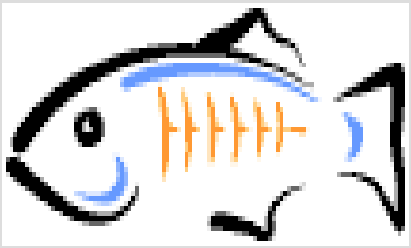
# Java Server Faces





# O que é?

- Tecnologia de componentização da camada View do modelo MVC;
- Especifica uma série de componentes visuais;
- Importante: É uma especificação! (JSR-127 e JSR-252);
- Semelhante ao ASP.NET;
- Rich-clients para Web.

# Principais motivações

- HTTP é stateless, “tudo” se perde entre uma requisição e outra;
- Front ends web complexos possuem muito HTML e javascript, complicado de dar manutenção;
- Não se consegue “ouvir” os eventos da página web no código do servidor;
- Padronização e referência.

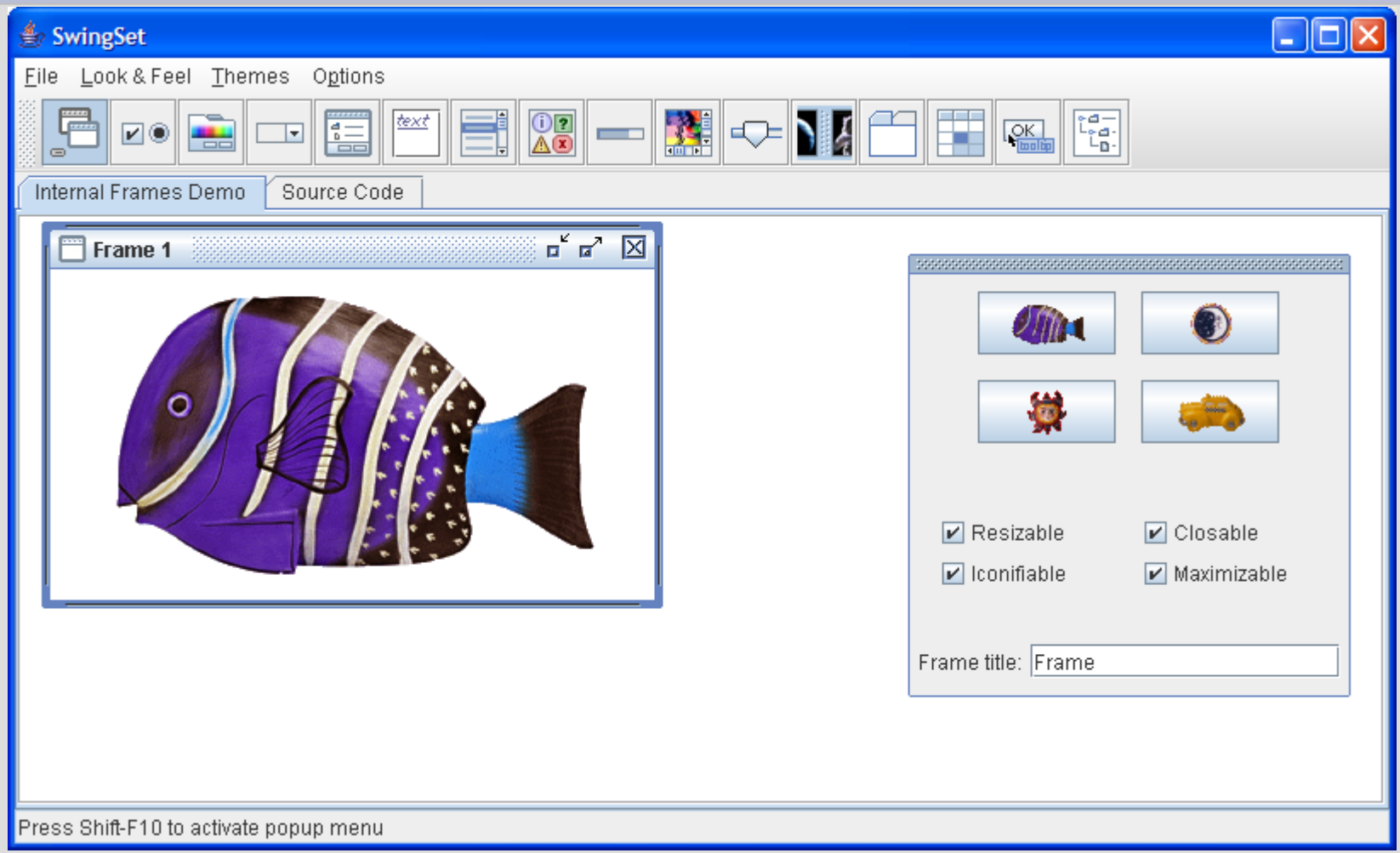
# Como JSF Ajuda?

- Salvando o estado da página, ou seja, o que era stateless agora é stateful;
- Variedade de componentes visuais (Grids, Calendários, caixas de texto, etc.);
- Tratamento dos eventos de tela no lado servidor;
- Não limitado somente a WEB.

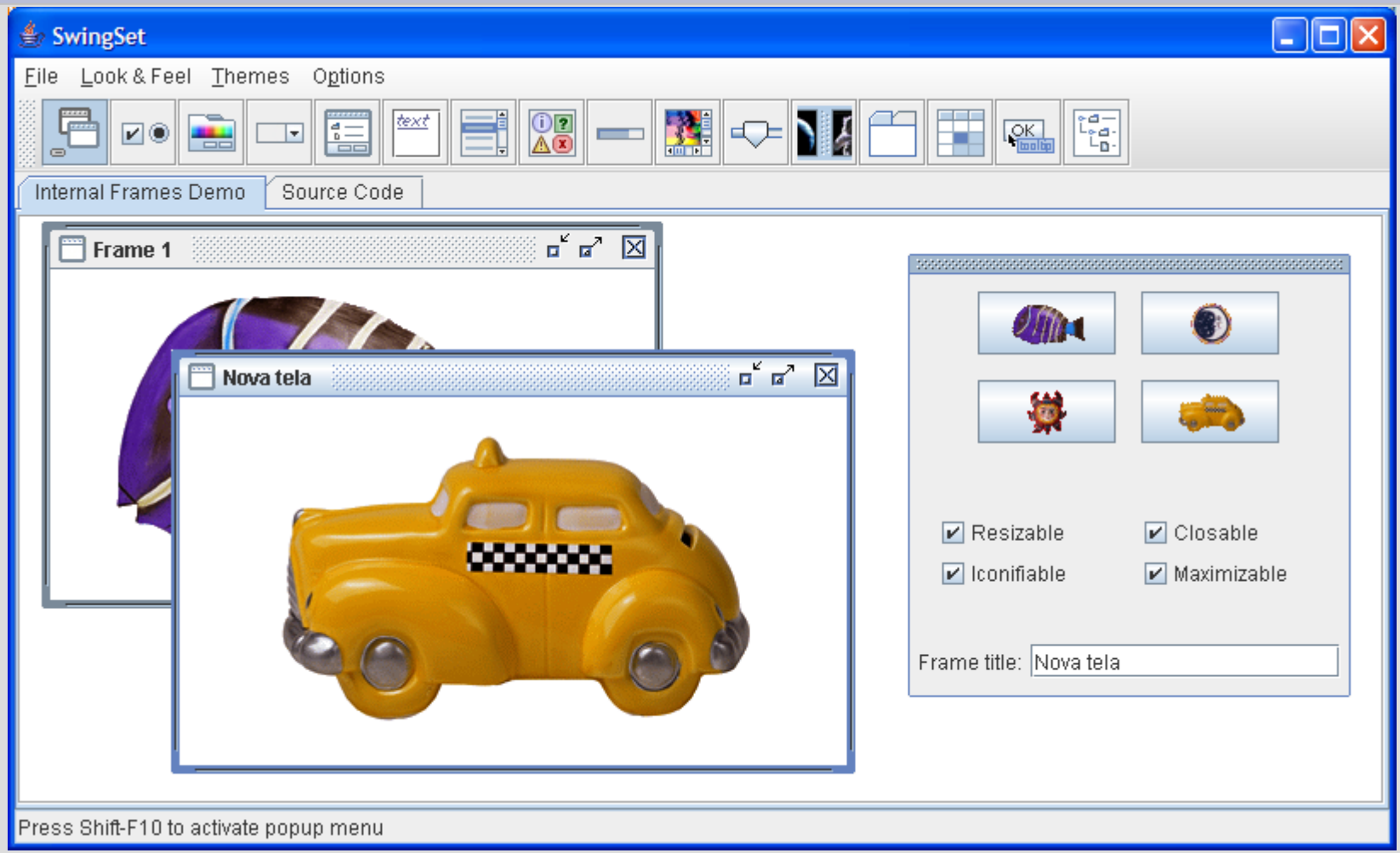
# Mais motivações..

- Necessidade de clientes cada vez mais interativos por parte dos usuários;
- Necessidade de clientes cada vez mais interativos por parte dos desenvolvedores;
- Necessidade de aplicações com front end web cada vez mais parecidas com aplicações desktop;
- Necessidade de um controle maior por parte dos desenvolvedores.

# Problema Clássico



# Problema Clássico



# Passos para WEB

- Lembrar do Get/Post!
- Puxar de algum lugar que havia um peixe na tela;
- Puxar as informações do nome da nova tela;
- **Desenhar o taxi com o nome escolhido;**
- Colocar o campo Frame title para o nome da tela criada.
- *Grande parte da programação Web envolve restaurar o estado de telas anteriores!*

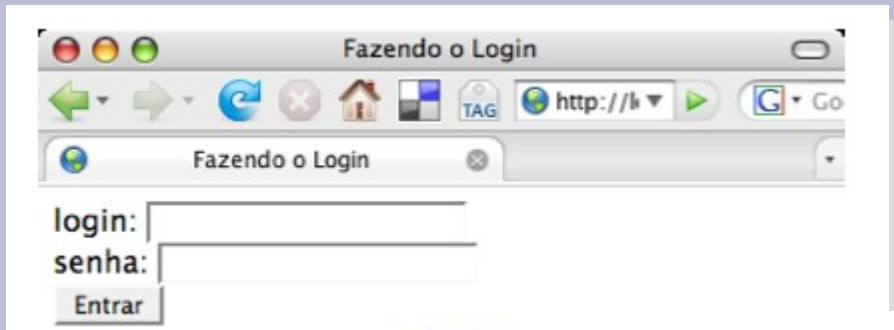
# Implementações de JSF

- Versão atual: 1.2
- Principais implementações:
  - MyFaces (Apache)
  - RI (Sun)
  - ICEFaces (ICESoft)*
  - RichFaces (Jboss, ex Ajax4jsf da Exadel)*
- Componentes opcionais
  - Tomahawk (Apache)

# Um arquivo JSF

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8" %>

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Fazendo o Login</title>
</head>
<body>
<f:view>
  <h:form>
    login: <h:inputText value="#{autenticador.login}"/><br/>
    senha: <h:inputSecret value="#{autenticador.senha}"/><br/>
    <h:commandButton value="Entrar"/>
  </h:form>
</f:view>
</body>
</html>
```

# Resultado




```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Fazendo o Login</title>
</head>
<body>

<form id="j_id_jsp_617040448_1" name="j_id_jsp_617040448_1" method="post"
    action="/projetojsf/olajsf.jsf" enctype="application/x-www-form-urlencoded">
    <input type="hidden" name="j_id_jsp_617040448_1" value="j_id_jsp_617040448_1" />

    login: <input type="text" name="j_id_jsp_617040448_1:j_id_jsp_617040448_2" /><br/>
    senha: <input type="password" name="j_id_jsp_617040448_1:j_id_jsp_617040448_3" /><br/>
    <input type="submit" name="j_id_jsp_617040448_1:j_id_jsp_617040448_4" value="Entrar" />
    <input type="hidden" name="javax.faces.ViewState" id="javax.faces.ViewState"
                                                          value="_id1" />

</form>
</body>
</html>
```

# web.xml

```
<!-- Configura a servlet do JSF. -->
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

# Managed Beans

- O que são JavaBeans?
- Quando se trata de JSF, podem assumir vários papéis diferentes:
  - Componentes de UI;
  - Agir como “Backing-Beans”;
  - TOs;
  - Serviços;
- Destaque para Backing-Beans
  - Contém algum ou todos os objetos de um formulário web.

# Exemplo

```
package br.com.caelum.fj26;

public class LoginHandler {

    private String login;

    private String senha;

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        System.out.println("mudando o login para: " + login);
        this.login = login;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

}
```

# faces-config.xml

```
<faces-config ...>
  ...
  <managed-bean>
    <managed-bean-name>autenticador</managed-bean-name>
    <managed-bean-class>br.com.caelum.fj26.LoginHandler</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
  ...
</faces-config>
```

# Restaurando a árvore de componentes

- Os componentes da tela (View State) são mantidos em uma árvore;
- A árvore pode ser salva no servidor, tipicamente na sessão do usuário, ou no cliente.

```
<!-- Configura onde salvar o estado dos componentes. O padrao é server -->
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>client</param-value>
</context-param>
```

# Árvore“Stringuificada”

```
<input type="hidden"
  name="javax.faces.ViewState" id="javax.faces.ViewState"
  value="H4sIAAAAAAAAAL1WXWgcRRyf23wnWvPRxg+IJKY1uZrs3TXXpiTRmE9z5S4nuRhtK14
  nd5PcHnu74+xsdptgSQtasFgrKqhELCr4Ul/0Rd+UPgiFCgb0wQcpIohgFUSo+qDO7N7t7l32b
  EVwH/bmbv7zm//8f7/ff+7SdVCnETCiklUR5qEZzWsrIsRYljKQSqoiTrAfUxRSlIAKXEVk7yJ
  BKEWJTnWC5tUs0vPiz69dXtn3xm0AmHgdgEAW9JahZdQCVhWkUAtrSULGgqpSsDuflrLpvIbTh
  yLD4Wg4Gj2cDgP7CYDePFyDprgCM0jzI0RoQRbn2GtWJQVfiEgRogb03wQipmCdLiLTP5UDoPT
  svxWcFMoQ5I805CAN3ARpSi0UoJKd1ClVFV+saAnK1AnoOB7ngKIMlVUxuZxHGTp64fMn3mrVg
  rLA2WBxgv400AWEslGjO2LUt1kYOpVYDlDLJSCua/j608udJ76oAcIsaJZVmJ2FGaqSGGii0YK
  0nCpnTTz+sJVIvdHIgfgGFOz1HtBAy0xI4tRjCwsz84vppdjM4+mFZHKRb95EQbdfbDLxaHKeR
  8emUyy3Dje3CULgybikUfP0dtfrn8E3a0AgBmo1aR1ZBw0Ytezd7U9BxDRxFcXxbHY5ZbBK0am
  qMoLK1W6y+eXWHz8JIHAM1K1BWWc7BTS+4A6AGV7z3GIinp6cSMWmKGgJqTLkcmf4DK7VzTyuZ
  qCMTv3WdmIr/PuPAqiNgcYcq3SGmScOGjKqrlBykoJ2i8OQTyHEDCYpq6Nx0Mi/6sx4fNs9LHw
  NEgkq1Ppq4r/YQ4EwucBemFIAyk8SUyhinm3/9uK7N06fPSzwkhVPUkrRipvXC8uIPHfp1a6WV
  66dK2mnBmPsSKWOD9qsF3aHznTAR2edHp3xUcMOyQT4z83WZIs1UeSwjjMVrmLM4nSkituK0wf
  8p6NMCXy7251Xq3UQapejpEfeXtjpGck9nmYYMgcNwxhcYZODOpGRwinM4srHOXaNT1F6nNEun
  7hG7FlS4fCiJt7fXvr+h66NR0osBWiRUpcLJoL7/VuN1cnnmH8RScE1RI5e+fDBl7euJgQgxEF
  TRoaaNg8LJbE1aywma62hoNOWp6SGUohJUJbW4bKMRk3M5d5ndXwMMzkk5qGGERGRLB7R8BLPb
  MbErG1orILw6PVvwrs/GuOXBQF32UmyyIowo3nz+LVP/nxJsMI6nDA34p1nn0/9cmx7zKqCMQD
  2e+zXHzk4EIkEu/t6N6BO2bEZfVmViLK6KinP9LF0ezzp7tw9VsByT+TIhbelMwJPlFdj1AiCP
  b54Xh/ZTRhjk10qQb8+NOInaI6/r6oU+Q3FROV6zuNBXOHHVk8ALlvEWOKf92wyfbjzzuMG+oQ
  5wgI+eu69dT3z0VDJ9Qc9cuWjYTvBkR1cHvLlUkNKDjIu+ZqHfLix5/8zNOMcv8/LTbuXG/vW/
  x/Z+Uee+GDah6J7nVG9hxj8L8oQ5cuCVctg/2EpHmKY94P+av0ggWhOzbpeG2/ZgF/dSV6wG8L
  djtMr4z4IXTn33Y3oeaHoxzHjARAs08nwQNTX8pBYlr+v3PKV+NzzZy+Of/zUe78mS55/0t/zD
  HCnrhxxA64wlyYK6mfY9Q5JBd+4GsP2rV0v6csFybrlp/8Gq6jhaaMLAAA=" />
```

# Tratando Eventos

- Eventos podem ser “ouvidos” tanto nos próprios beans como em implementações de listeners;
- Eventos podem ser, tanto ActionEvents como ValueChangeEvents;
- Eventos não controlam fluxos de navegação, eles ajudam as action a fazê-lo;

# Exemplo


```
<h:selectOneMenu value="#{form.country}" onchange="submit()"
   valueChangeListener="#{form.countryChanged}">
   <f:selectItems value="#{form.countryNames}"/>
</h:selectOneMenu>
```

```
private static String US = "United States";
...
public void countryChanged(ValueChangeEvent event) {
   FacesContext context = FacesContext.getCurrentInstance();

   if (US.equals((String) event.getNewValue()))
      context.getViewRoot().setLocale(Locale.US);
   else
      context.getViewRoot().setLocale(Locale.CANADA);
   }
}
```

# Exemplo

```
<h:selectOneMenu value="#{form.country}" onchange="submit()">
   <f:valueChangeListener type="com.corejsf.CountryListener"/>
   <f:selectItems value="#{form.countryNames}"/>
</h:selectOneMenu>
```

```
public class CountryListener implements ValueChangeListener {
   private static final String US = "United States";

   public void processValueChange(ValueChangeEvent event) {
      FacesContext context = FacesContext.getCurrentInstance();

      if (US.equals((String) event.getNewValue()))
         context.getViewRoot().setLocale(Locale.US);
      else
         context.getViewRoot().setLocale(Locale.CANADA);
   }
}
```

# Navegação

- Toda navegação é controlada por *actions* e *outcomes*;
- Actions podem ou não estar relacionadas a métodos.
- Ex:

```
<f:view>
  <h:form>
    login: <h:inputText value="#{autenticador.login}"/><br/>
    senha: <h:inputSecret value="#{autenticador.senha}"/><br/>
    <h:commandButton value="Entrar" action="sucesso"/>
  </h:form>
</f:view>
```

```
<h:commandButton value="Entrar" action="#{autenticador.logar}"/>
```

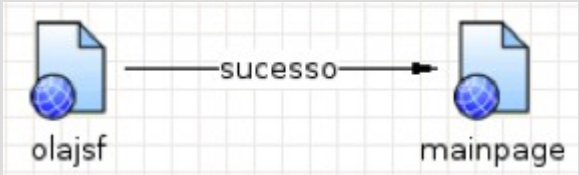
# Navegação cont..

```java
public class LoginHandler {

  // ...

  public String logar() {
    System.out.println("fazendo login... ");
    return "sucesso";
  }
}
```

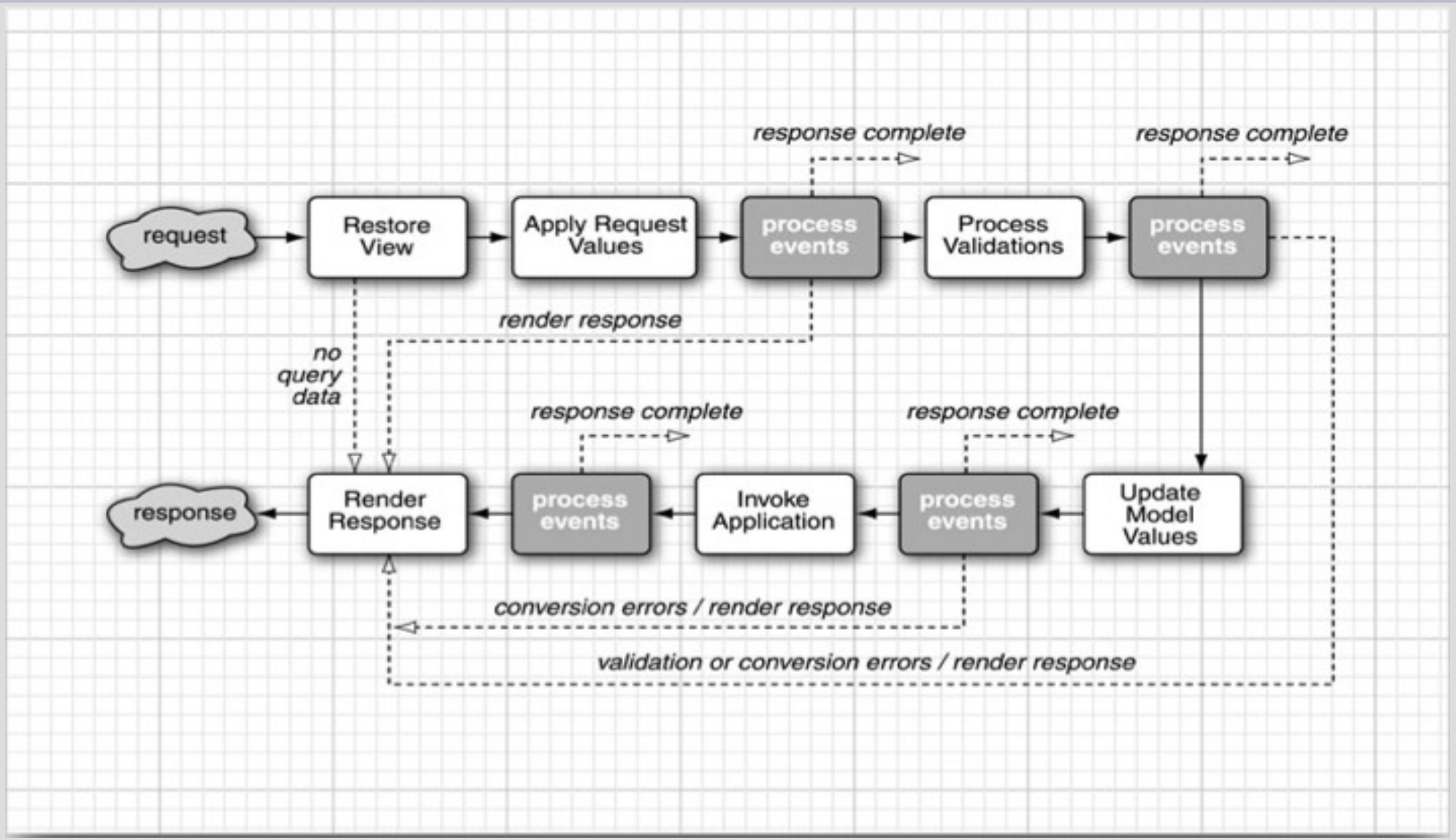
```xml
<navigation-rule>
  <from-view-id>/olajsf.jsp</from-view-id>
  <navigation-case>
    <from-outcome>sucesso</from-outcome>
    <to-view-id>/mainpage.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

# Por que disso?

- Abstrair detalhes do fluxo de navegação no nível de código Java!
- Plugins ajudam na visualização do grafo de navegação:



# Como o JSF faz tudo isso?



# Objetos Importantes

- FacesContext
- ViewHandler
- RenderKit
- Render
- LifeCycle
- Factories
- FactoryFinder
- UIViewRoot
- UIComponent
- Application
- StateManager

# Perguntas?