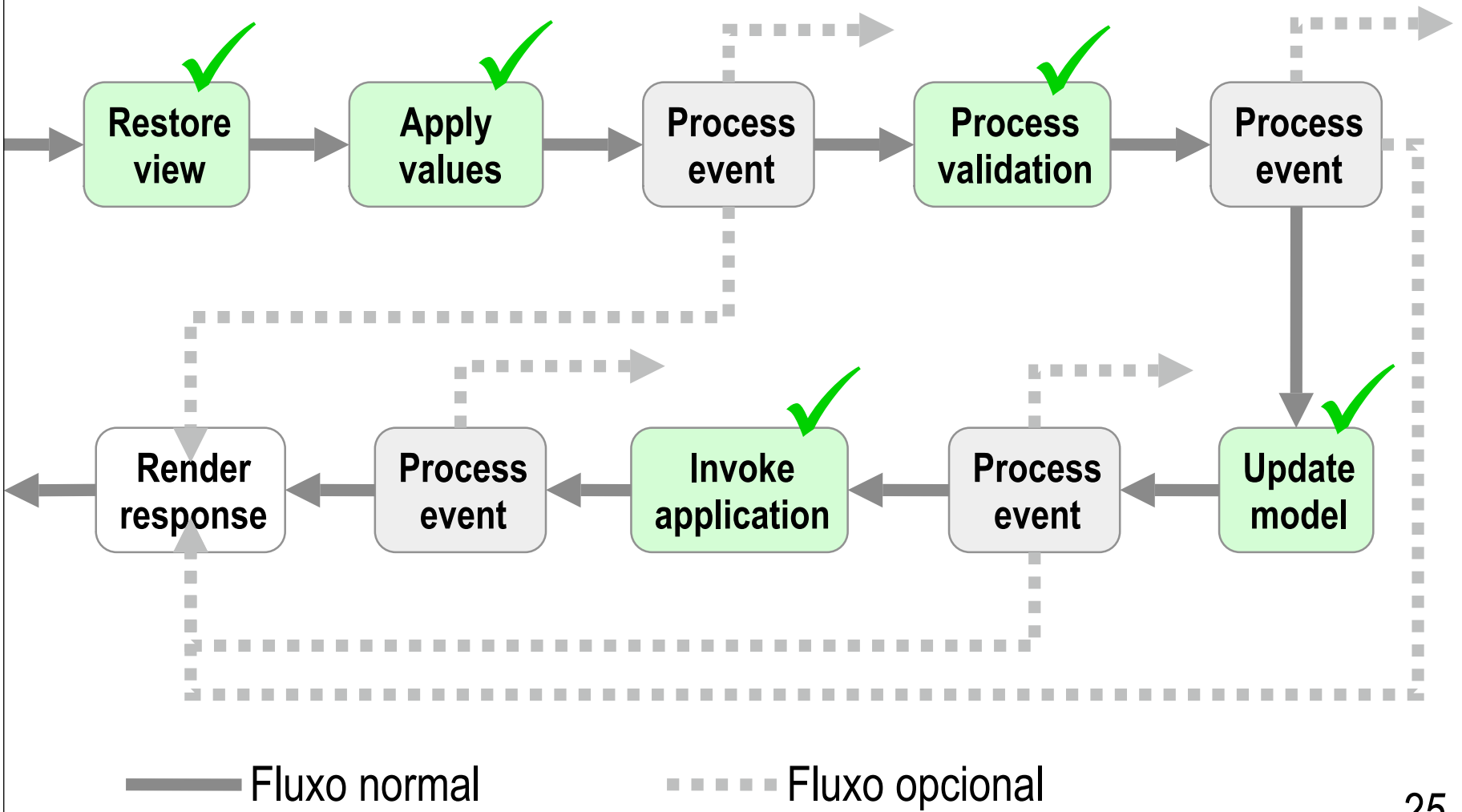
- Mas... **addControls**() não produz resultados (é void e não está registrado em um atributo **action**!)

```
<h:commandButton id="redisplayCommand"
      type="submit" value="Redisplay"
      actionListener="#{helloBean.addControls}"/>
```

  - Nestes casos, a própria página que originou a requisição é reexibida.



Ciclo de vida da requisição



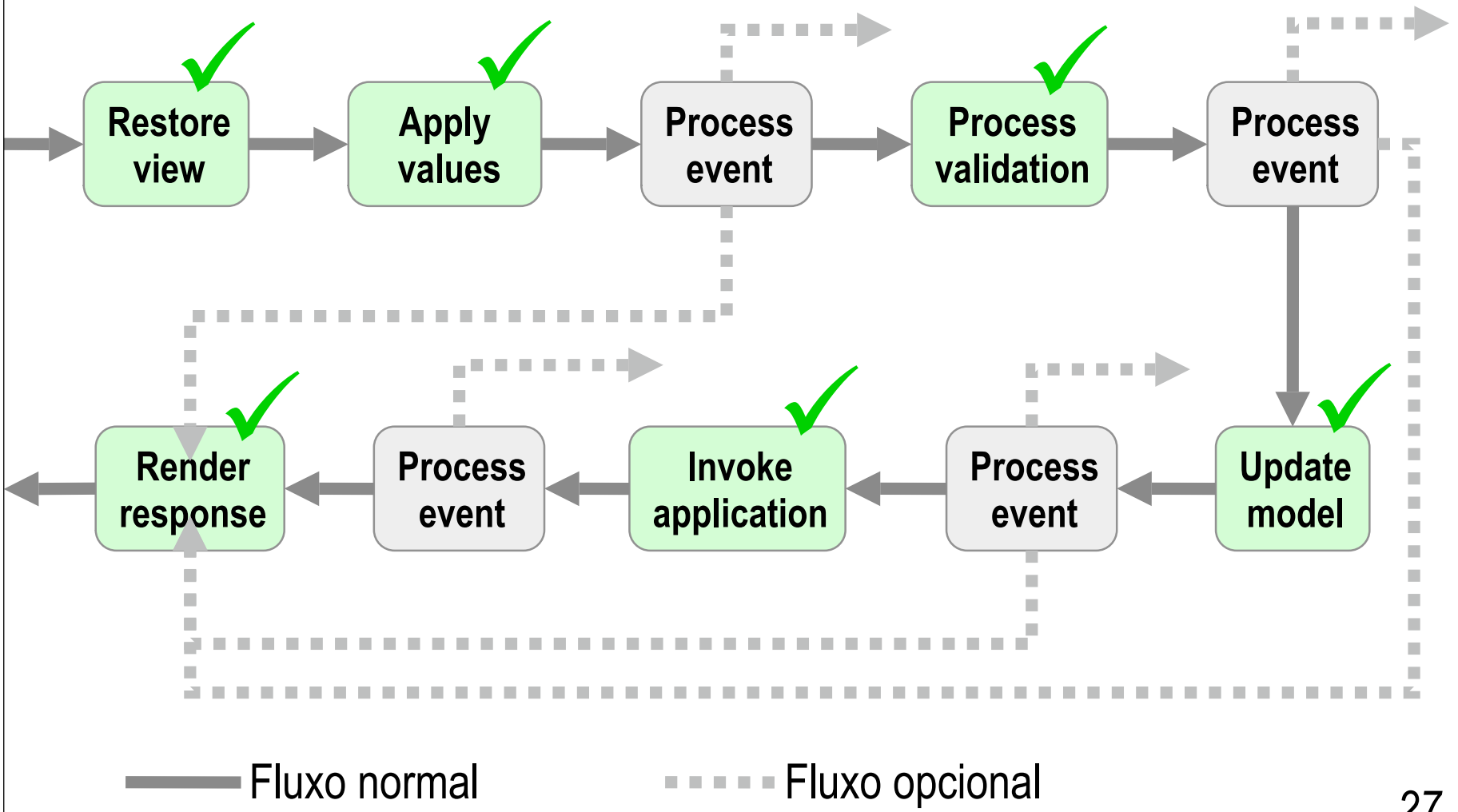
Fase 6: *Render response*

- Uma visão ***é produzida em resposta ao usuário*** e o seu ***estado atual é salvo*** no servidor para ser recuperado na próxima requisição
  - Nesta fase acontece a ***codificação*** template JSP+JSF →HTML
  - Conversores são utilizados novamente

```
<h:inputText id="helloInput"
    value="#{helloBean.numControls}" required="true">
```

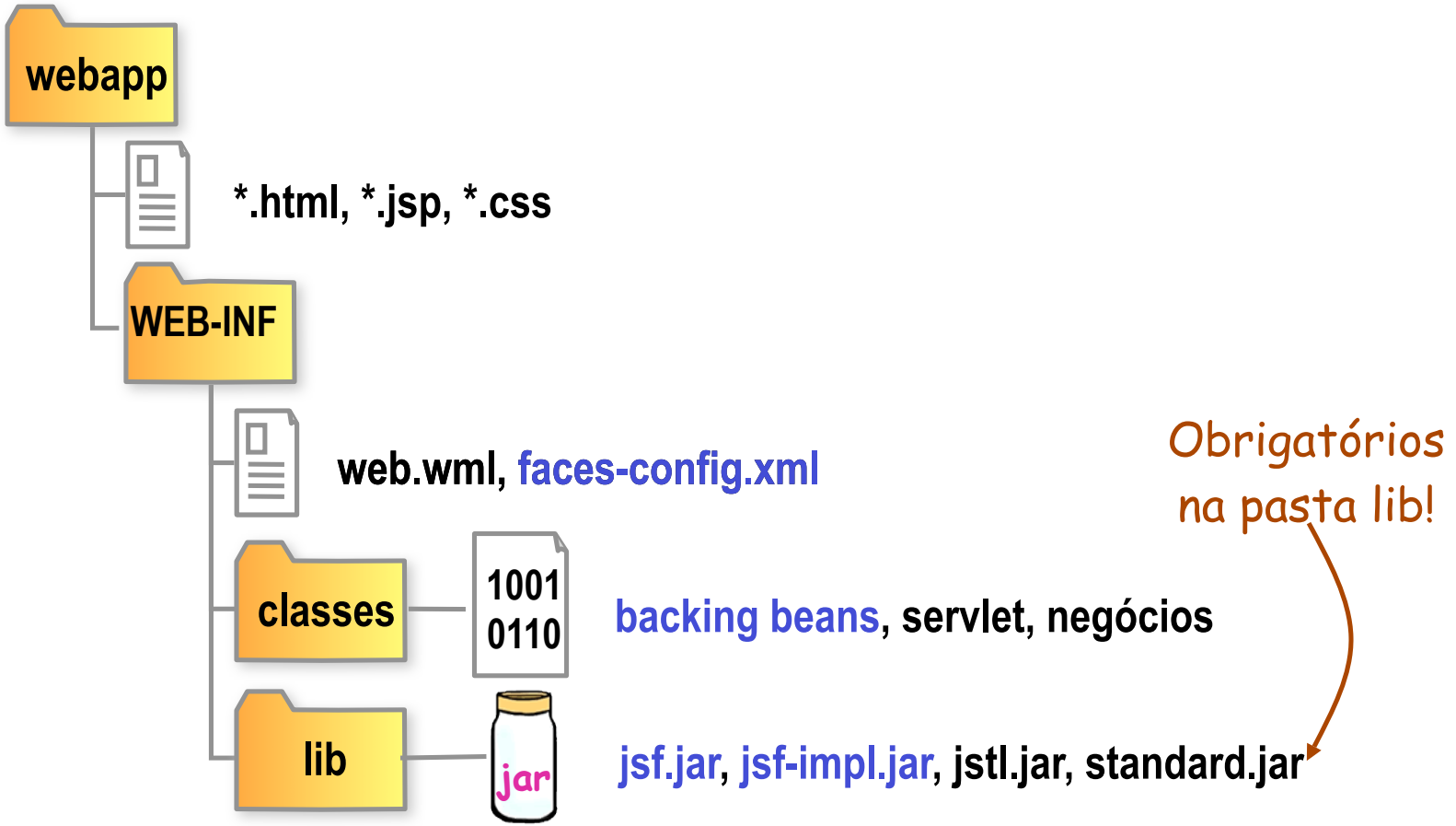
```
<input id="welcomeForm:helloInput" type="text"
    name="welcomeForm:helloInput" value="32">
```

Ciclo de vida da requisição



Configurando o ambiente

- Estrutura da aplicação web com JSF 1.2

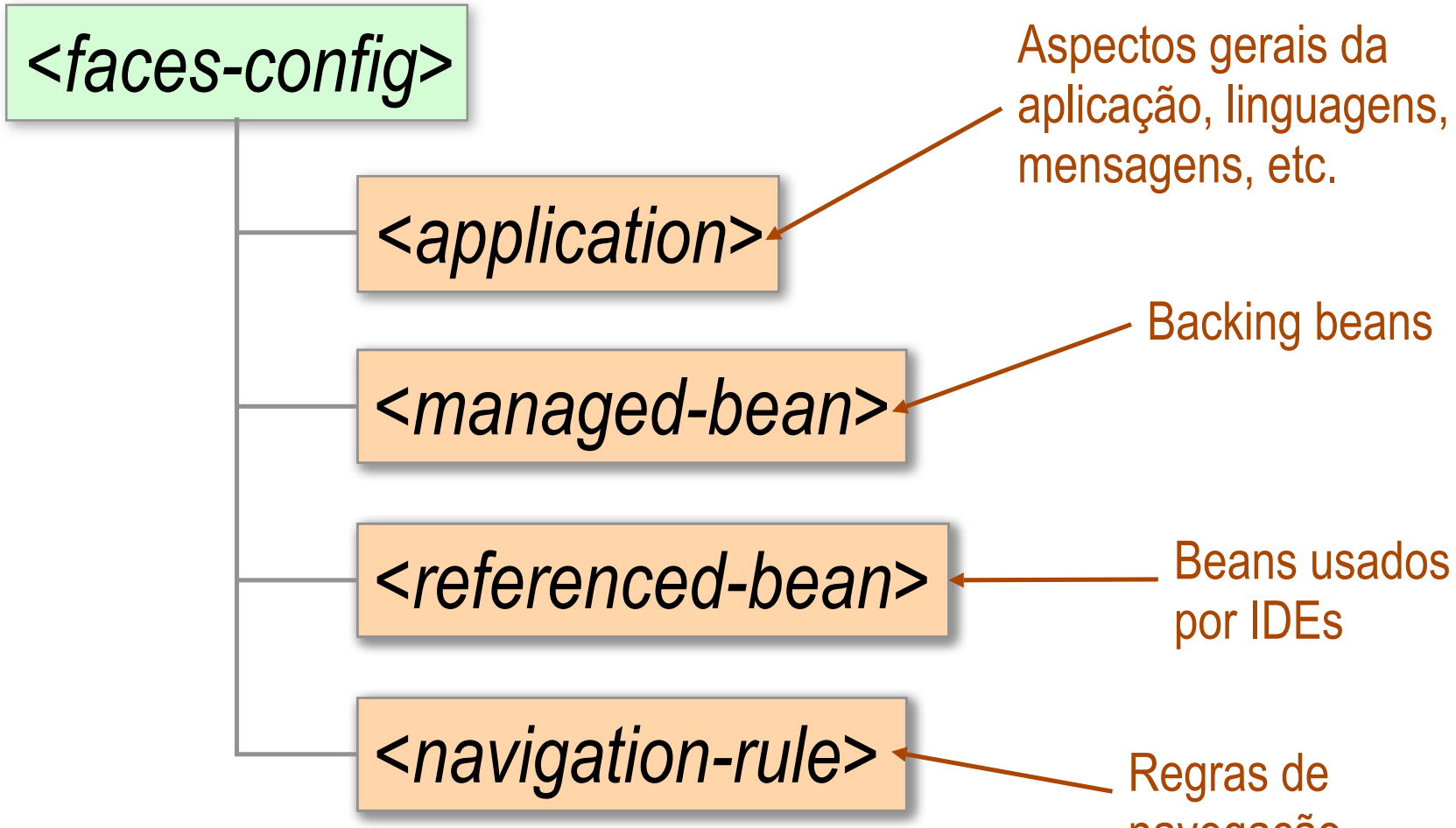


Configuração do faces servlet

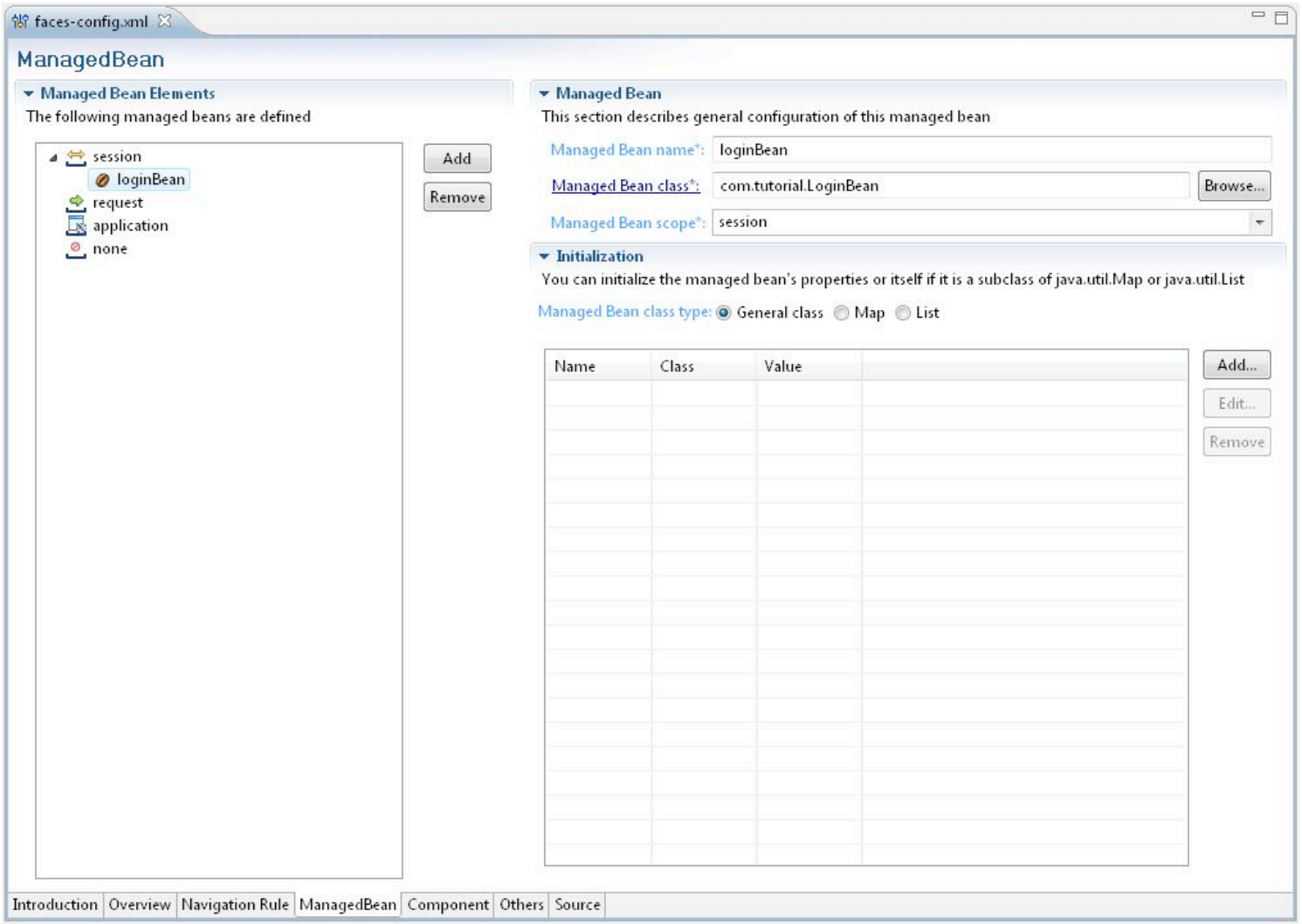
- No web.xml

```
<web-app>
...
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>
            javax.faces.webapp.FacesServlet
        </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
</web-app>
```

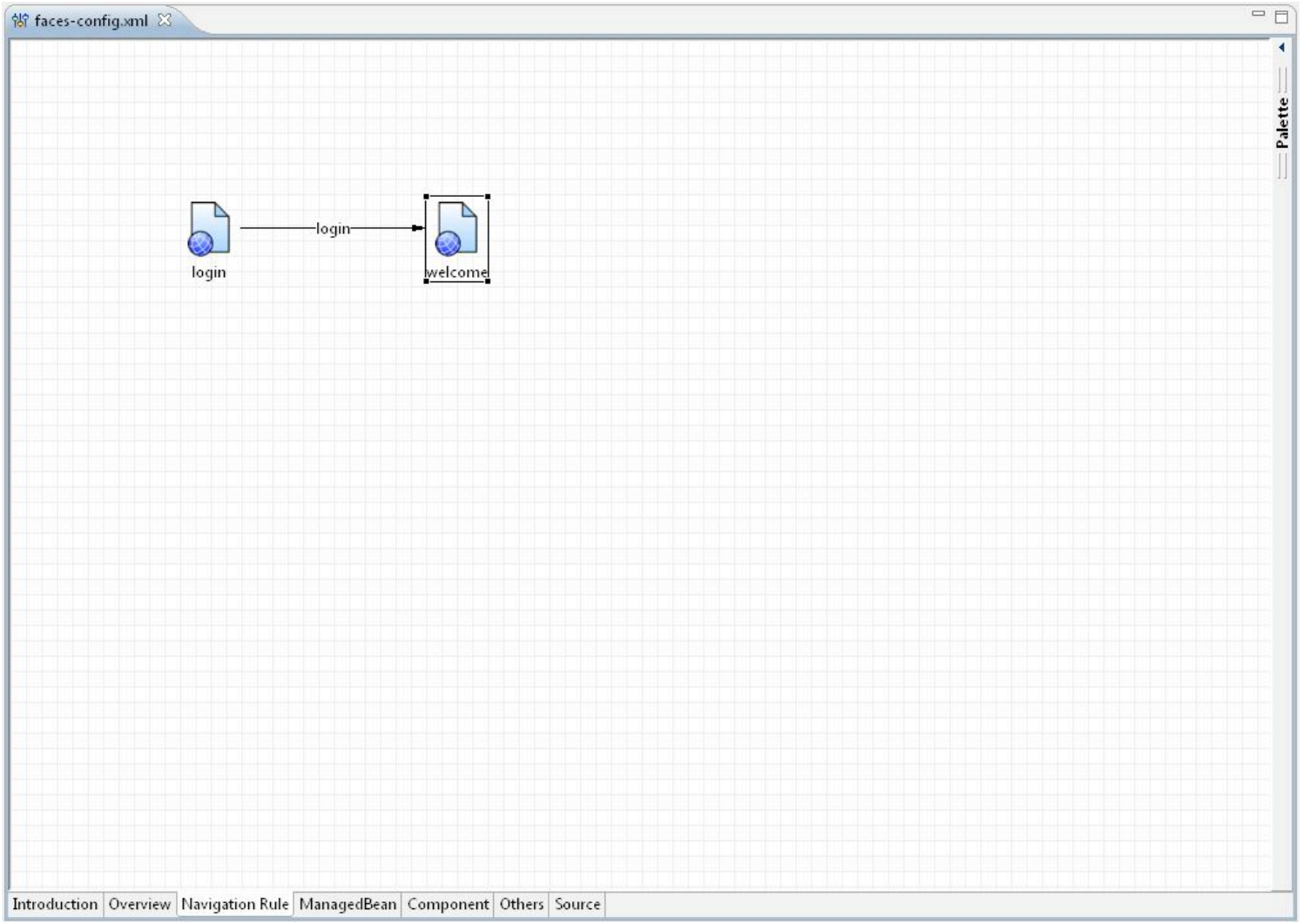
Configuração do faces-config.xml



Configuração do faces-config.xml



Configuração do faces-config.xml



Configuração do faces-config.xml

faces-config.xml

```
    "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
    "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">

<faces-config>
    <managed-bean>
        <managed-bean-name>
        loginBean</managed-bean-name>
        <managed-bean-class>
        com.tutorial.LoginBean</managed-bean-class>
        <managed-bean-scope>
        session</managed-bean-scope>
    </managed-bean>
    <navigation-rule>
        <display-name>
        login</display-name>
        <from-view-id>
        /login.jsp</from-view-id>
        <navigation-case>
            <from-outcome>
            login</from-outcome>
            <to-view-id>
            /welcome.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <validator>
        <display-name>
        Validate Password</display-name>
        <validator-id>
        com.tutorial.ValidatePassword</validator-id>
        <validator-class>
        com.tutorial.ValidatePassword</validator-class>
    </validator>


</faces-config>

```

Introduction | Overview | Navigation Rule | ManagedBean | Component | Others | Source

JSF e JSP

- JSP é a tecnologia *display* usada pelo JSF
  - Componentes JSF são ***tags*** JSP customizadas

```
<h:outputText id="helloBeanOutput"
              value="#{helloBean.numControls}"/>
```

"Custom tag" JSP

  - Principais *tag libs* do JSF:

URI	Nome	Prefixo	Descrição
http://java.sun.core/jsf/core	Core	f	Tags que independem do kit usado para renderização
http://java.sun.core/jsf/html	HTML	h	Tags para o kit de renderização HTML

Uso de includes do JSP

- *Includes* do JSP:
  - Dinâmicos: **<jsp:include>** ou **<c:import>**
  - Estático: **<@include>**
- Para os includes dinâmicos:
  - Páginas incluídas devem ser envolvidas com as tags <f:subview>...</f:subview>
  - Elementos não JSF (texto de conteúdo e tags não JSF) na página incluída devem ser envolvidos por <f:verbatim>...</f:verbatim>

Restrições de uso de JSF em JSP

- Não se pode usar tags JSF dentro de tags que iteram seu próprio corpo, como <c:forEach>
  - Nestes casos use um componente mais adequado, tal como um HtmlDataTable
- Não use <fmt:parseNumber>, prefira um conversor para componentes HtmlInputText

Backing beans: inicialização e criação

- Backing beans são objetos acessados por expressões JSF presentes nos documentos JSP
  - Possuem um nome e um escopo (são atributos!)
  - Podem ser configurados e inicializados no faces-config.xml
- Vantagens da inicialização em arquivo:
  - Ponto central para configuração e inicialização de beans
  - Controle de escopo
  - Mudanças no bean dispensam mudanças no código
  - Inicialização dinâmica de propriedades do bean (JSF-EL)

Backing beans: inicialização e criação

```
<managed-bean>
    <managed-bean-name>
        turmaBean</managed-bean-name>
    <managed-bean-class>
        br.pacote.TurmaBean</managed-bean-class>
    <managed-bean-scope>
        session</managed-bean-scope>
    <managed-bean-property>
        <property-name>
            nome</property-name>
        <value>
            Programação JSF</value>
    </managed-bean-property>
</managed-bean >
```

Backing beans: inicialização e criação

- Acessando o bean no documento JSP:

```
<h:outputLabel for="inputNome">
    <h:outputText value="Nome da turma:"/>
</ h:outputLabel>
<h:inputText id="inputNome"
    value="#{turmaBean.nome}"/>
```