



JSF II

Facelets, Richfaces

Advanced JSF

Petr Aubrecht
CA

An SQL query goes to a restaurant, walks up to 2 tables and says:
“Can I join you?”

Co dnes probereme

- Ukázka – servlet vs. JSP vs. JSF (plus komponenty)
- JSF 2.0
 - bean validation, action methods s argumenty, anotace backing bean, AJAX
 - knihovny komponent, std. komponenty, life cycle
 - value-change Listener
 - šablony
 - vlastní komponenta
 - composite components
 - resource handling (resources directory)
 - advanced navigation, bookmarkable pages

Bean Validation

- Validace je umístěna v JEE 6 v modelu jako anotace.
- Jakýkoliv framework je může využít – JPA, JSF ap.
- Není třeba tyto informace uvádět v každém frameworku znova.

```
public class Address {  
    @NotNull @Size(max=30)  
    private String addressline1;
```

- Je možné definovat i vlastní validátory (např. ZIP code).
- Jak se validace používá? Je automatická!

Backing bean metody s argumenty

- Od JSF 2 můžete používat argumenty u action metod.
- Dost návykové, snadno se stane špatným modelem a všechno chcete předávat přes argumenty. Je to potřeba používat s mírou.

```
<h:commandButton value="target.xhtml" action=  
    "# {myBean.testNavigation1('target.xhtml')}" />
```

Rušíme další XML – anotace backing bean

- @Named(„foo“)
@SessionScoped
public class Foo {
}
- Podobně @FacesComponent, @FacesRenderer,
@FacesConverter, @FacesValidator, @FacesBehavior
- XML má samozřejmě přednost jako vždy

Support for Ajax in JSF 2.0

- Hlavní myšlenkou jsou částečné updaty, kdy je změněna pouze část stránky. V RichFaces byla tato funkcionality poskytována knihovnou a4j.

```
<h:commandButton id="button1">
```

```
    <f:ajax execute="..." render="..." />
```

```
</h:commandButton>
```

- Tento způsob má několik velice výhodných vlastností:
 - rychlejší odezva (generuje se pouze malá část stránky)
 - lokální změna (např. zůstanou vybrané položky v listech, selekce textu ap.)

Demo

- PerformanceDemo
 - servlet vs. JSP vs. JSF (plus komponenty)
- JSF2Demo
 - drag&drop JSF/JSF data table & form, AJAX support
 - validate
 - navigace ve faces-config.xml, pouze targetem s příponou i bez
 - navigace immediate
 - navigace s redirect
 - použití argumentů

Knihovny

- Nad JSF se dají snadno budovat knihovny widgetů. Takovýchto knihoven se opět vyrojil bezpočet. Nejslavnější jsou richfaces (JBoss) a myfaces (Apache), ale vlastní má i Oracle, icefaces, ajax4jsf...
- Většinou se jedná o celé soubory UI komponent, nad kterými se dá postavit aplikace.
- Některé knihovny pouze rozšiřují stávající komponenty o AJAX chování.
- Zkuste v Googlu „jsf components“.

Richfaces

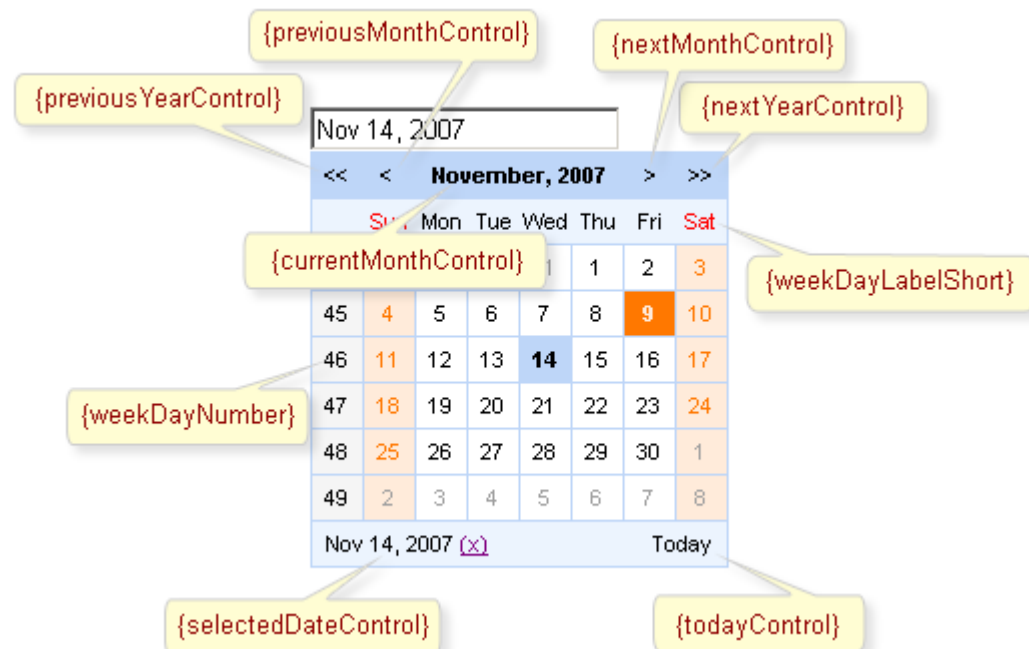
- http://www.jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html

- podle mě nejhezčí komponenty

```
<rich:calendar id="date" value="#{bean.dateTest}">
```

```
  <a4j:support event="ondateselected" reRender="mainTable"/>
```

```
</rich:calendar>
```



Rich Faces

- vyvíjeno v rámci projektu JBoss (open source JEE server, koupen před několika lety RedHatem, dnes nejvýnosnější divize)
- <http://www.jboss.org/file-access/default/members/jbossrichf>
- návod na instalaci zní lákavě, ovšem počítá s tím, že jste fandové Jakarta Commons a máte všechny knihovny
- použití s facelety:
http://wiki.java.net/bin/view/Projects/FaceletsFAQ#How_do_
- příklad: simple echo
- <http://www.jboss.org/file-access/default/members/jbossrichf>

Client

JSP Page

8. Update Page

1. JS Event

Ajax Engine

2. Submit Request

7. Send Response

Server

RichFaces

UIViewRoot

XML Filter

InternetResourceBuilder

AjaxViewRoot

6. Check XML

5. Build Resources

Ajax RenderKit

4. Encode Region

3. Progress Phases

JSF Phases

Restore View

Apply Request Values

Process Validations

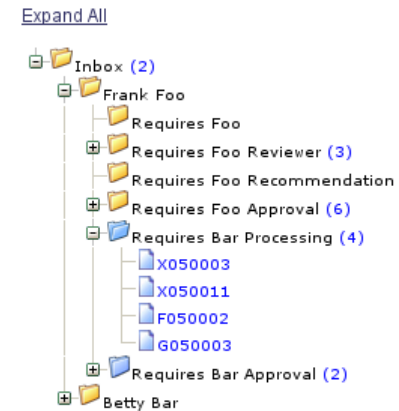
Process Updates

Invoke Application

Render Response

MyFaces(.apache.org)

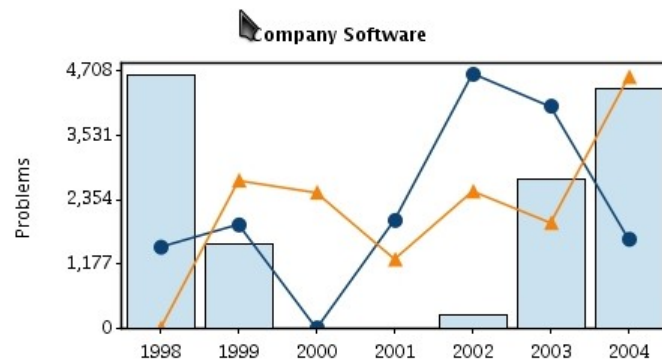
- Mají šikovnou vlastnost, kdy normální HTML tagy jsou doplněny o JSF renderery. Designer tedy vyrobí čistě HTML stránky, kterým programátor pouze přidá atribut, čím se budou renderovat. Soubor se tedy zobrazí staticky jako standardní stránka, ale dynamicky ji JSF nakreslí ve své režii.
- Knihovnu Trinidad (subset myfaces) používáme i my.
- <http://example.irian.at/myfacesexamples/home.jsf>



IceFaces(.org)



- MPL&commercial (just from \$500/year!)
- ICEsoft Technologies Inc.
- Implementací standardních komponent to nekončí, knihovny nabízí množství dalších, pokročilých komponent (rich-text editory, stromy, grafy ap.)



Primefaces

- poměrně nová knihovna

- viz demo na

<http://www.primefaces.org/showcase/ui/home.jsf>

PRIMEFACES

Change Theme: blitzer

UI Components TouchFaces FacesTrace

MOCK OS X ImageCropper
Ajax Engine ImageSwitch
Ajax Poll Inplace
Ajax Status InputText
Accordion InputTextarea
AutoComplete Keyboard
BreadCrumb Layout
Button LightBox
Calendar Masked Input
Captcha Media
Carousel Menu
Charts Menubar
CommandButton MenuButton
CommandLink Messages
ContextMenu NotificationBar
Collector OutputPanel
Color Picker Panel
ConfirmDialog Password
Dashboard PickList
Data Exporter Printer
DataGrid ProgressBar
DataList RequestContext
DataTable Resizable
Dialog RemoteCommand

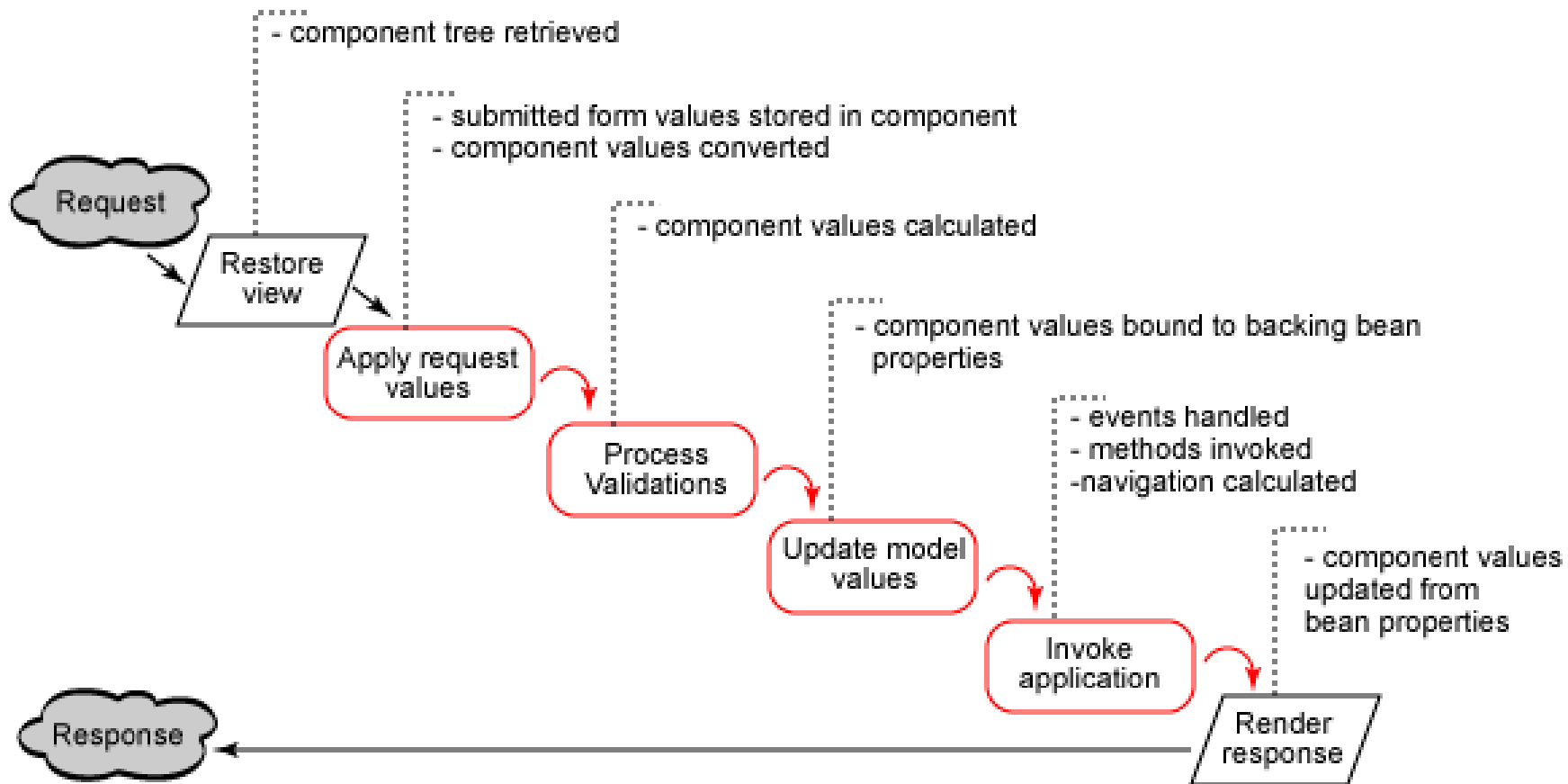
Data Exporter




Model	Year	Manufacturer	Color
96b3645c	1999	Volvo	Blue
3778e012	1973	Chrysler	Green
c823c909	2003	Ford	White
dfe7c88c	1971	Opel	Green
9c7ee722	1973	Volkswagen	Maroon
8db13b49	1979	Volkswagen	Silver
d19e7712	2000	Opel	Brown
cd3294dc	1974	BMW	Brown
dda86f3e	1966	Volvo	Green
e6e5742a	1979	BMW	Red

Export All Data Export Page Data

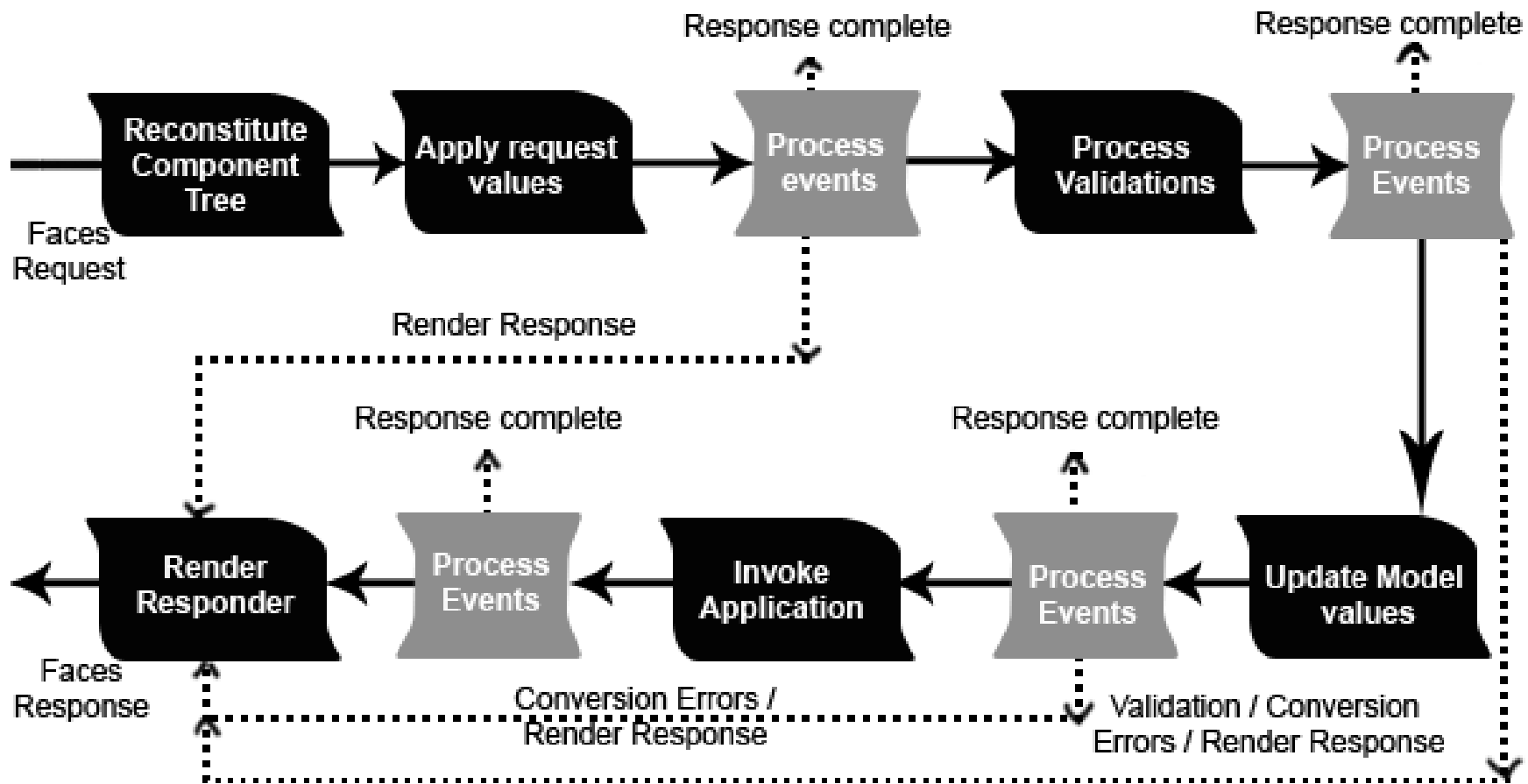
PDF CSV XML PDF CSV XML

Lifecycle - comments



	= System-level phases	Legend
	= Application-level phases	
	= Process events	
	- FacesContext.renderResponse () advances to Render Response phase	
	- FacesContext.responseComplete () ends JSF lifecycle	

Lifecycle – exceptions



Value change listeners

- explicitní informace, že se změnila hodnota
- `<h:inputText ... valueChangeListener="..."/>`
- neprovádí se okamžitě, ale až při zpracování formuláře
- pokud chcete okamžitou akci, je potřeba JavaScript (`onchange="submit()"`)

Templates I

- <http://x86.sun.com/thread.jspa?messageID=3792304>
- Is there any template system or layout manager for JSF like tiles?
 - Why not just use Tiles with JSF ?
 - The best solutions available is Facelets :-)
 - try this: tapestry
 - Or this: <http://shale.apache.org/shale-clay/index.html>
 - :) Tapestry's great and all... but the question was how to do layout in JSF.

Templates II

- podpora pomocí f:subview:

```
<f:view>
```

```
  <%@include file="templheader.jsp" %>
```

- v souboru templheader.jsp

```
<f:subview id="abc">
```

```
  ...
```

...ony budou ty facelety přeci jenom lepší

Templates III – Facelets

- Ježíš, zase další framework?

Templates III – Facelets

- Ježíš, zase další framework?
- Co znamená použití facelets:
 - faces-config.xml

```
<application>  
  <view-handler>  
    com.sun.facelets.FaceletViewHandler  
  </view-handler>  
</application>
```
- <http://www.jsfcentral.com/listings/FL300?link>

Templates IV – Facelets šablona

```
<?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://java.sun.com/jsf/facelets">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Facelets - Template Example</title>
  </head>
  <body>
    <h1><ui:insert name="title">Default
      Title</ui:insert></h1>
    <p><ui:insert name="body">Default
      Body</ui:insert></p>
  </body>
</html>
```

Templates IV – Facelets použití

```
<?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://java.sun.com/jsf/facelets"  
      xmlns:h="http://java.sun.com/jsf/html">
```

```
<body>
```

This text above will not be displayed.

```
<ui:composition template="/template.xhtml">
```

This text will not be displayed.

```
<ui:define name="title">Facelets</ui:define>
```

This text will also not be displayed.

```
<ui:define name="body">Hello from the Facelets  
client template!</ui:define>
```

This text will not be displayed.

```
</ui:composition>
```

This text below will also not be displayed.

```
</body>
```

```
</html>
```

Facelets – jsfc

- Facelety podporují konverzi jednoho tagu na jiný při zpracování:

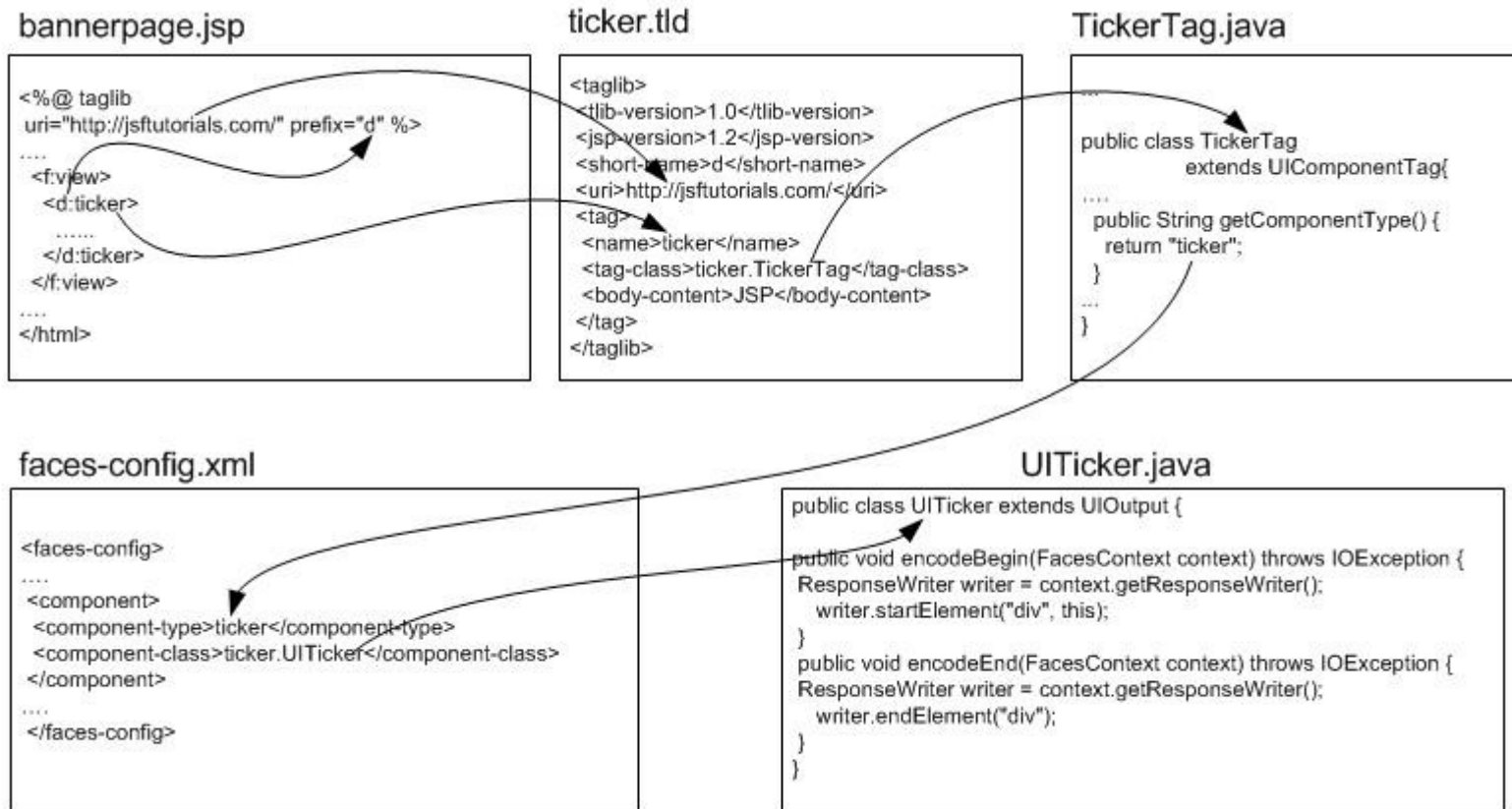
```
<!DOCTYPE html PUBLIC ...>  
<html xmlns="http://www.w3.org/1999/xhtml"  
      xmlns:h="http://java.sun.com/jsf/html">  
<body>  
  <input type="text" jsfc="h:inputText"  
  value="# {hello.world}" />  
</body>  
</html>
```


Vlastní custom componenta

- Mít vlastní komponenty je samozřejmě chladná vlastnost!
- Co je k tomu potřeba:
 - UIComponent/Tag Class
 - Renderer Class
 - Tag Library Description (.tld)
 - faces-config.xml Entries
 - Facelets .taglib.xml File

Komponenty

- provázání, definice jedné komponenty



UI Component class

- definiuje properties komponenty

pro JSF1.2

```
public class MyTag extends UIComponent &lt;UITag> {
```

```
    String style; + get/set
```

```
    public void release() {
```

```
        super.release();
```

```
        style = null ;
```

```
    }
```

```
    protected void setProperties(UIComponent component) {
```

```
        super.setProperties(component);
```

```
        if(style != null)
```

```
            component.getAttributes().put("style", style);
```

```
    }
```

```
    public String getComponentType() {
```

```
        return "my";
```

```
    }
```

```
    public String getRendererType() {  
        // null means the component renders itself  
        return null;  
    }
```

Renderer Class

- hlavně implementuje `encodeBegin()` a `encodeEnd()` (často jen jednu)

```
public class UIMyComponent extends UIOutput {  
    public void encodeBegin(FacesContext context) throws IOException {  
        ResponseWriter writer = context.getResponseWriter();  
        writer.startElement("div", this);  
        String style = (String)getAttributes().get("style");  
        if (style!=null)  
            writer.writeAttribute("style", style, null);  
    }  
    public void encodeEnd(FacesContext context) throws IOException {  
        ResponseWriter writer = context.getResponseWriter();  
        writer.endElement("div");  
    }  
}
```

Tag Library Description (.tld)

```
1. <?xml version="1.0" encoding="UTF-8"?>
2.
3. <taglib xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
4.   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
jsptaglibrary_2_1.xsd"
5.   version="2.1">
6.   <description><![CDATA[Your description here]]></description>
7.   <tlib-version>1.0</tlib-version>
8.   <short-name>foo</short-name>
9.   <uri>http://foo.com/foo</uri>
10.  <tag>
11.    <description><![CDATA[Your description here]]></description>
12.    <name>foo</name>
13.    <tag-class>com.foo.Foo</tag-class>
14.    <body-content>JSP</body-content>
15.    <attribute>
16.      <description><![CDATA[Your description here]]></description>
17.      <name>id</name>
18.      <required>>false</required>
19.      <rtexprvalue>>true</rtexprvalue>
20.    </attribute>
21.    <attribute>
22.      <description><![CDATA[Your description here]]></description>
23.      <name>bar</name>
24.      <required>>true</required>
25.      <deferred-value><type>java.lang.Object</type></deferred-value>
26.    </attribute>
27. </taglib>
```

faces-config.xml

```
<?xml version="1.0"?>
<faces-config
  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"
  version="1.2">
  <component>
    <description><![CDATA[Your description here]]></description>
    <display-name>Foo</display-name>
    <component-type>com.foo.Foo</component-type>
    <component-class>com.foo.component.Foo</component-class>
    <component-extension>
      <renderer-type>com.foo.FooRenderer</renderer-type>
    </component-extension>
  </component>
  <render-kit>
    <description>Renderkit implementation for the Download
component</description>
    <renderer>
      <component-family>com.foo.Foo</component-family>
      <renderer-type>com.foo.FooRenderer</renderer-type>
      <renderer-class>com.foo.render.FooRenderer</renderer-class>
    </renderer>
  </render-kit>
```

Facelets .taglib.xml File

- Optional, but recommended

1. `<?xml version="1.0"?>`
2. `<!DOCTYPE facelet-taglib PUBLIC`
3. `"-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"`
4. `"http://java.sun.com/dtd/facelet-taglib_1_0.dtd">`
- 5.
6. `<facelet-taglib>`
7. `<namespace>http://foo.com/foo</namespace>`
8. `<tag>`
9. `<tag-name>foo</tag-name>`
10. `<component>`
11. `<component-type>com.foo.Foo</component-type>`
12. `<renderer-type>com.foo.component.Foo</renderer-type>`
13. `</component>`
14. `</tag>`
15. `</facelet-taglib>`

Web Fragments (1)

- web.xml lze rozdělit na více částí
 - web-fragment.xml

```
<web-fragment>
```

```
  <servlet>
```

```
    <servlet-name>...</servlet-name>
```

```
    <servlet-class>...</servlet-class>
```

```
  </servlet>
```

```
</web-fragment>
```


Web Fragments (2)

- Web fragments dovoluje pořadí fragmentů.
- Zpracování fragmentů lze i potlačit (<metadata-complete/>).
- Konečně se frameworky mohou samy registrovat!
- Webové frameworky nepotřebují definovat front controller, definují si ho samy.

Shared Framework Pluggability

- Kontejner prochází třídy a vyhledává všechny implementace rozhraní `ServletContainerInitializer`.
- Knihovny se tak mohou registrovat i programově.

```
@HandlesTypes(AnnotationA.class)
```

```
AServletContainerInitializer implements ServletContainerInitializer {
```

```
    public void onStartup(Set<Class<A>>c, ServletContext ctx)
```

```
        throws ServletException {
```

```
        // Framework-specific code here to initialize the runtime
```

```
        // and setup the mapping etc.
```

```
        ServletRegistration reg = ctx.addServlet("AServlet",  
                                             "com.f.AServlet");
```

```
        reg.addServletMapping("/foo");
```

Composite Components

- Často je potřeba custom component, ale není třeba speciální vykreslování, jen složení již existujících.
- JSF 2.0 podporuje tzv. composite component, která přesně toto definuje. Příkladem budiž label a edit společně s chybovou (validační) hláškou.
- V komponentě se definují akce, které jsou posléze využívány pomocí listenerů.

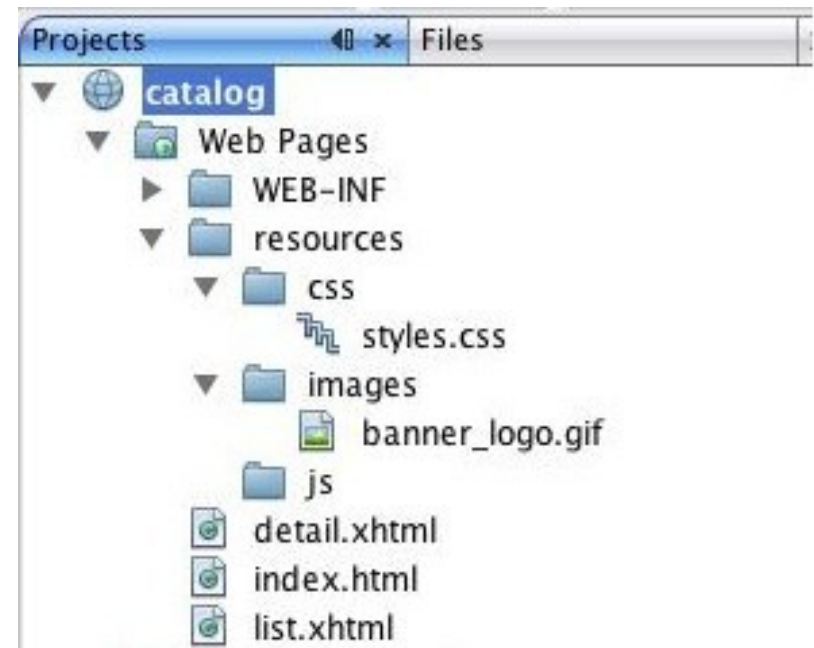
```
<h:body>
  <composite:interface>
    <composite:actionSource name="loginEvent"/>
  </composite:interface>
  <composite:implementation>
    <p>Username:<h:inputText id="username" /></p>
```

```
<ez:loginPanel>
  <f:actionListener
    for="loginEvent"
    type="LoginListener"
  />
```

loginPanel.jsp

Resource Handling

- Zdroje jako css, obrázky, Javascript jsou nyní standardně umístěny v adresáři resources.
- Existuje k nim i nový interface.
- Pro aplikaci to nemá žádnou výraznou výhodu, ale pro knihovny je to výrazný pokrok.



Advanced Navigation

- pokud není nalezeno odpovídající pravidlo, zkusí se použít outcome jako view-id (tedy lze přímo odkazovat stránky a není třeba dělat navigation rules)
- podmíněná navigace

```
<navigation-case>  
  <from-outcome>success</from-outcome>  
  <if>#{foo.someCondition}</if>  
</navigation-case>
```

Bookmarkable Pages

- Všechny JSF stránky využívají POST, tedy nedají se vložit do bookmarků. U některých aplikací je to poměrně velké omezení (např. knihovna).

- Podpora GET

```
<f:metadata>
```

```
    <f:viewParam name="foo" value="bean.attr" />
```

```
    <f:event type="preRenderView"
```

```
        listener="# {bean.prepareData}">
```

```
</f:metadata>
```

- podporuje zpracování **příchozích atributů** do EL

- ...a jejich zpracování **Javovým kódem**

...a ještě odkazy na takové stránky

- `<h:link outcome="success">`
 `<f:param name="foo" value="param" />`
 `</h:link>`
- dtto `h:button`

Knížky a linky

- Chris Schalk, Ed Burns: **JSF: The Complete Reference**

- Kito D. Mann: **JavaServer Faces in Action**

- RichFaces DevGuide:

<http://www.jboss.org/file-access/default/members/j>

- Tag Library Documentation Generator

http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/tl

- Facelets fits JSF like a glove

<http://www.ibm.com/developerworks/java/library/j-facelets/>

Links

- <http://java.sun.com/javaee/5/docs/tutorial/doc/bnaph.html>
 - JSF v JEE tutorialu
- <http://jsfcentral.com>
- <http://www.coreservlets.com/JSF-Tutorial/>
- <https://facelets.dev.java.net/nonav/docs/dev/docbook.html>
- <http://www.netbeans.org/kb/docs/web/quickstart-facelets-in->
- <http://www.icefaces.org/main/resources/tutorials.iframe>
- <http://balusc.blogspot.com/2006/06/using-datatables.html>
- <http://exadel.com/tutorial/jsf/jsftags-guide.html>

Závěr

- Nejlepší cesta (když teď chápete základy)
 - projděte si nějaký tutoriál
 - zkuste si svoji aplikaci (začněte na semestrálce)
- Hodně úspěchů ve vytváření graficky přitažlivých a funkčně bohatých aplikací!