12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

J13
Extreme Rich Clients

Ajax Applications with JFS 2 and the New RichFaces 4

Max Katz

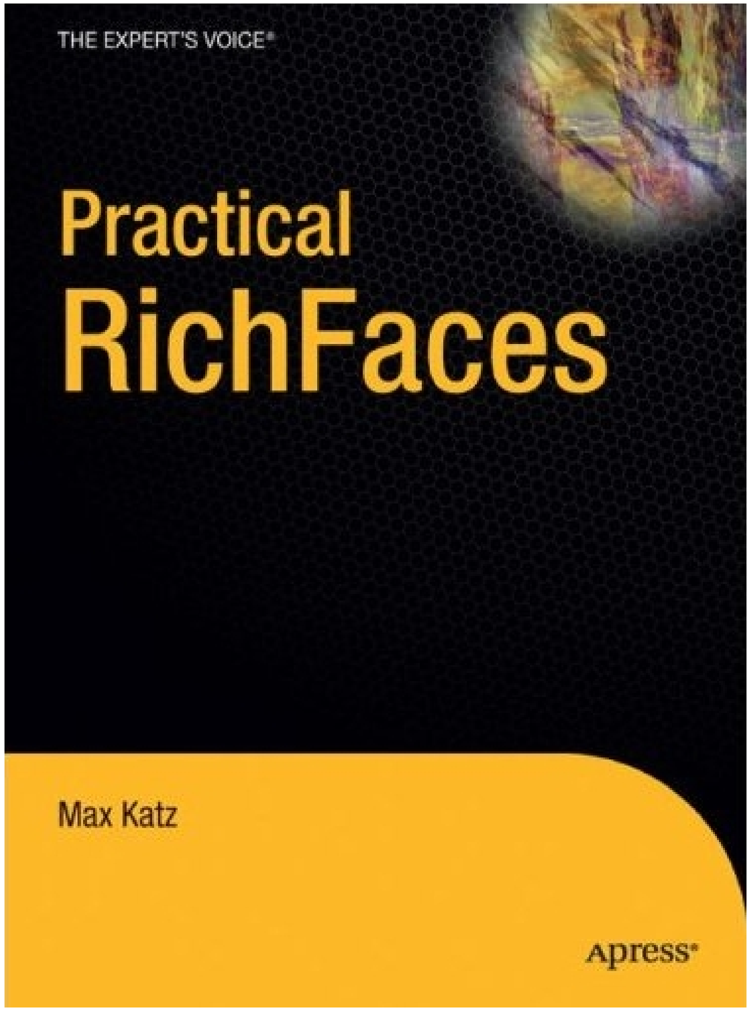
Ajax Applications with JSF 2 and New RichFaces 4

Herbstcampus
September 13th, 2010

Max Katz
Exadel

Who is this guy?

- Senior Systems Engineer, RIA strategist at Exadel
  - http://mkblog.exadel.com
  - http://twitter.com/maxkatz
- JSF/RichFaces consulting, training
- Leads a number of projects:
  - Exadel Tiggr
  - Exadel Flamingo
  - Exadel Fiji
  - Exadel JavaFX Plug-in for Eclipse



THE EXPERT'S VOICE®

Practical RichFaces

Max Katz

Apress®

#44

DZone Refcardz

Exadel

JBoss RichFaces

By Nick Belaevski, Ilya Shaikovsky, Jay Balunas, and Max Katz

Get More Refcardz! Visit refcardz.com

www.dzone.com

JBoss RichFaces

Author of Practical RichFaces (Apress)

Co-author of RichFaces DZone Refcard

Exadel

- Products and services company
- Founded in 1998, headquarters in Concord, CA
- 350+ employees

City	Country	Year
Concord	California, USA	1998
Moscow	Russia	1999
Minsk	Belarus	2002
Vitebsk	Belarus	2005
Donetsk, Kharkov	Ukraine	2006

Products

- Open Source with JBoss
  - RichFaces
  - JBoss Tools/JBoss Developer Studio
- Tiggr – create and share mockups online
- Flamingo
- Fiji (JSF – JavaFX/Flex integration)
- jsf4birt (JSF – BIRT/Actuate integration)
- JavaFX Plug-in for Eclipse

Services

- Rich enterprise application development
- Eclipse development
- Custom rich component development
- Mobile development
- Training
- Most projects are done in Eastern Europe

The Plan

- Ajax features in JSF 2
- The new RichFaces 4

JSF 2

- JSF 2 is a <u>major</u> upgrade over JSF 1.x
- Many features, ideas taken from projects such as Seam and RichFaces, and others

JSF 2 new features

- Facelets
- Composite components
- Navigation
- GET support
  - Page parameters
  - h:link, h:button
- Resource loading
- New scopes
  - Flash, View, custom
- Configuration
  - Annotations
- Bean Validation support
- Ajax

JSF 2 <f:ajax>

- Basic Ajax functionality
- Greaty inspired by RichFaces 3 <a4j:support> tag
- Ajax in JSF in 3 easy steps:
  - How to send an Ajax request
  - Partial view processing
  - Partial view rendering

Event based on which to fire the Ajax request

What to **execute** on the server:
- @all
- @this (default)
- @form
- @none
- Id's
- EL

```
<h:inputText>
  <f:ajax event="keyup"
        execute="@form"
        render="id1 id2"
        listener="#{bean.ajaxListener}"
        onevent="someFunction();"/>
<h:inputText>
```

Optional server listener

Partial view **rendering**:
- @all
- @this
- @form
- @none (default)
- Id's
- EL

Optional name of JavaScript function to execute during Ajax request. Invoked 3 times, at: *begin*, *success, complete* steps

Using <f:ajax>

```
<h:commandButton value="Submit" action="#{bean.action}" >
   <f:ajax event="focus" execute="@form" render="output"/>
</h:commandButton>
<h:panelGrid id="output">
 ...
</h:panelGrid>
```

Attaching to button, specifying event

```
<h:commandButton value="Submit" action="#{bean.action}" >
   <f:ajax execute="@form" render="output output2"/>
</h:commandButton>
<h:panelGrid id="output">
 ...
</h:panelGrid>
<h:panelGrid id="output2">
 ...
</h:panelGrid>
```

No event specified, using default event

Using <f:ajax>

Default event is *onchange*

```
<h:inputText value="#{bean.text}" >
   <f:ajax event="keyup" render="text"/>
</h:inputText>
<h:outputText id="text" value="#{bean.text}" />
```

No event specified so using default

```
<h:selectOneListbox id="list" value="#{bean.choice}" >
   <f:selectItems value="#{cean.choiceList}" />
   <f:ajax render="info" listener="#{bean.change}"/>
</h:selectOneListbox>

<h:panelGrid id="info">
 ...
</h:panelGrid>
```

```
<f:ajax>
  <h:panelGrid>                         No default event, no Ajax added
    <h:selectBooleanCheckbox>                                  onchange
    <h:inputText>                                  onchange
    <h:commandButton>                                  onclick
  </h:panelGrid>
</f:ajax>
```

```
<f:ajax event="click">
  <h:panelGrid>                                  onclick
    <h:selectBooleanCheckbox>                                  onclick
    <h:inputText>                                  onclick
    <h:commandButton>                       onclick and onfocus
     <f:ajax event="focus"/>
    </h:commanButton>
  </h:panelGrid>
</f:ajax>
```

```
<f:ajax event="valueChange">
  <h:panelGrid>                              No Ajax added
    <h:selectBooleanCheckbox>                                  onchange
    <h:inputText>                                  onchange
    <h:commandButton>                                  onfocus
     <f:ajax event="focus"/>
    </h:commanButton>
  </h:panelGrid>
</f:ajax>
```

RichFaces 4 – rich JSF framework

- JSF 2 based
- Ajax components
  - a4j:* tag library (core)
  - rich:* tag library (UI)
- Skins and themes
- CDK – Component Development Kit

RichFaces versions

Version	JSF 1.1	JSF 1.2	JSF 2
RichFaces 3.1.x	●		
RichFaces 3.3.3*		●	●
RichFaces 4			●

* Note: RichFaces 3.3.3 only has basic JSF 2 support

What about deployment?

- All servers
- All browsers
    - (Event IE 6.0)

RichFaces history

- 2004: started by Alexander Smirnov
- 2004-2007: Ajax4jsf – free, open source
  RichFaces – commercial
  Exadel
- 2007: JBoss takes over

  Exadel team continues to develop the framework

Just tell me when RichFaces 4 is going to be released?

RichFaces 4

JavaScript in RichFaces is now entirely based on the popular jQuery library.

RichFaces 4

- All components are reviewed for consistency, usability
- Redesigned following semantic HTML principles
- Server-side and client-side performance optimization
- Strict code clean-up and review

RichFaces 4

- New and easy to use CDK (Component Development Kit)
- Quickly build your own custom rich components

Google App Engine

- Deploy RichFaces application in Google App Engine (GAE)
- Special maven-based plug-in available

RichFaces <a4j:ajax>

- 100% based on standard <f:ajax>
- Just replace f: with a4j: and get exactly the same functionality
- But, you get extra features...

<a4j:ajax> attributes

Attribute	Description
onbegin	JavaScript to execute before Ajax request
onbeforedomupdate	JavaScript to execute after response comes back but before DOM update
oncomplete	JavaScript to execute after DOM update
bypassUpdates	Skips Update Model and Invoke Application phases, useful for form validation
limitRender	Skips all a4j:outputPanel ajaxRender=”true” areas. Only renders what is set in current render
status	Status to display during Ajax request
focus	Sets focus on component after Ajax request

RichFaces 4

That's not all, there are more RichFaces goodies...

RichFaces 4 core action

- a4j:ajax
- a4j:commandButton
- a4j:commandLink
- a4j:jsFunction
- a4j:poll
- a4j:push

<a4j:commandButton> – Ajax button

```
<h:commandButton value="Save" action="#{bean.action}">
   <f:ajax execute="@form"
           render="output"/>
</h:commandButton>
```

```
<a4j:commandButton value="Save"
     execute="@form"
     render="output"
     action="#{bean.action}" />
```

<a4j:jsFunction> – fire Ajax request from any JavaScript function

```
<table>
   ...
   <td onmouseover="update('yellow')"/>
   ...
</table>
<a4j:jsFunction name="update"
                action="#{bean.change}"
                reRender="panel">
  <a4j:param value="param1" assignTo="#{bean.color}"/>
</a4j:jsFunction>
```

<a4j:poll> – periodically send an Ajax request

```
<a4j:poll interval="1000"
          action="#{bean.count}"
          render="output"
          enabled="#{bean.pollEnabled}" />

<h:panelGrid id="output">
...
</h:panelGrid>
```

RichFace 4 core

- a4j:outputPanel
- a4j:status
- a4j:region
- a4j:queue
- a4j:repeat
- a4j:log

<a4j:outputPanel> – auto rendered panel

```
<h:selectOneMenu value="#{bean.fruit}">
   <a4j:ajax listener="#{bean.change}"/>
</<h:selectOneMenu>
<a4j:outputPanel ajaxRendered="true">
   <h:panelGrid>
      ...
   </h:panelGrid>
   <h:panelGrid>
      ...
   </h:panelGrid>
</a4j:outputPanel>
```

Rendered on every Ajax request

<a4j:status> – Ajax request status

```
<a4j:status startText="Loading..."/>

<a4j:status name="ajaxSpecial">
  <f:facet name="start">
    <h:graphicImage value="ajaxStatus.jpg"/>
  </f:facet>
</a4j:status>

<h:form>
   <a4j:commandButton />
   <a4j:commandButton status="ajaxSpecial"/>
</h:form>
```

<a4j:region> – declaratively define execute region

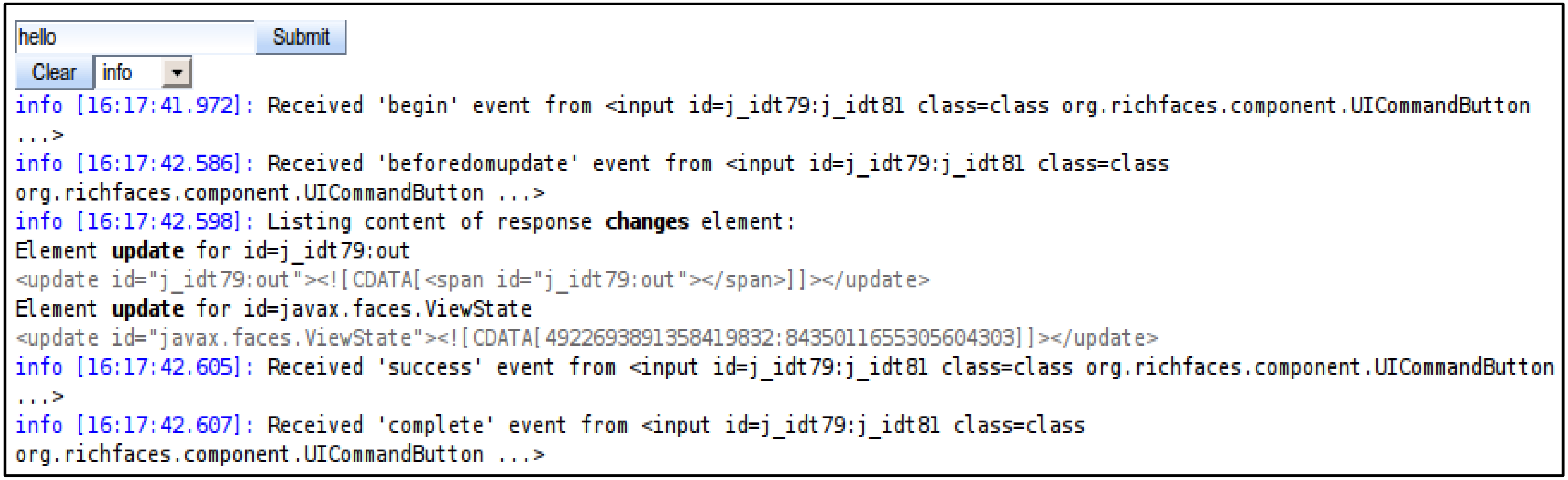
```
<h:form>
  <a4j:region>
     <h:inputText />
     <h:inputText />
     <a4j:commandButton execute="@region"/>
  <a4j:region>
</h:form>
```

Execute options:
- @all
- @this (default)
- @form
- @none
- Id's
- EL
- **@region**

```
<h:form>
  <a4j:region>
     <h:inputText />
     <h:inputText />
     <a4j:commandButton />
  <a4j:region>
</h:form>
```

<a4j:log> – Ajax request information

- Levels:
  - debug, info, warn, error



```
info [16:17:41.972]: Received 'begin' event from <input id=j_idt79:j_idt81 class=class org.richfaces.component.UICommandButton
...>
info [16:17:42.586]: Received 'beforedomupdate' event from <input id=j_idt79:j_idt81 class=class
org.richfaces.component.UICommandButton ...>
info [16:17:42.598]: Listing content of response changes element:
Element update for id=j_idt79:out
<update id="j_idt79:out"><![CDATA[<span id="j_idt79:out"></span>]]></update>
Element update for id=javax.faces.ViewState
<update id="javax.faces.ViewState"><![CDATA[4922693891358419832:8435011655305604303]]></update>
info [16:17:42.605]: Received 'success' event from <input id=j_idt79:j_idt81 class=class org.richfaces.component.UICommandButton
...>
info [16:17:42.607]: Received 'complete' event from <input id=j_idt79:j_idt81 class=class
org.richfaces.component.UICommandButton ...>
```

JavaScript interactions

```
<a4j:commandLink value="Link"
   onbegin="alert('Link clicked')"
   onbeforedomupdate="alert('Response received')"
   oncomplete="alert('DOM updated')">
</a4j:commandLink>
```

Easier to use than the standard `onevent`

Advanced rendering option

```
<a4j:commandButton render="output"/>
<a4j:commandButton render="output" limitRender="true"/>

<h:panelGrid id="output">
...
</h:panelGrid>

<a4j:outputPanel ajaxRendered="true">
...
</a4j:outputPanel>
```

Turns off all auto rendered panels, only renders what is set in current render

Skipping phases when validating

```
<h:inputText id="name" value="#{bean.name}"/>
   <a4j:ajax event="blur" bypassUpdates="true"/>
</h:inputText>
<rich:message for="name"/>
```

1. **Restore View**
2. **Apply Request Values**
3. **Process Validation**
4. Update Model
5. Invoke Application
6. **Render Response**

Skipped

JSF 2 queue

- JSF 2 has very basic queue functionality
- Events are queued and fired one at a time
    - Only one request is processed on the server at a time

RichFaces queue upgrades

- Delay firing of a request
- Combine requests from one or more controls
- Cancel DOM updates if the same request was fired
- Define queue as:
  - Global (all views have queue)
  - View-based
  - Form-based
  - Named (used by particular components only)

RichFaces <a4j:queue>

```
<a4j:queue requestDelay="2000"/>
...
<a4j:commandButton value="Button1"/>
<a4j:commandButton value="Button2"/>
```

```
<a4j:queue requestDelay="2000"/>
...
<a4j:commandButton>
   <a4j:attachQueue requestDelay="1000"/>
</a4j:commmandButton>
<a4j:commandButton />
```

a4j:queue – “combining” events

```
<a4j:queue requestDelay="2000"/>
...
<a4j:commandButton>
   <a4j:attachQueue requestGroupingId="mainGroup"/>
</a4j:commmandButton>
<a4j:commandButton>
   <a4j:attachQueue requestGroupingId="mainGroup"/>
</a4j:commmandButton>
```

a4j:queue – ignoring “stale” responses

```
<a4j:queue requestDelay="2000
           ingoreDupResponses="true"/>

<h:inputText value="#{bean.state}">
   <a4j:ajax event="keyup" listener="#{bean.load}"
             render="states"/>
</h:inputText>

<h:dataTable id="states">
  <rich:column/>
</h:dataTable>
```

RichFaces UI components

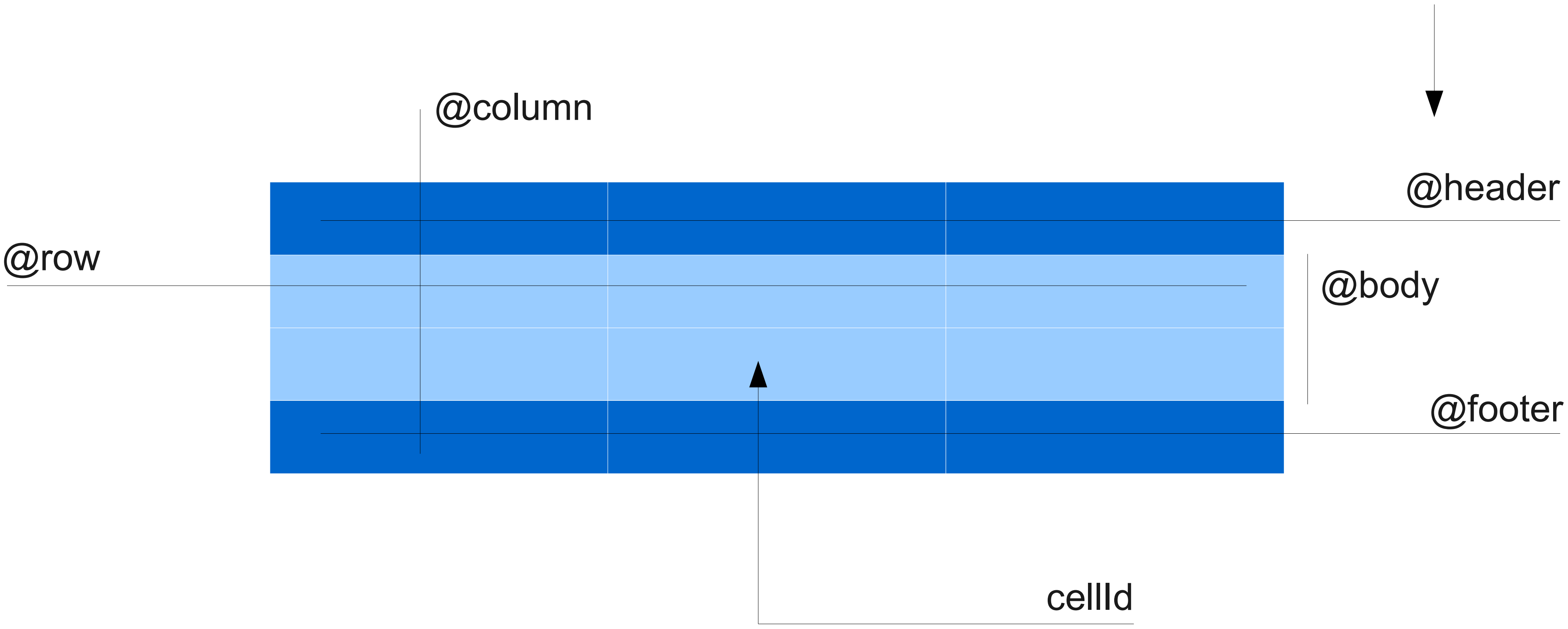
- Data iteration
- Output, panels
- Input
- Menu
- Trees
- Selects
- Layout
- Client side validation
- Miscellaneous

Rich data iteration

- a4j:repeat
- rich:dataTable
- rich:extendedDataTable
- rich:subTable
  - rich:subTableToggleControl
- rich:list
- rich:dataGrid
- rich:dataScroller
- rich:column
- Column and row spanning
- Filtering, sorting

Partial update

Outside the table:
- tableId@header
- tableId@body
- tableId@footer



Partial update

- Cell from within current row:

```
render="cellId"
```

- Row(s) from outside the table:

```
render="tableId:rowKey"
render="tableId:#{bean.rowKeySet}"
```

- Cell(s) from outside the table:

```
render="tableId:rowKey:cellId"
render="tableId:#{bean.rowKeySet}:cellId"
```

Rich output, panels

- rich:panel
- rich:togglePanel
- rich:accordion
- rich:popupPanel
- rich:tabPanel
- rich:panelBar
- rich:panelMenu
- rich:collapsiblePanel
- rich:message(s)
- rich:paint2D
- rich:separator
- rich:toolBar
- rich:toolTip

Rich input

- rich:autocomplete
- rich:inputNumberSlider
- rich:inputNumberSpinner
- rich:inplaceInput
- rich:calendar
- rich:colorPicker
- rich:editor
- rich:fileUpload

Rich menu

- rich:contextMenu
- rich:dropDownMenu

Rich tree

- rich:tree
- rich:treeNode
- Listeners

Rich selects

- rich:orderingList
- rich:pickList
- rich:listShuttle
- rich:selectBox
- rich:inplaceSelect

Rich layout

- rich:page
- rich:layout
    - rich:layoutPanel
- rich:splitter

Bean Validation (JSR 303)

- JSF 2 has support for Bean Validation (on the server)

```
public class Bean {
  @Email
  private String email;
}
```

```
<h:inputText id="email" value="#{bean.email}">
   <a4j:ajax event="blur"/>
</h:inputText>
<rich:message for="email"/>
```

RichFaces client validation

```
Public class Bean {
  @Email
  private String email;
}
```

```
<h:inputText id="email" value="#{bean.email}">
   <rich:clientValidator event="blur"/>
</h:inputText>
<rich:message for="email"/>
```

RichFaces client validation

```
<h:form>
   <rich:clientValidator />
   <h:inputText/>
   <h:inputText/>
   <h:inputText>
       <rich:clientValidator event="blur"/>
   </h:inputText>
   <h:inputText>
      <rich:clientValidator disabled="true"/>
   </h:inputText>
</h:form>
```

RichFaces functions

- *rich:clientId(id)* - returns component client id
- *rich:element(id)* - returns DOM element
- *rich:component(id)* - returns RichFaces client component instance to call JS API method
- *rich:isUserInRole(role)* - returns if the user has specified role
- *rich:findComponent(id)* - returns component instance for given short id

rich:component(id) function

- Allows to call JS API on a component

```
<input type="button"
       onclick="#{rich:component('popup')}.show();"
      value="Open" />

<rich:popupPanel id="popup">
   <h:outputLink value="#"
         onclick="#{rich:component('popup')}.hide();
              return false;">
             <h:outputText value="Close"/>
    </h:outputLink>
</rich:popupPanel>
```

Rich miscellaneous

- rich:componentControl
- rich:jQuery
- rich:hotKey
- rich:gmap
- rich:virtualEarth

<rich:componentControl>

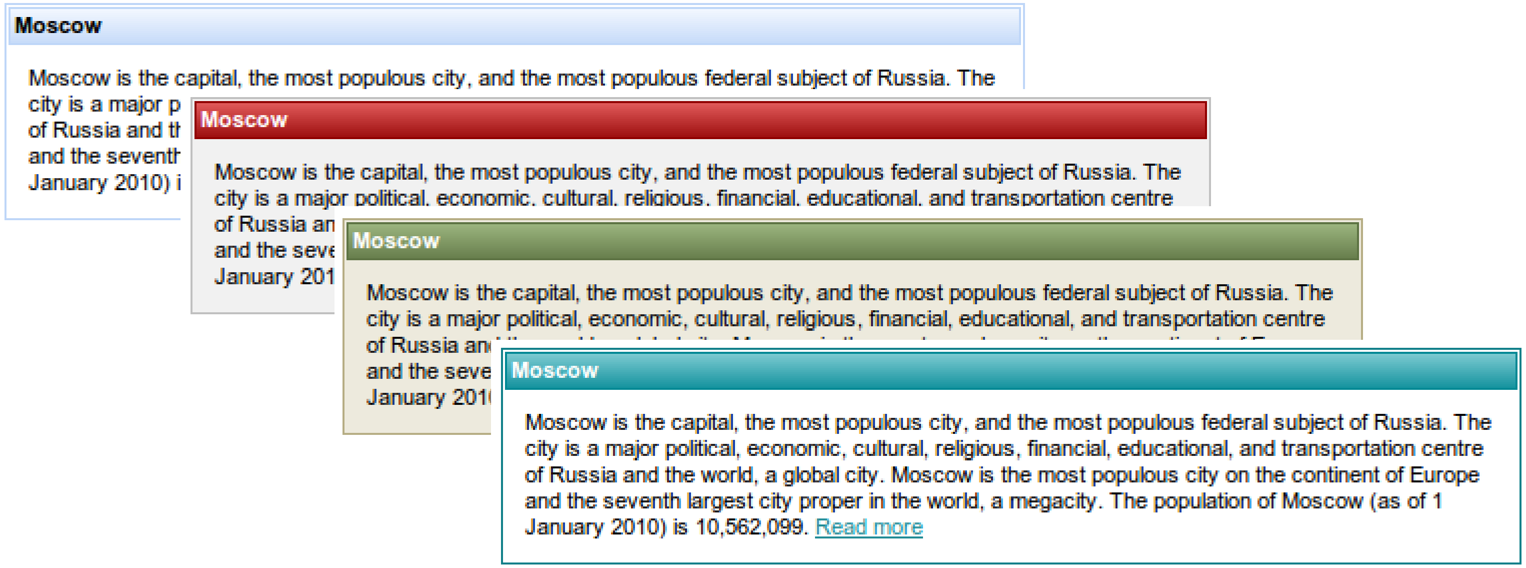
- Allows to call JS API on a component in declarative fashion

```
<h:outputLink id="openLink" value="#">
   <h:outputText value="Open" />
   <rich:componentControl event="click"
                           operation="show"
                           target="popup" />
</h:outputLink>

<rich:popupPanel id="popup">
...
</rich:popupPanel>
```

<rich:jQuery>

```
<input type="button" id="changeButton"
                      value="Change title" />
<rich:jQuery selector="#changeButton"
 query="click(function(){
   $('#panel #panel_header').text('Capital of Russia');
})"/>


<rich:panel header="Moscow" id="panel">
   Moscow is the capital, the most populous ...
</rich:panel>
```

Skins



Moscow

Moscow is the capital, the most populous city, and the most populous federal subject of Russia. The
city is a major p
of Russia and th
and the seventh
January 2010) i

Moscow

Moscow is the capital, the most populous city, and the most populous federal subject of Russia. The
city is a major political, economic, cultural, religious, financial, educational, and transportation centre
of Russia an
and the seve
January 201

Moscow

Moscow is the capital, the most populous city, and the most populous federal subject of Russia. The
city is a major political, economic, cultural, religious, financial, educational, and transportation centre
of Russia an
and the seve
January 201

Moscow

Moscow is the capital, the most populous city, and the most populous federal subject of Russia. The city is a major political, economic, cultural, religious, financial, educational, and transportation centre of Russia and the world, a global city. Moscow is the most populous city on the continent of Europe and the seventh largest city proper in the world, a megacity. The population of Moscow (as of 1 January 2010) is 10,562,099. Read more

Skins

- Lightweight extension on top of CSS
- Change look and feel of all Rich component with a few minor changes
- Can be applied to standard JSF and HTML tags as well

Ready-to-use skins

- classic
- emeraldTown
- blueSky
- ruby
- wine
- deepMarine
- plain
- japanCherry
- laguna
- glassX
- darkX

```
<context-param>
  <param-name>org.richfaces.skin</param-name>
  <param-value>ruby</param-value>
</context-param>
```

Skin file (properties file)

```
#Colors
headerBackgroundColor=#900000
headerGradientColor=#DF5858
headerTextColor=#FFFFFF
headerWeightFont=bold

generalBackgroundColor=#f1f1f1
generalTextColor=#000000
generalSizeFont=11px
generalFamilyFont=Arial, Verdana, sans-serif

controlTextColor=#000000
controlBackgroundColor=#ffffff
additionalBackgroundColor=#F9E4E4
```

Skins

- Modify existing or create your own

```
<context-param>
  <param-name>org.richfaces.skin</param-name>
  <param-value>myCustomSkin</param-value>
</context-param>
```

- Change skins in runtime

```
<context-param>
  <param-name>org.richfaces.skin</param-name>
  <param-value>#{bean.skin}</param-value>
</context-param>
```

Overwriting skins

```
<style>
 .rf-p-hr {
    // your custom style, applied to all panels on
    // on page
  }
 .panelHeader {
    // custom header style
  }
</style>

<rich:panel id="panel1">
...
</rich:panel id="panel2">
<rich:panel headerClass="panelHeader">
...
</rich:panel>
```

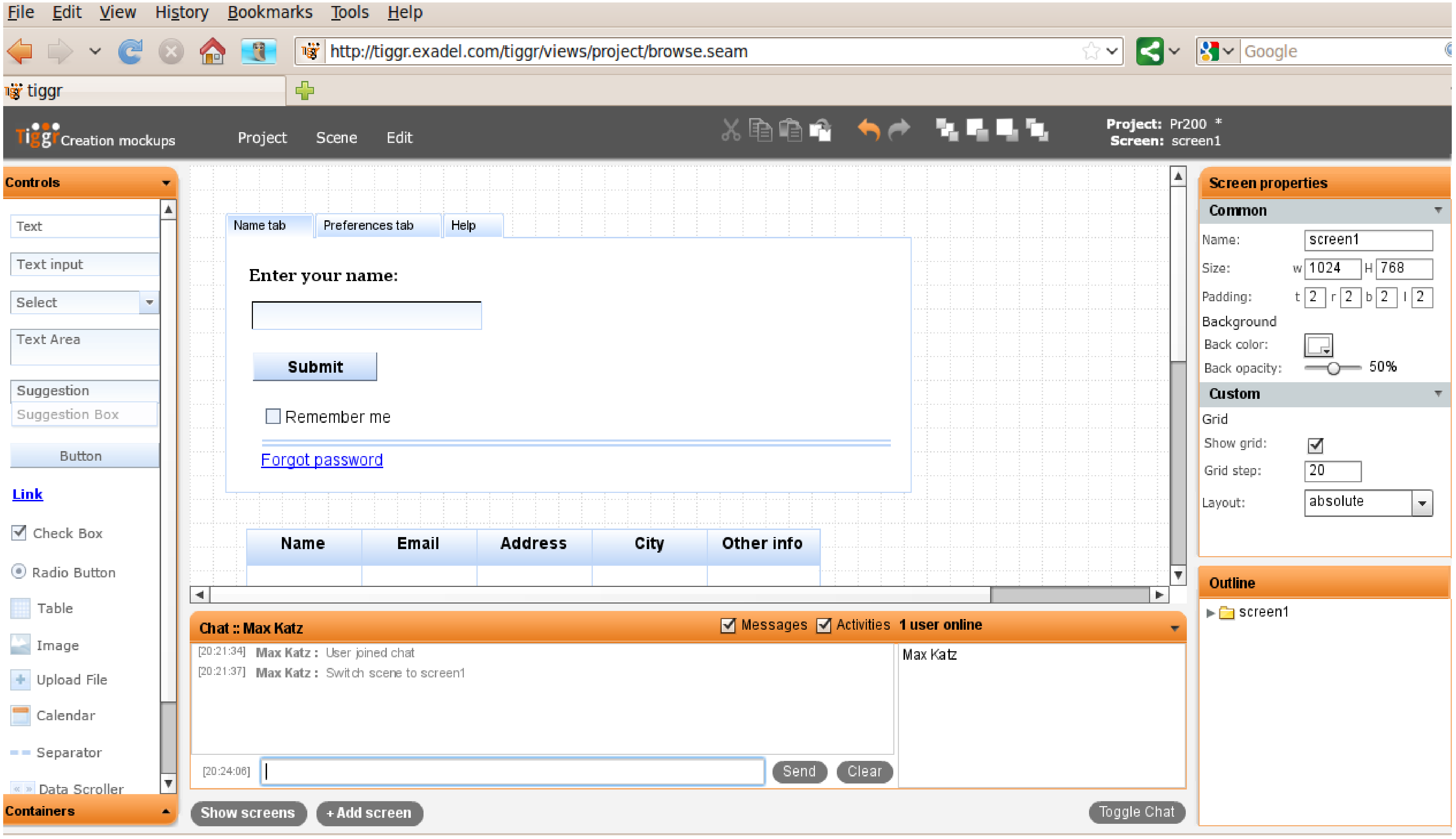
Exadel Tiggr

- Create, collaborate and share mockups online
- RichFaces palette
- Upcoming features
  - HTML generation
  - More widgets and controls
- Give it a try - http://tiggr.exadel.com

Exadel Tiggr – create, share and collobrate on application mockups

http://tiggr.exadel.com

Exadel Tiggr



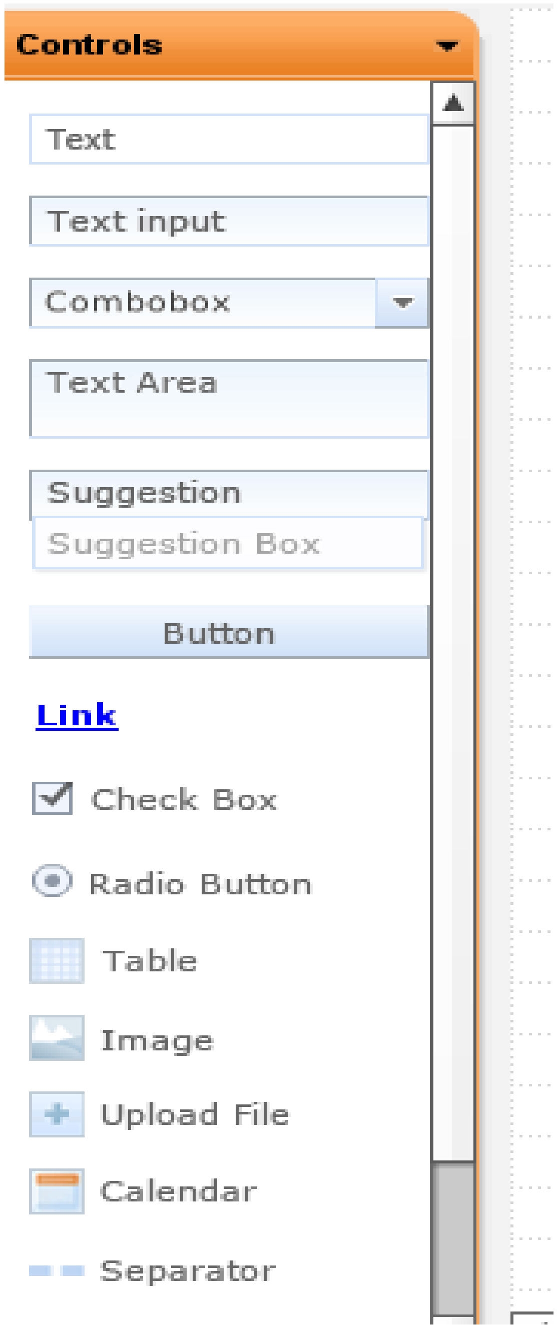
Name: screen1
Size: w 1024 H 768
Padding: t 2 r 2 b 2 l 2
Background
Back color:
Back opacity: 50%
Grid
Show grid: ✓
Grid step: 20
Layout: absolute

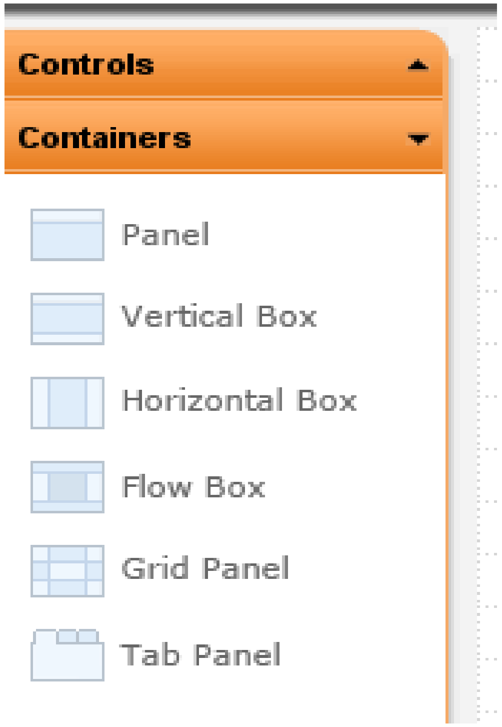
Chat :: Max Katz ✓ Messages ✓ Activities 1 user online

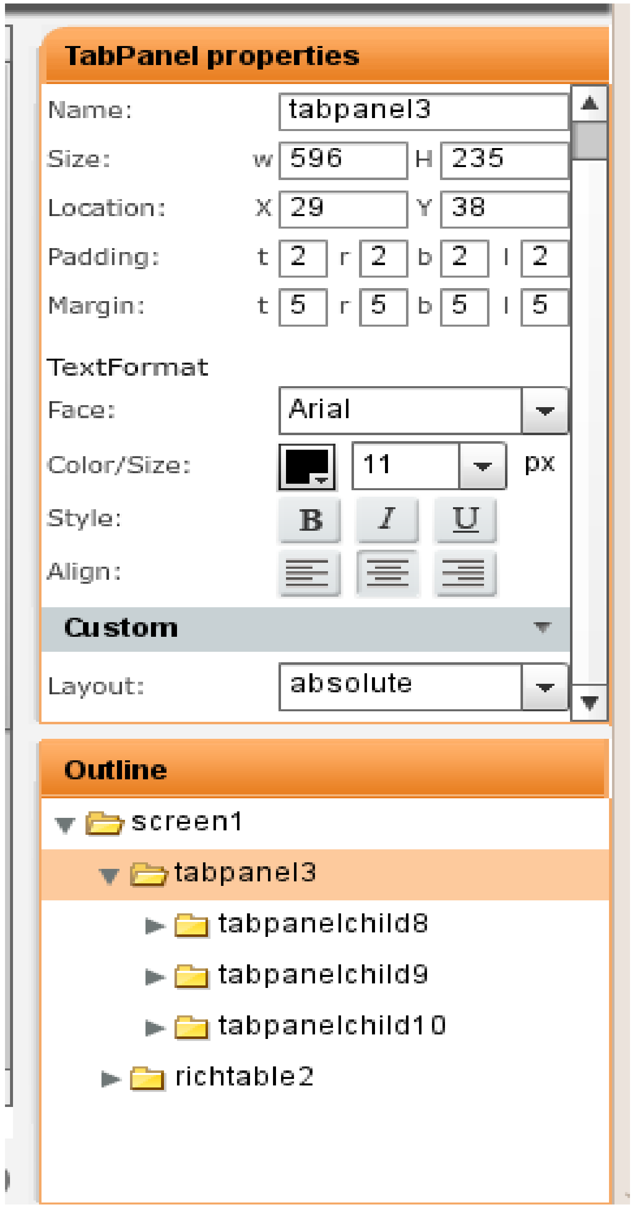
[20:21:34] Max Katz : User joined chat
[20:21:37] Max Katz : Switch scene to screen1

Max Katz

[20:24:06]






TabPanel properties

Name: tabpanel3

Size: W 596 H 235

Location: X 29 Y 38

Padding: t 2 r 2 b 2 l 2

Margin: t 5 r 5 b 5 l 5

TextFormat

Face: Arial

Color/Size: 11 px

Style: B I U

Align:

Custom

Layout: absolute

Outline

- screen1
  - tabpanel3
    - tabpanelchild8
    - tabpanelchild9
    - tabpanelchild10
  - richtable2

Upload your own images as assets

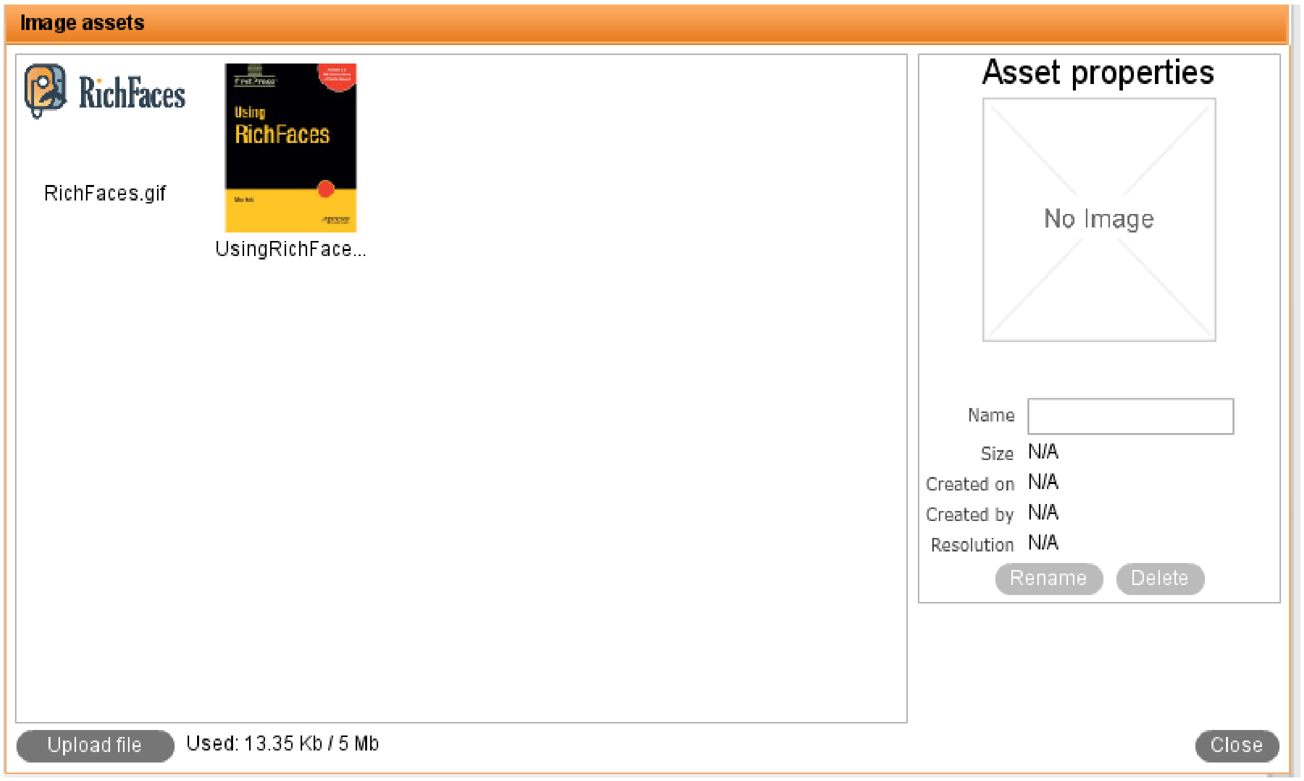


Image assets

Asset properties

No Image

Name

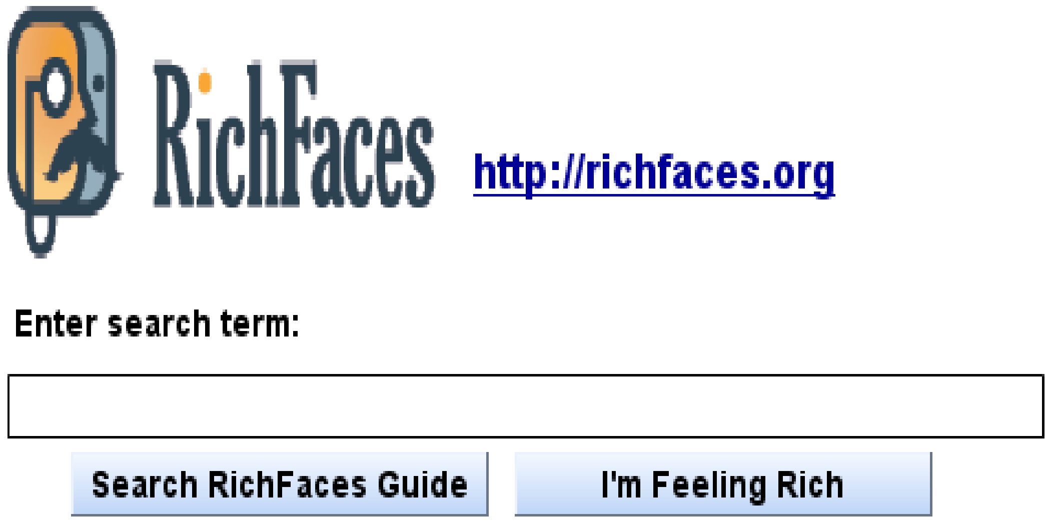
Size N/A

Created on N/A

Created by N/A

Resolution N/A

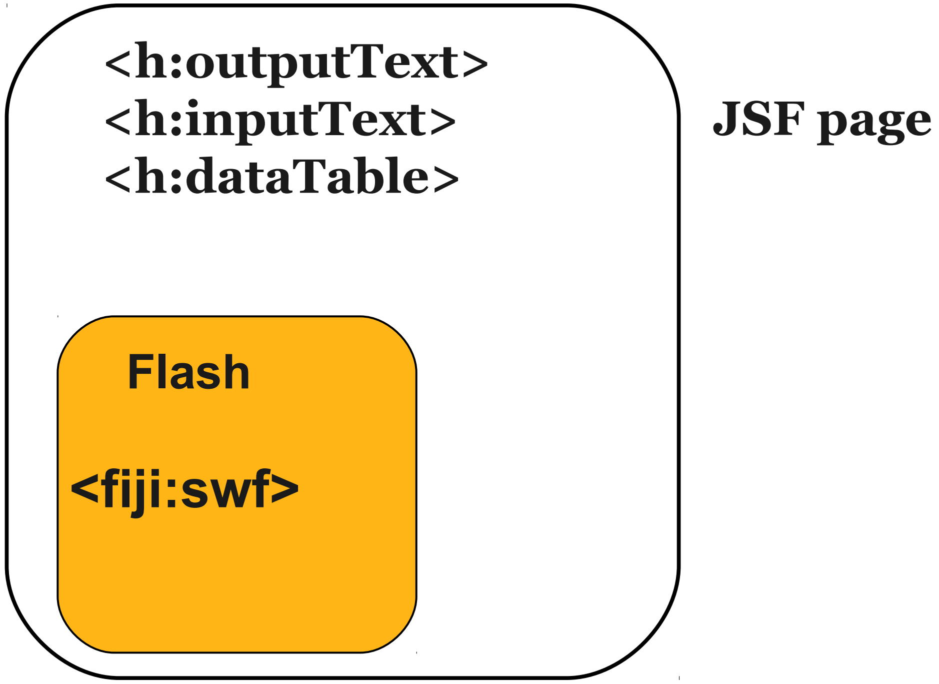
Used: 13.35 Kb / 5 Mb



http://tiggr.exadel.com

Exadel Fiji

- Insert any Flash/JavaFX widget into as JSF component into a JSF page
  - Comes with 7 ready-to-use charts (Flash)



How can we help with RichFaces

- Web application development with RichFaces
- Custom component development
- Training:

Training	Days
JSF 1.2, 2	1-2
RichFaces 3, 4	1-2
JSF and RichFaces	2-3
RichFaces 3 to 4	1-2

Thank you!

- max@exadel.com
- http://mkblog.exadel.com
- http://twitter.com/maxkatz
- http://exadel.org