Introduction à JSF

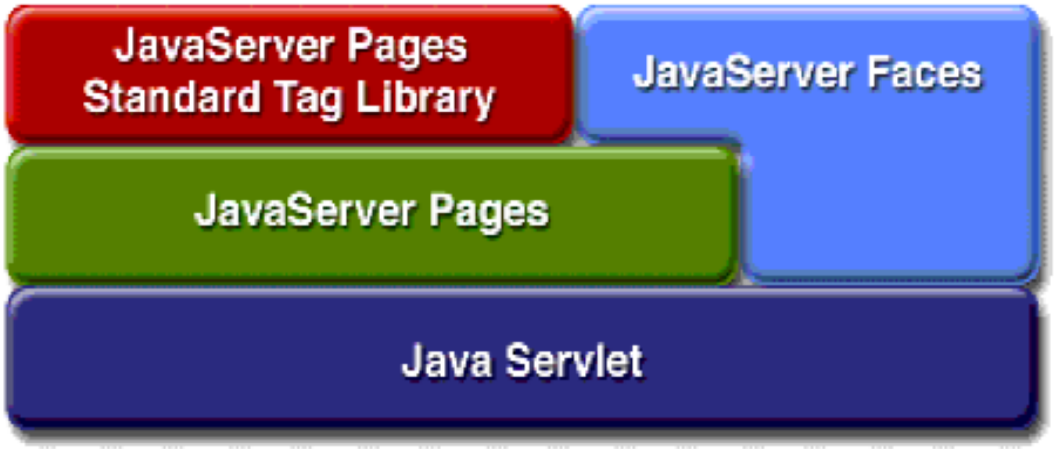
Bien débuter avec Java Server Face

Pascal Urso
François Charoy

License Creative Commons

JSF Késako ?

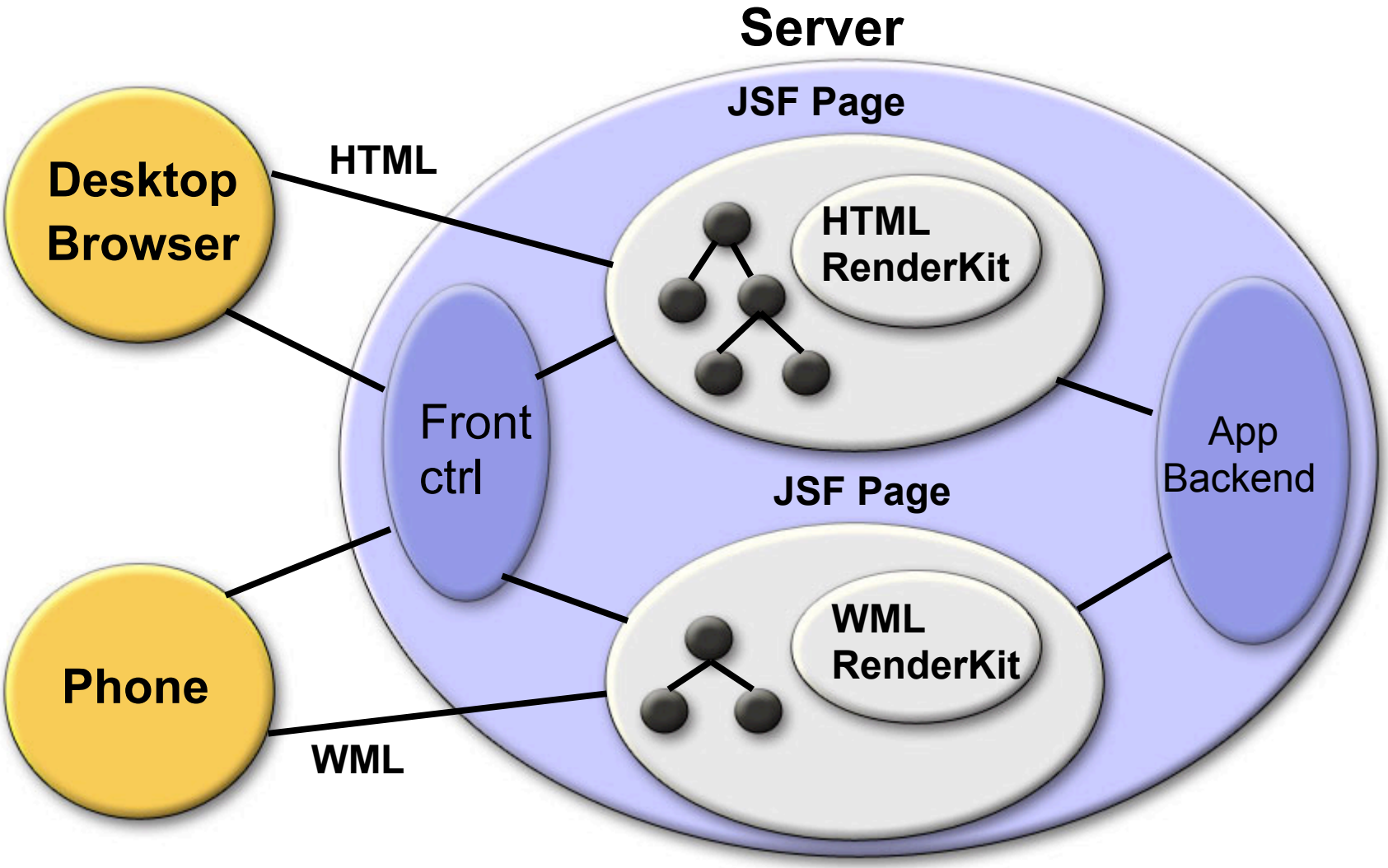
- Un framework pour le developement d'applications web
- Principes essentiels :
  - Orienté composants
  - MVC2



Parmis d'autres

v • d • e	List of Web Application Frameworks [hide]
ASP.NET	ASP.NET MVC Framework · BFC · DotNetNuke · MonoRail · Umbraco
ColdFusion	ColdSpring · Fusebox · Mach-II · Model-Glue · onTap
Java	Apache Cocoon · Apache Struts · AppFuse · Aranea framework · Click Framework · Cooee framework · framework.fleXive · Google Web Toolkit · Grails · Hamlets · IT Mill Toolkit · ItsNat · **JavaServer Faces** · JBoss Seam · Makumba · Mentawai · Oracle ADF · OpenLaszlo · OpenXava · Reasonable Server Faces (RSF) · Restlet · RIFE · Shale Framework · SmartClient · Spring Framework · Stripes · Tapestry · ThinWire · WebObjects · WebWork · Wicket framework · XTT Framework · ZK Framework
Client-side	AJILE · Clean AJAX · Dojo Toolkit · Echo · Ext · jQuery · Microsoft AJAX Library · MochiKit · MooTools · OpenLink AJAX Toolkit · Prototype JavaScript Framework · qooxdoo · Rialto Toolkit · Rico · script.aculo.us · SmartClient · Spry framework · Yahoo! UI Library
Perl	Catalyst · Interchange · Maypole · Mason
PHP	Akelos PHP Framework · CakePHP · Canvas · CodeIgniter · Drupal · eZ Publish · FUSE · Horde · Joomla! · KohanaPHP · PHP For Applications · PHPOpenbiz · PRADO · Qcodo · Seagull PHP Framework · Simplicity PHP framework · SilverStripe · Symfony · Zend Framework · Zoop Framework
Python	CherryPy · Django · Karrigell · Nevow · Porcupine · Pylons · Spyce · TurboGears · TwistedWeb · Webware · Zope
Ruby	Camping · Nitro · IOWA · Ramaze · Cerise · Merb · Ruby on Rails
Other/ Multiple languages	Alpha Five · Fusebox (ColdFusion and PHP) · Helma Object Publisher (Server-side JavaScript) · OpenACS (Tcl) · Seaside (Smalltalk) · UnCommon Web (Common Lisp) · Yaws (Erlang)

Architecture



Propriétés importantes

- Modèle de composants de vue extensible
- Modèle de rendu flexible
- Modèle de gestion des évènements
- Environnement de validation
- Support pour le flot de pages
- Internationalisation
- Accessibilité

Techniquement

- Du code JSP avec des les lib **html** et **core**

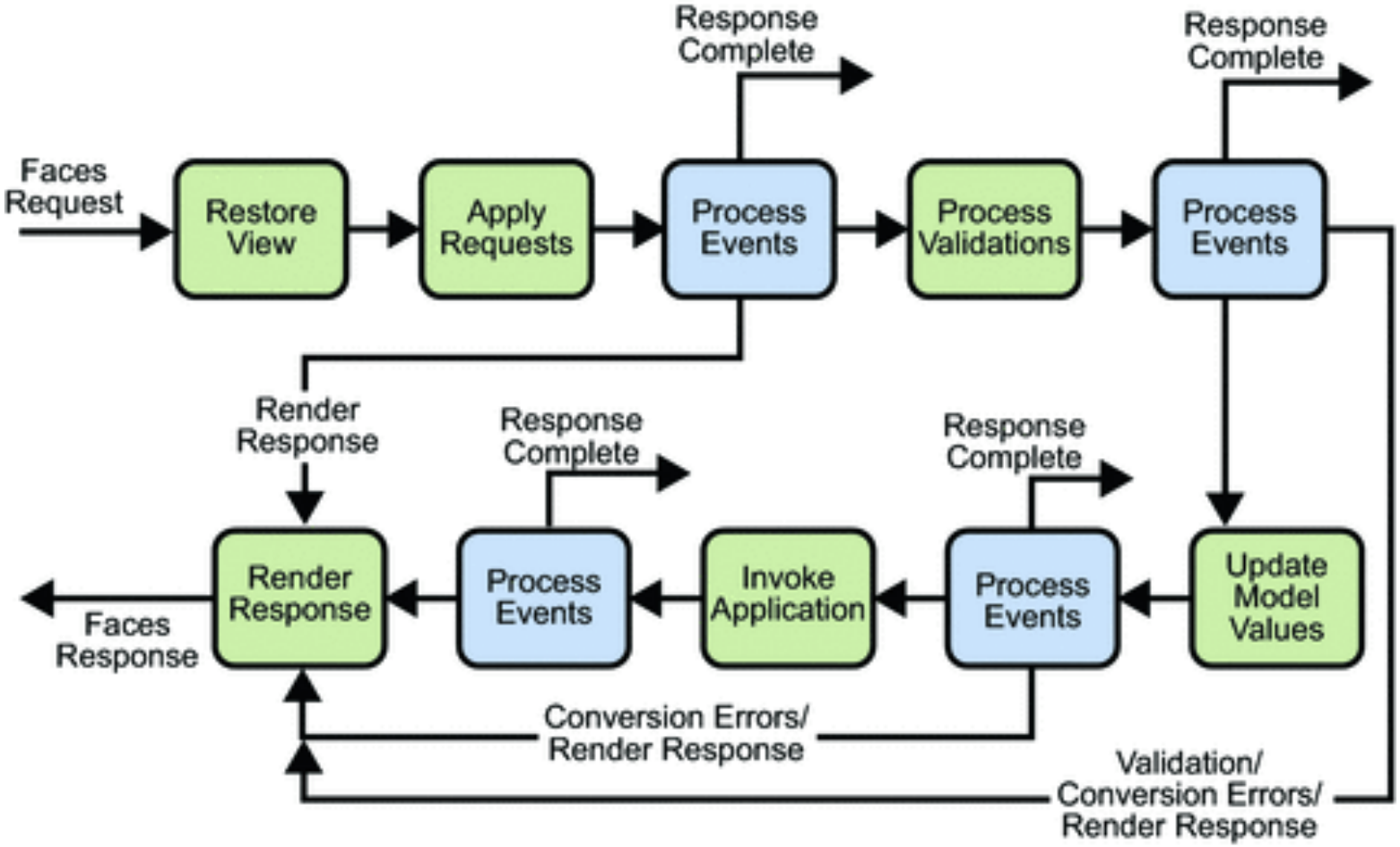
```
<%@ page contentType="text/html" %>
<%@ taglib uri="http://java.sun.com/jsf/html"
  prefix="html" %>
<%@ taglib uri="http://java.sun.com/jsf/core"
  prefix="core" %>
<core:view>
   <html:outputText value="Hello ! (en JSF !)" />
</core:view>
```

- Résultat produit par des
  - com.sun.faces.taglib.*
  - javax.faces.component.*

NetBean : Intro

- création
- code java

Traitement d'une requête



(c) Sun

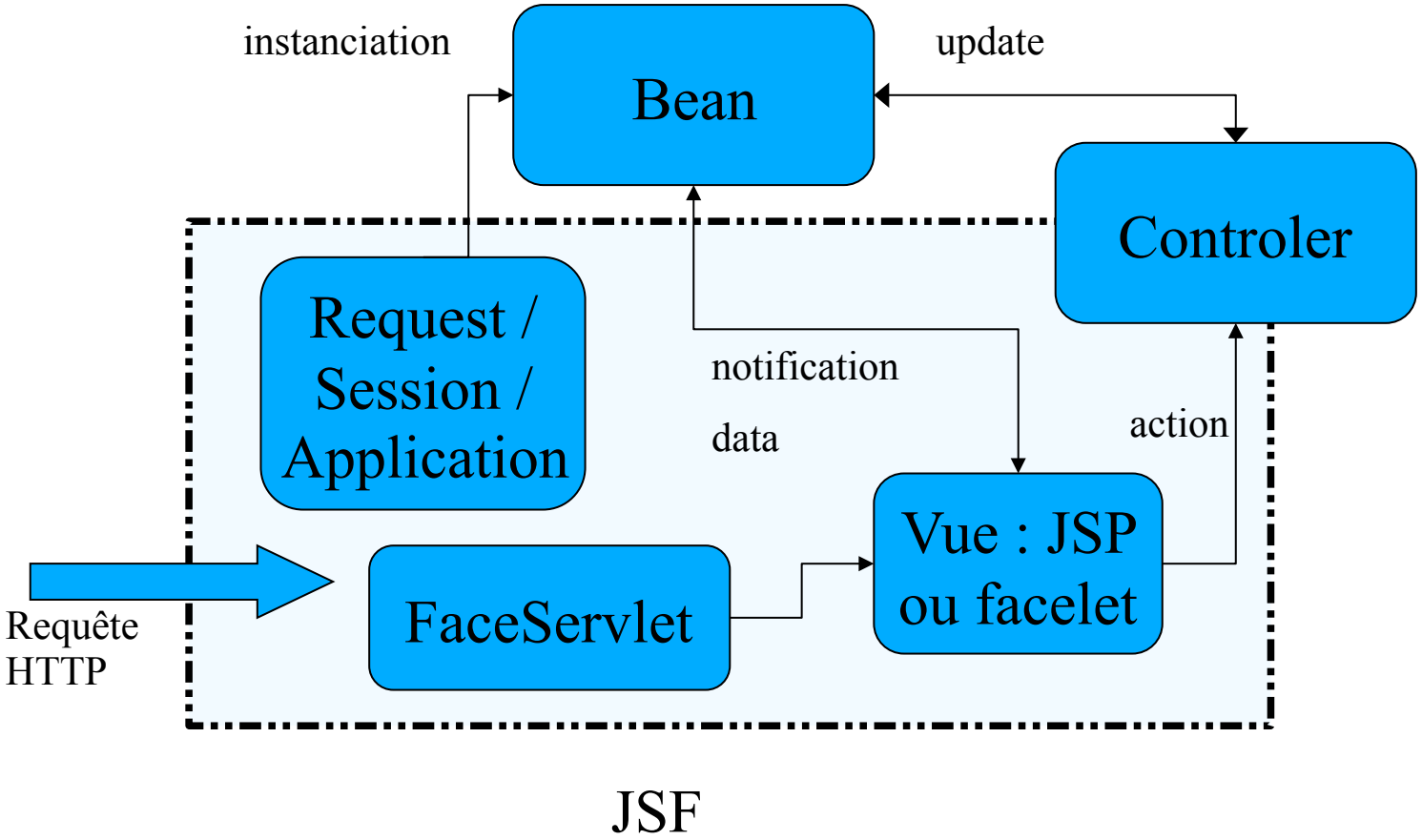
Un servlet pour les contrôler tous

- Toutes les requêtes passent par lui…
- Attention à l'URL pattern

```
<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>
</servlet-mapping>
```

Architecture MVC2

- Centré sur les beans



Beans Managés

JSF

Beans managés

- Géré par JSF
- Déclaré dans `faces-config.xml`

```
<faces-config>
 ...
  <managed-bean>
    <managed-bean-name>unBean</managed-bean-name>
    <managed-bean-class>monPackage.MonBean</managed-
  bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>
</faces-config>
```

Expression Language (EL)

```
<%@ page contentType="text/html" %>
<%@ taglib uri="http://java.sun.com/jsf/html"
prefix="html" %>
<%@ taglib uri="http://java.sun.com/jsf/core"
prefix="core" %>
<core:view>
    <html:outputText value="Un champ :" />
    <html:outputText value="#{unBean.unChamp}" />
</core:view>
```

- Appelle la méthode `getUnChamp()`.
- ! différente de l'EL jsp `${...}`
- Composition `#{unBean.unAttribut.unChamp}`

EL JSP vs EL JSF

- Unifiés depuis JSP 2.0

```
<core:view>
    <html:outputText value="#{unBean.unChamp}" />
</core:view>
```

pareil que **${unBean.unChamp}**

- EL JSP : évaluation immédiate
- EL JSF : évaluation différée (à la demande)
  - affectation
  - actions

NetBean : bean mystère

- getX
- request/session

Formulaire de Saisie

- Utilisation du tag `form`

```
<html:form>
Nouvelle valeur :
   <html:inputText value="#{unBean.….unChamp}"/>
</html:form>
```

- Positionné à la valeur du champ
- Appelle la méthode `setUnChamp()`
- Conversion automatique depuis/vers String
- Mais aussi inputTextArea, inputSecret, selectBooleanCheckbox, selectOneRadio, ...

Properties

- `<html:outputText value="#{unBean.prop}" />`

- `getProp()` peut être définie
  - sans attribut Prop
  - sans setter

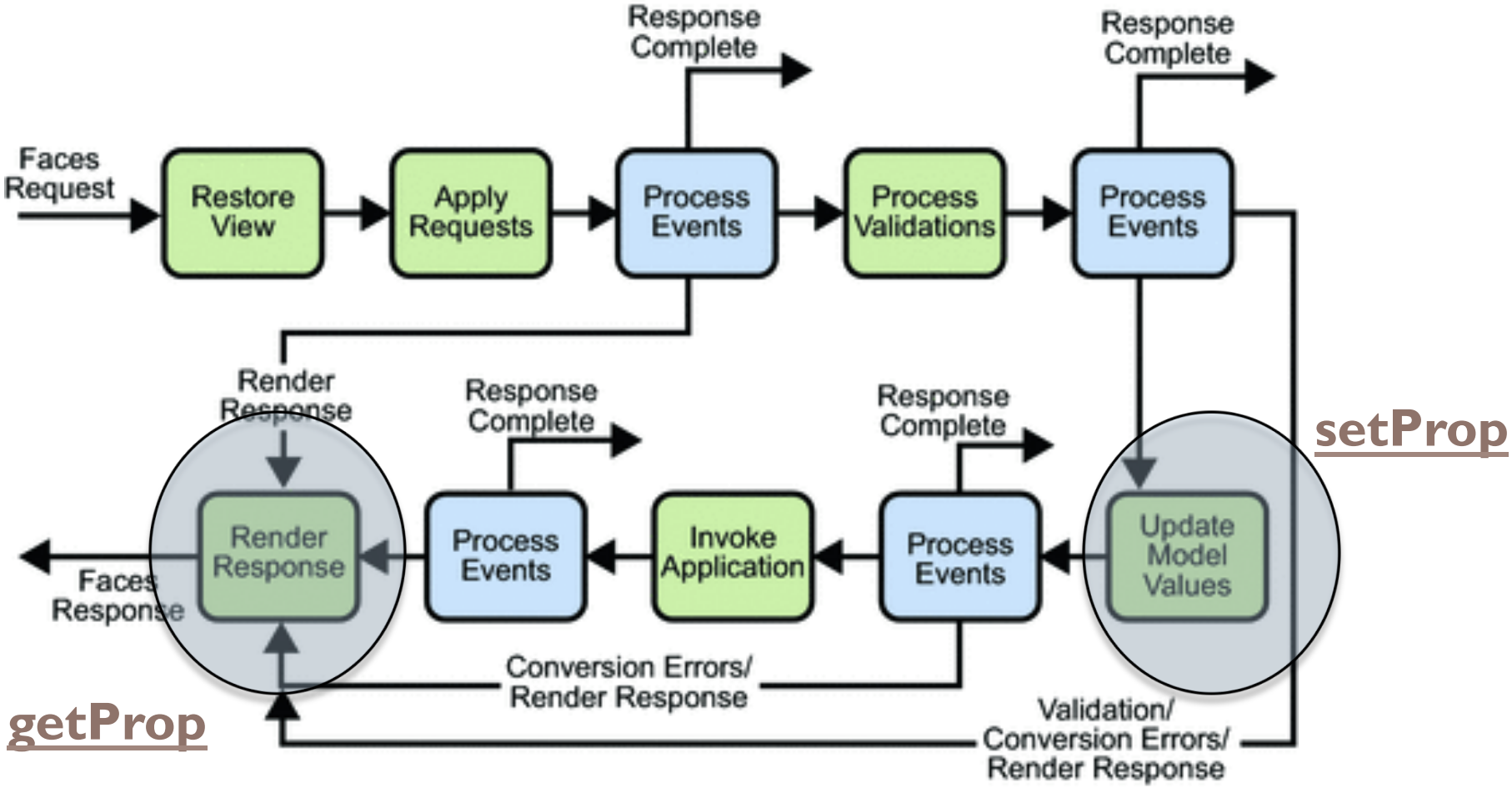
```
<html:inputText value="#{unBean.prop}" />
```

- La méthode `setProp(type valeur)` peut être définie
  - sans attribut prop
  - mais pas sans getter (valeur par défaut)

Syntaxe (*depuis JSF 2.0)*

- Bean managé : classe annoté
  - @ManagedBean("name")
- Visibilité
  - @ApplicationScoped, @SessionScoped, @RequestScoped

Traitement d'une requête



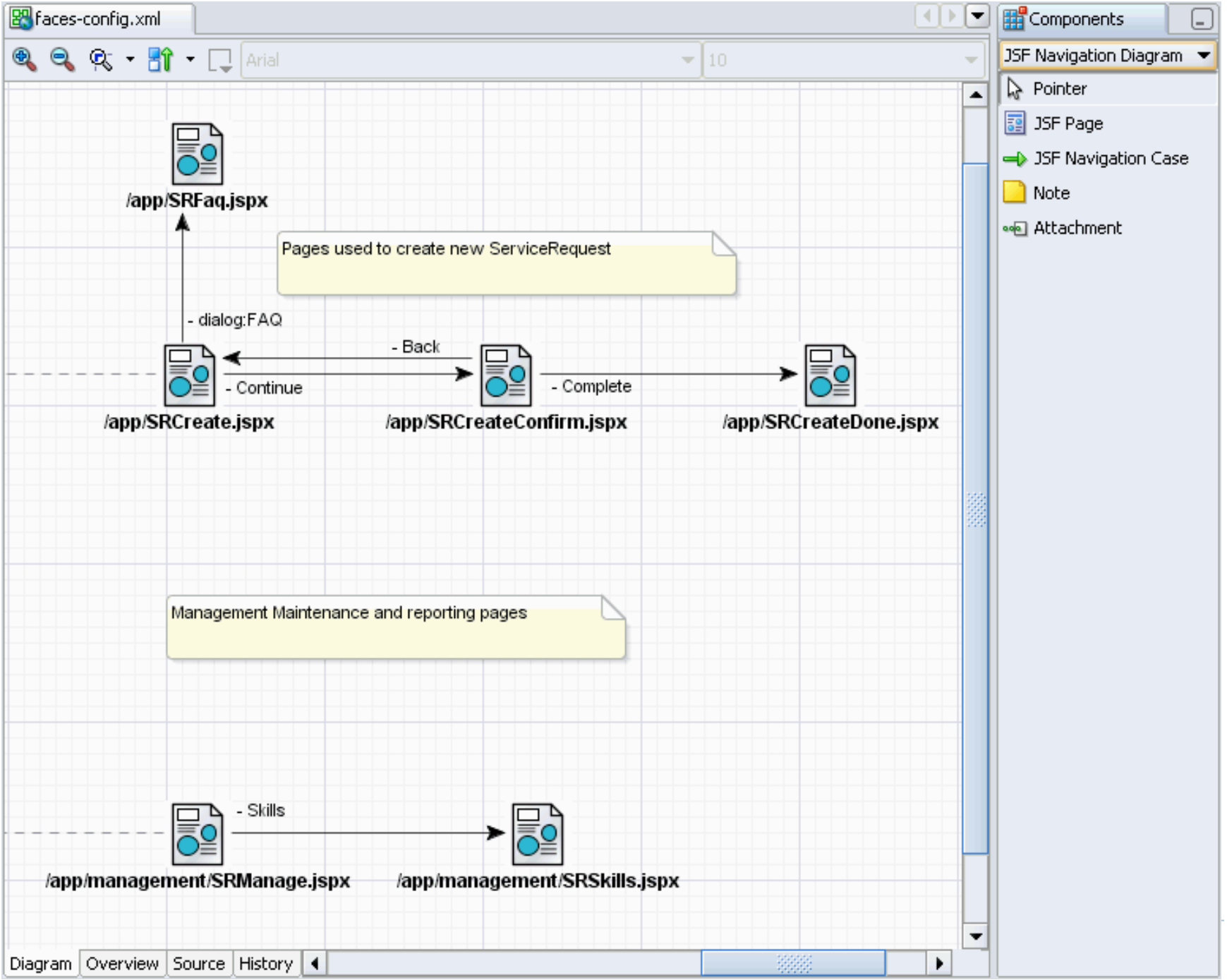
(c) Sun

Navigation

JSF

La navigation

- JSF gère un ensemble de page reliées par un flot de contrôle.
- Définit dans faces-config.xml
- Navigation
  - statique
  - dynamique


faces-config.xml
Arial
10
Components
JSF Navigation Diagram
Pointer
JSF Page
JSF Navigation Case
Note
Attachment
/app/SRFaq.jspx
Pages used to create new ServiceRequest
- dialog:FAQ
- Back
- Continue
- Complete
/app/SRCreate.jspx
/app/SRCreateConfirm.jspx
/app/SRCreateDone.jspx
Management Maintenance and reporting pages
- Skills
/app/management/SRManage.jspx
/app/management/SRSkills.jspx
Diagram
Overview
Source
History

La navigation

- JSF gère seul la navigation
- Attention à l'accès direct à une page
- Prévoir un mécanisme pour bloquer l'accès (si besoin)
  - tests
  - filtres

Navigation Statique

- Dans la page JSP : toujours dans un formulaire
  - HyperLien HTML

```
<html:form>
    <html:commandLink action="actionA" value="Par ici" />
</html:form>
```

  - Boutton HTML

```
<html:form>
    <html:commandButton action="actionA" value="Par ici
  aussi" />
</html:form>
```

Navigation Statique

- Dans `faces-config.xml`

```
<navigation-rule>
  <from-view-id>/page1.jsp</from-view-id>
  <navigation-case>
      <from-outcome>actionA</from-outcome>
      <to-view-id>/page2.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
      <from-outcome>actionsB*</from-outcome>
      <to-view-id>/page3.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation Statique (wildcard)

- Depuis plusieurs pages jsf

```
<navigation-rule>
  <from-view-id>*</from-view-id>
  <navigation-case>
      <from-outcome>actionA</from-outcome>
      <to-view-id>/page2.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
      <from-outcome>actionsB*</from-outcome>
      <to-view-id>/page3.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation Statique (default)

- Sans *outcome* : comportement par défaut

```
<navigation-rule>
  <from-view-id>/page1.jsp</from-view-id>
  <navigation-case>
      <from-outcome>actionA</from-outcome>
      <to-view-id>/page2.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
      <to-view-id>/default.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

NetBean : Navigation

- form

Navigation Dynamique

- Appel d'une méthode du bean pour calculer l'outcome

```
<html:form>
  <html:commandButton value="GO"
      action="#{unBean.uneMethode}"/>
</html:form>
```

- Méthode renvoyant un String
- Outcome non prévu dans le flot :page courante rechargée

Navigation Dynamique

- Possibilité de préciser la méthode

```
<navigation-rule>
  <from-view-id>/page1.jsp</from-view-id>
  <navigation-case>
      <from-action>#{unBean.methA}</from-action>
      <from-outcome>resultA</from-outcome>
      <to-view-id>/page2.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
      <from-action>#{unBean.methB}</from-action>
      <from-outcome>resultA</from-outcome>
      <to-view-id>/page3.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation implicite *(depuis JSF 2.0)*

- Plus « usuel »
- On indique directement le nom de la page cible

```
<html:commandLink action="unePage.jsp" value="Par
  ici" />
<html:commandButton action="unePage.jsp"
  value="Soumettre" />
```

- Marche aussi avec une méthode revoyant le nom d'une page

```
<html:command… action="#{...}" value="Par là" />
```

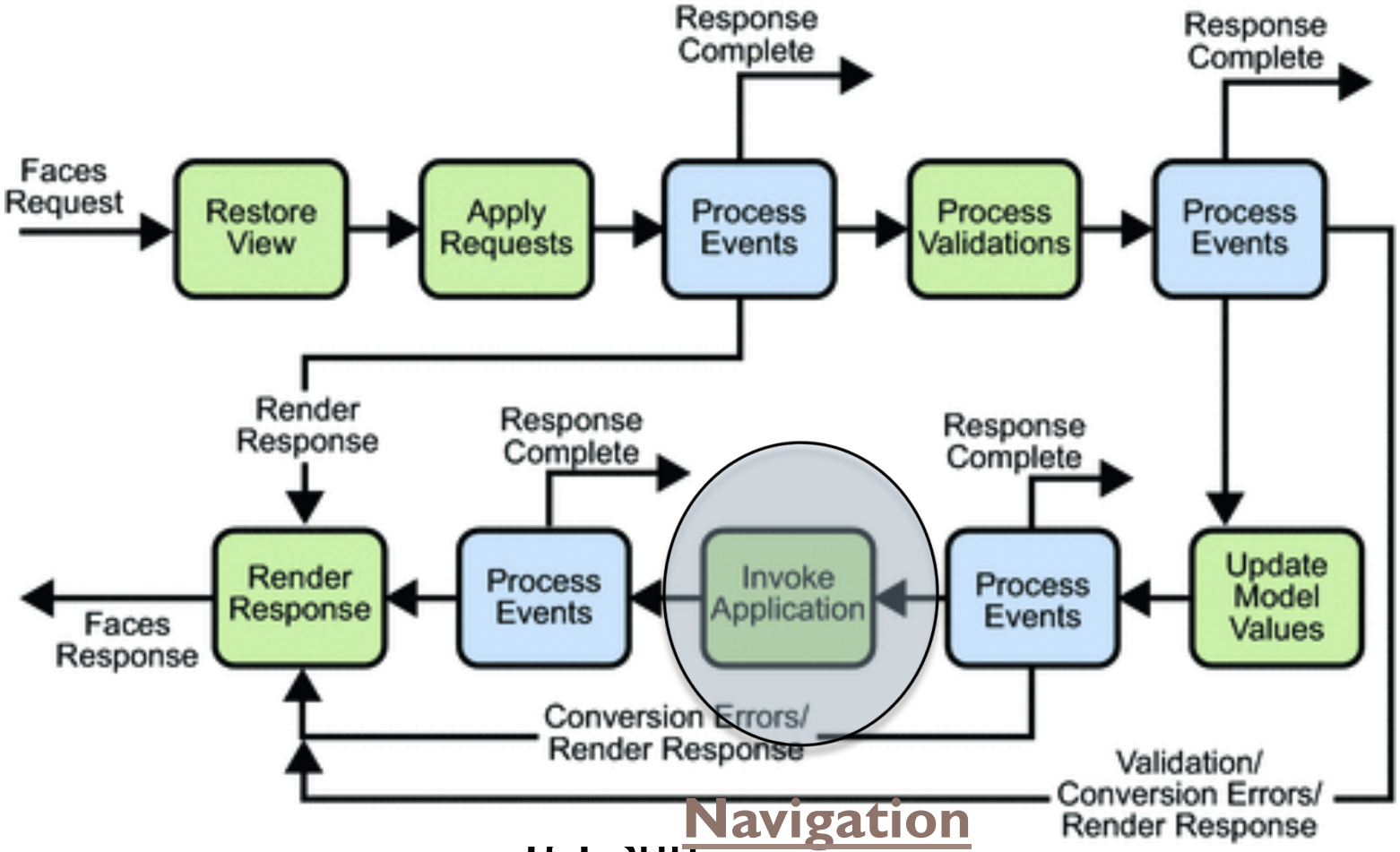
Limitations

- Pas d'argument pour les méthodes (get, set, actions)
- Impossible de définir une action par défaut dans les formulaires
  - un seul bouton « action », le reste « actionListener » ou commandLink
- Séparation modèle/vue/contrôle
  - Vue : JSF
  - Contrôle : bean managé
  - Modèle : autres classes java

NetBean : play

- guess
- 3 outcomes
- Link +1
- html

Traitement d'une requête



Navigation

Conception MVC

JSF

Conception MVC

- *Deux niveaux de contrôle*
- Contrôle des requêtes : pris en charge par JSF
- Contrôle applicatif : bean managé
  - **Modèle** : Classes métier
  - **Vue** : Pages JSP (ou facelet)
  - **Contrôle** : Beans managés
- Séparer le contrôle applicatif des classes métiers !!

NetBean : properties

- Refactor
- MVC : choose

- triche

Table de Données

JSF

Table de données

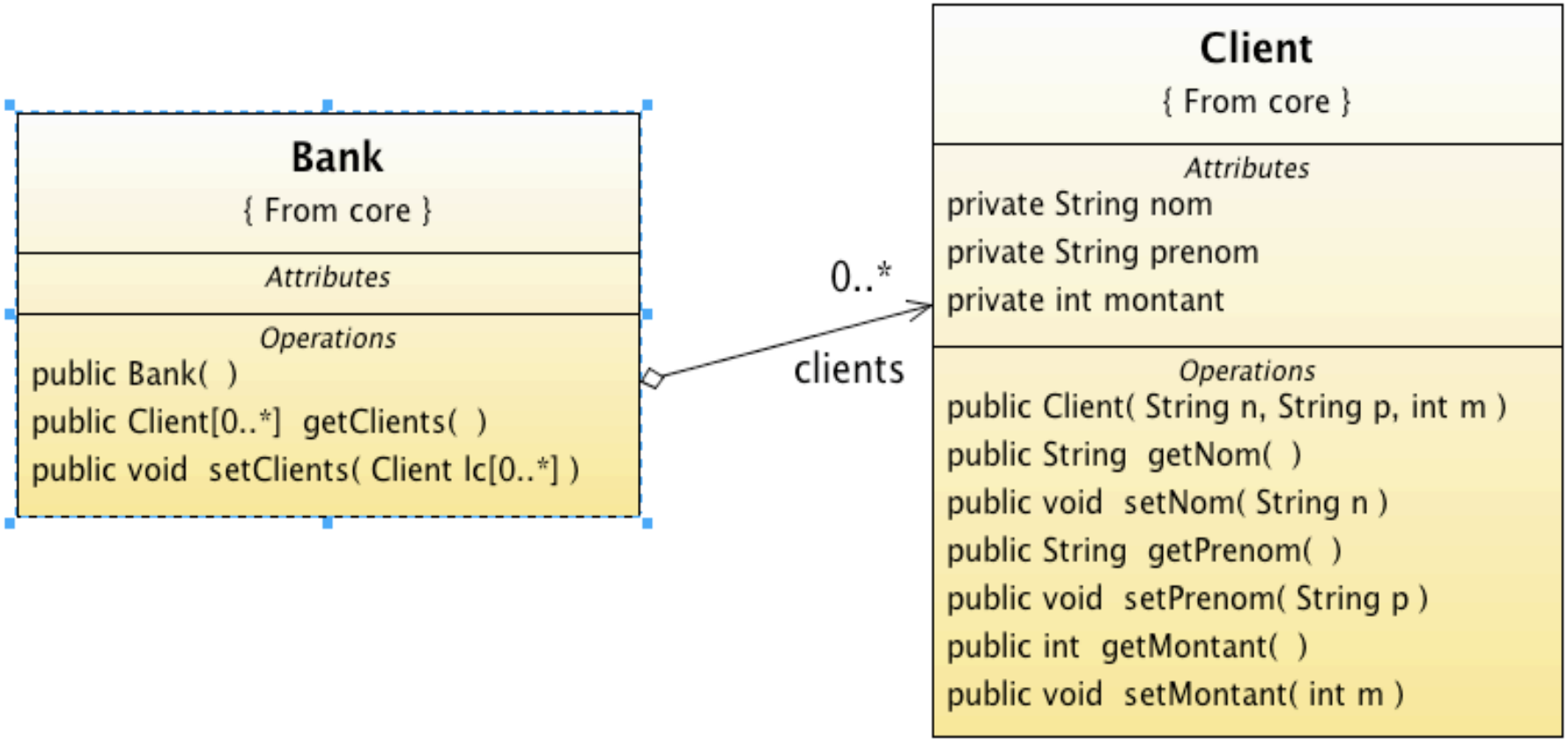


Table de données

```
<html:dataTable value="#{bank.clients}" var="client"
  border="1">
    <html:column>
       <html:outputText value="#{client.nom}"/>
    </html:column>
    <html:column>
       <html:outputText value="#{client.prenom}"/>
    </html:column>
</html:dataTable>
```

Table de données : Header

```
<html:dataTable value="#{bank.clients}" var="client"
  border="1">
    <html:column>
       <core:facet name="header">
          <core:verbatim>Nom</core:verbatim>
       </core:facet>
       <html:outputText value="#{client.nom}"/>
    </html:column>
    ...
</html:dataTable>
```

- **facet** : association avec le tag (et non avec le rendu du contenu)

NetBean : table

- demo
- avec input ?

Binding

```
<html:dataTable binding="#{bank.dataTable}" value="#
  {bank.clients}" var="client" border="1">
    <html:column>
       <core:facet name="header">
          <core:verbatim>Nom</core:verbatim>
       </core:facet>
       <html:outputText value="#{client.nom}"/>
    </html:column>
    ...
</html:dataTable>
```

- Lier le bean avec le composant UI
- *N'importe quel composant* : input, box, column...

Binding

```
import javax.faces.component.UIData;
public class Bank {
 // UIData ancêtre de HtmlDataTable
    private UIData dataTable;
    public UIData getDataTable(){
        return this.dataTable;
    }
    public void setDataTable(UIData dT){
        this.dataTable = dT;
    } ...
```

- Composant UI : descendant de `javax.faces.component.UIComponent`

Binding example

```
<html:dataTable binding="#{bank.dataTable}" value="#
  {bank.clients}" var="client" border="1">
      ...
   <html:column>
   <html:selectBooleanCheckbox
            binding="#{bank.checkbox}"/>
   </html:column>
</html:dataTable>
<html:commandButton value="Supprimer les clients"
  action="#{bank.supprClientSelection}"/>
<html:inputText binding="#{bank.nvNom}"/>
<html:inputText binding="#{bank.nvPrenom}"/>
<html:commandButton value="Ajouter un client"
  action="#{bank.ajoutClient}"/>
```

Binding Example

```
class Bank {
  ...
  private UIInput nvNom;
  public UIInput getNvNom() {
    return this.nvNom;
  }
  public void setNvNom(UIInput c) {
    this.nvNom = c;
  }
  public void ajoutClient(){
    Client c = new Client((String) nvNom.getValue(),
                                (String) nvPrenom.getValue(),
                            0);
    this.clients.add(c);
  }
```

Binding Example

```
  public void supprClientSelection() {
    int size = this.dataTable.getRowCount();
    List<Client> selectedCustomers = new ArrayList<Client>
   ();
    for (int i = 0; i < size; i++) {
      this.dataTable.setRowIndex(i);
      if (this.checkbox.isSelected())
        selectedCustomers.add(this.clients.get(i));
    }
    this.clients.removeAll(selectedCustomers);
  }
}
```

- checkbox associé à la colonne

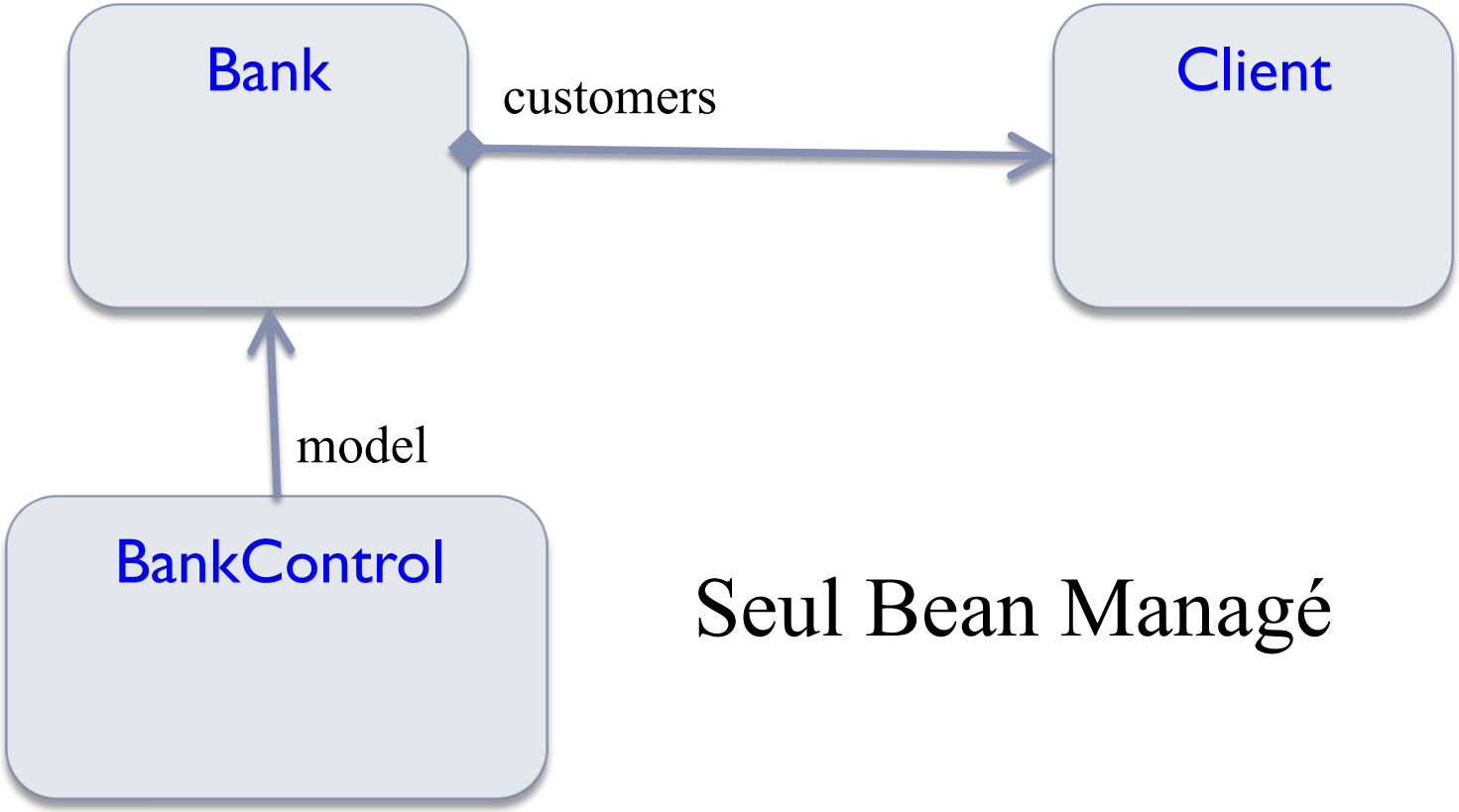
NetBean : Binding

- demo
- breakpoint

Et le MVC alors ?

- La classe Bank s'occupe à la fois de
  - la vue (composants UI)
  - le modèle (données)
  - le contrôle (suppression, ajout)
- Réifions tout ça !

MVC : Conception



Seul Bean Managé

MVC : JSP

```
<html:dataTable binding="#{bankCtrl.dataTable}"
  value="#{bankCtrl.model.customers}" var="customer"
  border="1">
  ...
  <html:column>
     <html:selectBooleanCheckbox binding="#
  {bankCtrl.checkbox}"/>
  </html:column>
</html:dataTable>
<html:commandButton value="Supprimer les clients"
  action="#{bankCtrl.removeSelectedCustomers}"/>
<html:inputText binding="#{bankCtrl.nvNom}"/>
<html:commandButton value="Ajouter un client"
  action="#{bankCtrl.addCustomer}"/>
```

MVC : faces-config.xml

```
<managed-bean>
  <description>
    Une un peu moins bete liste de client
  </description>
  <managed-bean-name>bankCtrl</managed-bean-name>
  <managed-bean-class>core.mvc.Controler</managed-
  bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Agir sur les éléments d'une Table

```
<html:dataTable binding="#{bankCtrl.dataTable}"
  value="#{bankCtrl.model.customers}" var="customer"
  border="1">
    <html:column>
<html:inputText value="#{customer.cash}"/>
    </html:column>
    <html:column>
<html:commandLink value="Retrait" action="#
  {customer.withdraw}"/>
    </html:column>
</html:dataTable>
```

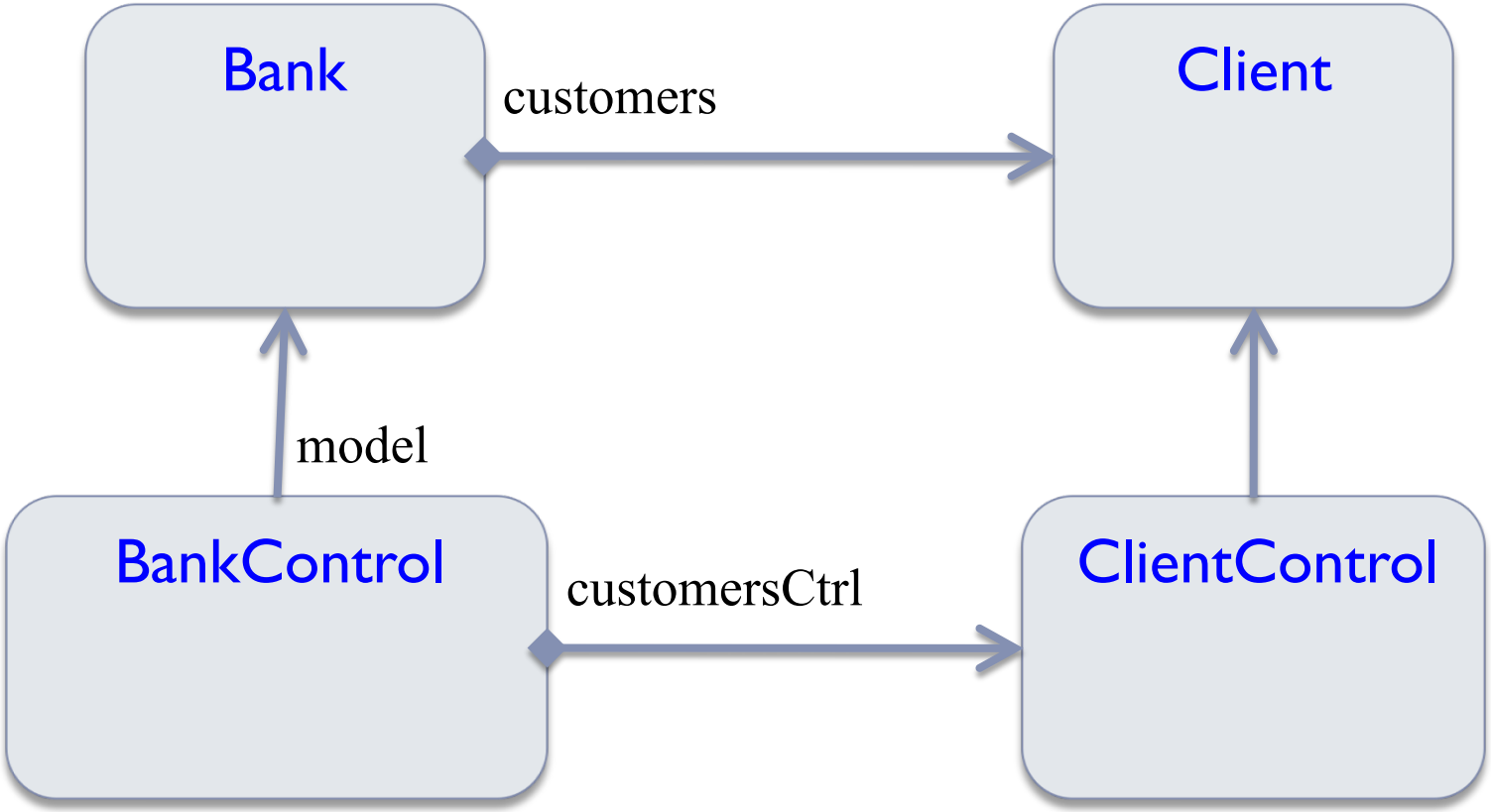
– *PAS MVC*

Agir sur les éléments d'une Table

```
<html:dataTable binding="#{bankCtrl.dataTable}"
  value="#{bankCtrl.customersCtrl}" var="customer"
  border="1">
    <html:column>
<html:inputText value="#{customer.cash}”/>
    </html:column>
    <html:column>
<html:commandLink value="Retrait" action="#
  {customer.withdraw}"/>
    </html:column>
</html:dataTable>
```

- *Une liste contrôleur associés à chaque client*

MVC : Conception 2



Seul Bean Managé

Conclusion MVC

- **Vue** :
  - Pages jsf
- **Contrôle** :
  - Beans managés
- **Model** :
  - Classes métiers

Example

- Projet JSF promotion 2007-2008
  - « un (portail de) jeu asynchrone en ligne à l'aide du framework JSF. »
  - *« Vous pouvez choisir le jeu que vous voulez ... du moment qu'il s'agit d'un jeu à au moins 2 joueurs au tour-par-tour. »*

Plus sur les Formulaires

JSF

<h:form>

- Questions ?
- http://java.sun.com/javaee/javaserverfaces/reference/api/
  - Tag lib documentation
    - html **<h:**
    - core **<f:**
  - API **java.faces.*.***
- Formulaires <h:form>
- Méthode toujours POST
  - car toujours un effet de bord (bean, compUI)

Eléments Simples

- <h:outputText value="#{unBean.uneProp}"/>

- <h:inputText value="#{unBean.uneProp}"/>
  - valeur pour l'initialisation et la soumission

Nom: Toto

- <h:inputSecret value="#{unBean.uneProp}"/>
  - valeur à la soumission uniquement

Mot de passe: ****

Boutons

- *<h:commandButton*
  - *value="Label"*
- Peut être une EL "#{unBean.uneProp}"
  - *action="#{unBean.uneMethode}"*
- Peut être statique "actionX"
  - *actionListener="#{unBean.uneMethode}" immediate="true"*
- Remplace l'action. Le formulaire est réaffiché **sans être soumis** (setter non appelés).
  - image="Img"
- Bouton image (coordonnés du click : *listener*)

Lien

- *<h:commandLink>*
  - attributs *value, action, actionListener*
  - paramètres <f:param name="..." value="..." />
  - exemple

```
<html:dataTable ... var="elem">
   ...
    <html:column>
       <html:commandLink value="Supprimer"
   action="delete">
          <f:param name="id" value="#{elem.id}">
       </html:commandLink>
    </html:column>
</html:dataTable>
```

Valeur des Paramètres

- Directement : **EL** `"#{param.id}"`
- Dans un Bean : **faces-config.xml**

```
<managed-bean>
   ...
   <managed-property>
    <property-name>id</property-name>
    <value>#{param.id}</value>
   </managed-property>
</managed-bean>
```

- Dans le code d'un bean managé

```
FacesContext.getCurrentInstance().getExternalContext
   ().getRequest()
```

Elément avec valeur

- Check boxes
- Comboboxes
- List boxes
- Radio Buttons
- Textfields

- <h:selectBooleanCheckbox>
- <h:selectOneMenu>
  <h:selectManyMenu>
- <h:selectOneListbox>
  <h:selectManyListbox>
- <h:selectOneRadio>
- <h:inputText>
  <h:inputTextArea>
  <h:inputSecret>

Tous avec attribut valueChangeListener="#{unBean.uneMethode}"

Sélection

- Combo, List, Radio

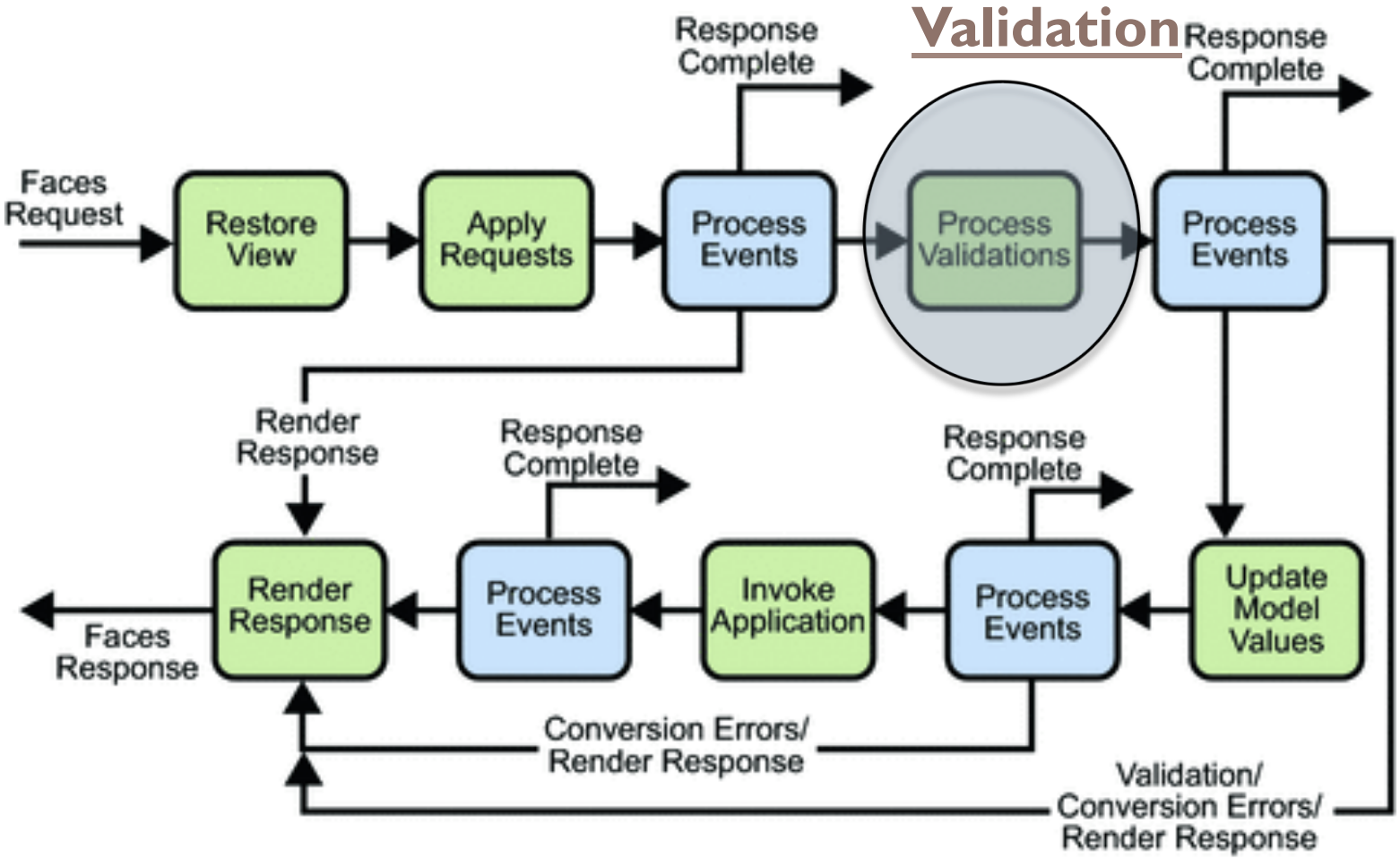
```
<h:select... value="#{unBean.uneProp}">
    <f:selectItems value="#{unBean.unePropList}"/>
</h:select...>
```

- *uneProp* : valeur initiale et de soumission
  - une *String* pour *selectOne*...
  - une *List<String>* pour *selectMany*...
- *unePropListe* : une collection (liste ou tableau) de *java.faces.model.SelectItem*

Validation

JSF

Traitement d'une requête



(c) Sun

Validation de Formulaire

- Indispensable
- Peut être fait à la main au moment de la soumission (chiant)
- Validation JSF
  - Identifiant des champs
  - Fonctionnement par exception

Champ Requis

- <h:inputText value="#{unBean.uneProp}" **required="true" id="unId"**/>
- <h:message **for="unId"** styleClass="..."/>

- style CSS : errorMessage (par exemple)

Nom:

Erreur de validation: Valeur requise.

Validation Explicite

- <f:validateLength minimum="X" maximum="Y"/>
- <f:validateLongRange minimum="X" maximum="Y"/>
- <f:validateDoubleRange minimum="X" maximum="Y"/>
- Ou que minimun ou que maximun
- Exemple

```
<h:inputText value="#{bidBean2.userID}" id="userID">
    <f:validateLength minimum="6"/>
</h:inputText>
<h:message for="userID" styleClass="..."/>
```

Validation Ad-hoc

- <h:inputText value="#{unBean.uneProp}" id="unId" validator="#{unBean.uneMethode}"/>
- <h:message for="unId" styleClass="..."/>

- Méthode :

```
public void validate(FacesContext context,
UIComponent componentToValidate, Object value)
throws ValidatorException { ... }
```

Validation par annotation Hibernate *(JSF 2.0)*

- **@Min**: The annotated element must be a number whose value must be higher or equal to the specified minimum.
- **@Max**: The annotated element must be a number whose value must be lower or equal to the specified maximum.
- **@Size**: The annotated element must be between specified minimum and maximum boundaries.
- **@NotNull**: The annotated element must not be null.
- **@Pattern**: The annotated element must match the specified Java regular expression.
- **@Valid:** For a collection or a map, checks that all the objects they contain are valid
- **@Email:** Checks whether the string conforms to the email address specification
- …

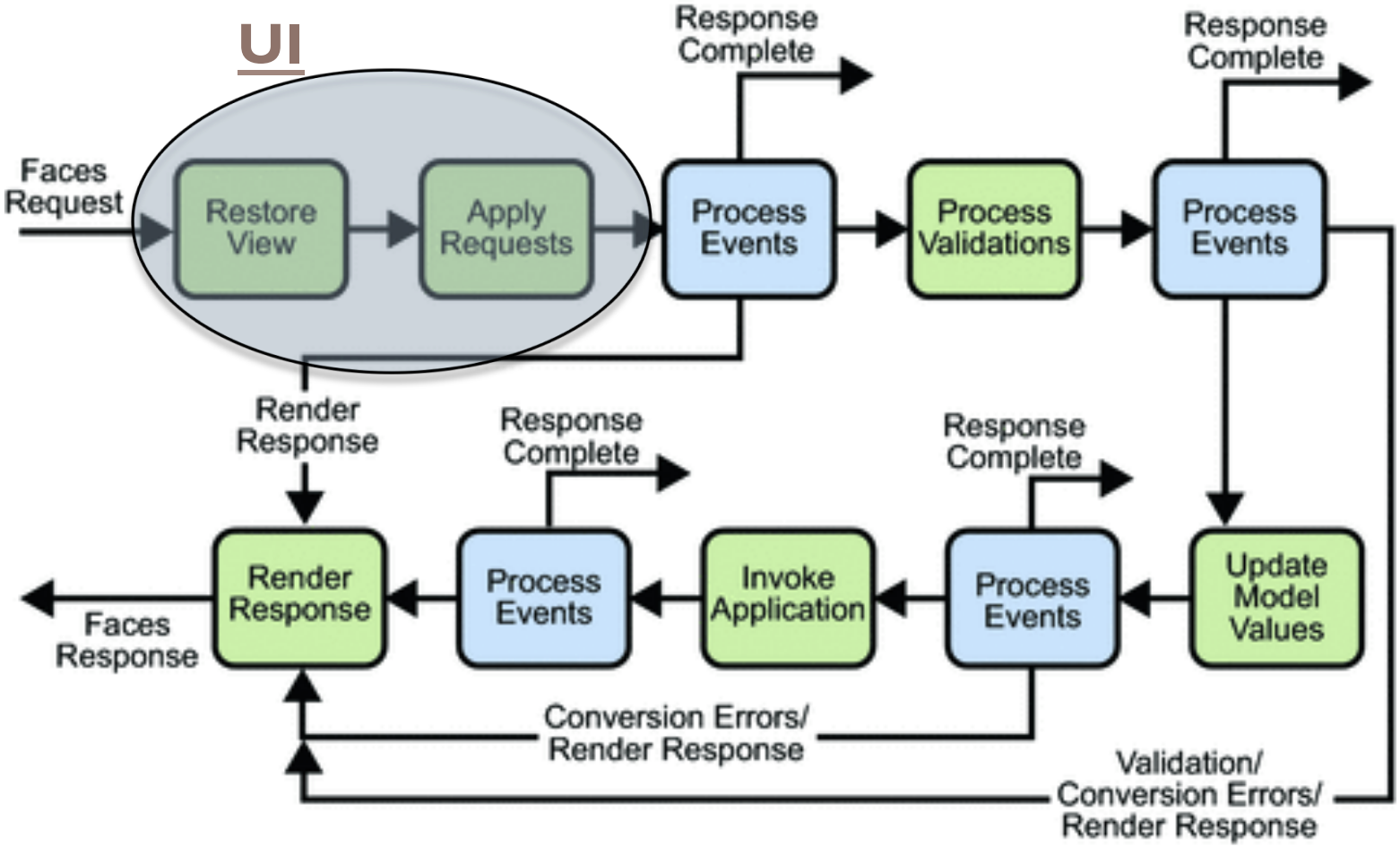
Composant UI

JSF

Composants UI

- Tous les éléments d'interface (<h:input…>, <h:select…>) sont des composants.
- Mais aussi view, outputText,

Traitement d'une requête



(c) Sun

Facelet

JSF

Facelet

- Kesako ?
  - Une autre façon (que JSP) propre à JSF pour présenter la vue HTML
- Pourquoi ?
  - JSP = servlet (confusion, cycle de vie)
  - Plus léger
  - Templates
  - Extensible
- Format : ***XHTML***

helloWorld.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">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        Hello from Facelets
    </h:body>
</html>
```

- A première vue rien de neuf

Définition du template *template.xhtml*

```
<!DOCTYPE html …>
<html xmlns:ui="http://java.sun.com/jsf/facelets" …> <head>
    <title>
        <ui:insert name="titre" />
    </title>
  </head>
  <body>
    <f:view>
        <p> <ui:insert name="entete">
                <h:outputText value="entete generique" />
        </ui:insert> </p>
        <div> <ui:insert name="menu" /> </div>
        <p> <ui:insert name="contenu" /> </p>
    </f:view>
</body> </html>
```

Valeur par défaut

Utilisation du template

```
<!DOCTYPE html …> <html …>
<ui:composition template="/template.xhtml">
  <ui:define name="titre">FAQ Java</ui:define>
  <ui:define name="entete"> <h:outputText value="Entete Java" > </
  ui:define>
  <ui:define name="menu">
    <h:form>
        <h:commandLink action="FAQ.xhtml" value="FAQ" />
        <h:commandLink action="tutoriels.xhtml
  "action="tutoriels.xhtml" value= "Tutoriels" />
        <h:commandLink action="forums.xhtml" value="Forums" />
    </h:form>
  </ui:define>
  <ui:define name="contenu"> <h:outputText value="Les FAQs Java" > </
  ui:define>
</ui:composition>
</html>
```

Création de composant : *zone.xhtml*

- Similaire a la définition de tag jsp (mais sans java)
- Un composant est une composition

```
<!DOCTYPE html …> <html …>
<ui:composition>
  <ui:insert />
  <h:outputText value="#{titre}" styleClass="titre" />
  <h:inputText value="#{valeur}" styleClass="zoneTexte" />
</ui:composition>
</html>
```

- Avec des paramètres #{…}
- Et un corps `<ui:insert />`

Déclaration du composant

- Fichier */WEB-INF/taglib.xml*

```
<?xml version="1.0"?>
<!DOCTYPE facelet-taglib PUBLIC "-//Sun Microsystems,
  Inc.//DTD Facelet Taglib 1.0//EN" "facelet-
  taglib_1_0.dtd">
<facelet-taglib>
  <namespace>http://www.uhp.fr/monappli</namespace>
  <tag>
      <tag-name>zoneDeTexte</tag-name>
      <source>composants/zone.xhtml</source>
  </tag>
</facelet-taglib>
```

Déclaration de la taglib

- Fichier *web.xml*

```
<context-param>
  <param-name>facelets.LIBRARIES</param-name>
  <param-value>/WEB-INF/taglib.xml</param-value>
</context-param>
```

Utilisation du tag

```
<!DOCTYPE html …>
<html … xmlns:mt=" http://www.uhp.fr/monappli">
  <body>
    <f:view>
       <mt:zone titre="Hello World"
              valeur="#{unManagedBean.unChamp}">
         <h:graphicImage value="img/arrow.png" alt="-&gt;"/>
       <mt:zone/>
    </f:view>
  </body>
</html>
```

Quelques mots supplémentaire sur la conception Web

Au milieu d'un océan

Ce que l'on n'a pas vu

- Persistance
  - Framework Hibernate, JDO, JPA
- Client (mi-)lourd : Javascript, AJAX
  - Framework AJAX: RichFaces, IceFaces, GWT…
- Tests
  - Framework HTTPUnit, JSFUnit, …
- Passage à l'échelle
  - Cache, cache distribué
  - **Du gros, du lourd , du cloud:** NoSQL, App engine, MAP/REDUCE, CASSANDRA
- Ergonomie
  - CSS, …
- Web Sémantique
- …

Test et servlet

Tests et servlet

- A la base pas facile
  - Appel Web
  - Impact du container
  - Problèmes de session
  - Concurrence, etc
- Librarie HTTPUnit, JSFUnit, …
- A la base des mocks sur des appels aux méthodes des servlet (doGet, …)

Quelques mots sur la persistance

JPA + JSF

Persistance

- Peut se faire
  - Par des fichiers classique
  - Par des appels JDBC (c'est du java)
  - Par la librairie de tag sql (jsp)
  - Par des framework spécifique (hibernate, JDO, …)
  - Par le standard javaee JPA