



# JSF: The MyFaces Custom Components (Tomahawk)

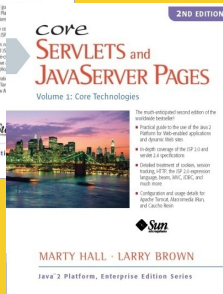
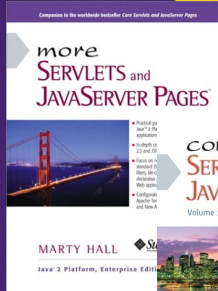
Originals of Slides and Source Code for Examples:

<http://www.coreservlets.com/JSF-Tutorial/>

This somewhat old tutorial covers JSF 1, and is left online for those maintaining existing projects. All **new** projects should use JSF 2, which is both simpler and more powerful. See <http://www.coreservlets.com/JSF-Tutorial/jsf2/>.

**Customized Java EE Training:** <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live training on JSF 1 or 2, please see courses at <http://courses.coreservlets.com/>.**



**Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.**

- Courses developed and taught by Marty Hall
    - JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 6 or 7 programming, custom mix of topics
    - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
  - Courses developed and taught by [coreservlets.com](http://coreservlets.com) experts (edited by Marty)
    - Spring, Hibernate/JPA, EJB3, GWT, Hadoop, SOAP-based and RESTful Web Services
- Contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for details**

## Topics in This Section

- **Popular component libraries**
- **Getting the Tomahawk components**
- **Configuring MyFaces to use Tomahawk**
- **Sample components**
  - Date input
  - Tabbed panes
  - Popups
  - Data lists
  - Tables with column flow
- **Sample validators**
  - Regular expressions
  - Email addresses
  - Credit card numbers
  - Equality

4

## Popular Component Libraries (From MyFaces)

- **Tomahawk**
  - Original MyFaces component library
    - Currently the most popular library
    - Poor documentation
- **Tobago**
  - Apache library now under the MyFaces umbrella
    - Currently the second-most popular library
- **Trinidad**
  - Originally Oracle ADF components
    - Donated by Oracle to Apache in 2006
    - Very extensive and professional
    - Very poorly integrated and documented at this early stage
    - Likely to become one of the (the?) most popular libraries

5

## Popular Free Component Libraries (From Other Vendors)

- **RichFaces (and Ajax4jsf)**
  - <http://labs.jboss.com/jbossrichfaces/> (free)
- **IceFaces**
  - <http://www.icefaces.org/> (free)
- **Backbase Ajax/JSF Library**
  - <http://www.backbase.com/products/enterprise-ajax-for-jsf/overview/>
  - Main version is not free, but scaled-down community version is free
- **Infragistics NetAdvantage**
  - <http://www.infragistics.com/java/netadvantage/> (not free)
- **GWT-JSF Integration Library**
  - <https://ajax4jsf.dev.java.net/>
  - Still in early stages, but it wraps the Google Web Toolkit in JSF components, and GWT is perhaps *the* most important Ajax library
- **Others (Free and For-Cost)**
  - See summary at <http://www.jsfmatrix.net/>

6

## Downloading Tomahawk

- **Base JAR file**
  - Download Tomahawk ZIP file from <http://myfaces.apache.org/download.html>
    - For tomahawk.jar
- **Required undocumented JAR files**
  - JAR files for fileupload, validator, and ORO regular expressions needed, but not documented
  - Can be found in WEB-INF/lib directory of Tomcat examples app
    - Omitted from all "release" download pages in Sept-Nov 2006
    - Can be found at <http://people.apache.org/builds/myfaces/nightly/>
- **All files packaged in one neat Web app**
  - jsf-blank-with-extensions
    - from <http://www.coreservlets.com/JSF-Tutorial/>

7

## Using the MyFaces Components

- **Add JAR files to WEB-INF/lib**

- Documented:
  - tomahawk.jar
- Undocumented
  - commons-fileupload.jar
  - commons-validator.jar
  - oro-x.x.x.jar

- **Import the extended tag library**

```
<%@ taglib uri="http://myfaces.apache.org/tomahawk"  
        prefix="t"%>
```

- Notes
  - Older documentation and examples use "x" as the prefix
  - As of Nov 2006, some of the online examples still use the wrong extension

8

## Using the MyFaces Components (Continued)

- **Enable the extensions filter**

- Add a filter and two filter-mapping entries to web.xml
  - Delivers JavaScript code and style sheets for "fake" subdirectories in your Tomahawk Web apps
  - Components that use JavaScript will totally fail without these web.xml entries
    - The single most common beginner problem
- See <http://myfaces.apache.org/tomahawk/extensionsFilter.html>

- **Simple "blank" Web app with all required JAR files and filter settings**

- jsf-blank from <http://www.coreservlets.com/JSF-Tutorial/>

9

## Using Tomahawk: Shortcuts

- **Copy jsf-blank-with-extensions**
  - Download from <http://www.coreservlets.com/JSF-Tutorial/>
  - Preconfigured
    - All necessary JAR files and web.xml entries
- **Rename it**
  - my-tomahawk-app
- **Use it as the starting point for your Web apps**

10

## Tomahawk Documentation

- **Poorly organized**
  - Several of the most important parts are not given in the top-level Tomcat page and are *very* hard to find
- **Wiki usage page**
  - <http://myfaces.apache.org/tomahawk/>
- **Online API for Java code**
  - <http://myfaces.apache.org/tomahawk/apidocs/>
- **Online API for tag libraries**
  - <http://myfaces.apache.org/tomahawk/tlddoc/>
- **Examples with source code**
  - <http://www.irian.at/myfaces/home.jsf>

11

# t:inputDate

- **Purpose**
  - To gather a java.util.Date value
- **Attributes**
  - value
    - the Date value
  - type
    - date, time, or both. Default is date
  - popupCalendar
    - If true, button added that pops up JavaScript calendar for input. Default is false
- **Notes**
  - A null input value results in the current date being shown
  - Enabling JavaScript is tricky: filter and filter-mapping entries in web.xml needed.
    - HTML source is not very readable because it refers to JavaScript files via "fake" paths

12

# t:inputDate: Example Code

- **Main page**

```
<h:form>
  Date: <t:inputDate value="#{sample.date}"
                    popupCalendar="true"/><BR>
  <h:commandButton action="show-date"/>
</h:form>
```
- **Bean**

```
package coreservlets;
import java.util.*;

public class SampleBean {
  private Date date;

  public Date getDate() { return(date); }

  public void setDate(Date date) {
    this.date = date;
  }
}
```

13

# t:inputDate: Example Code

- **faces-config.xml**

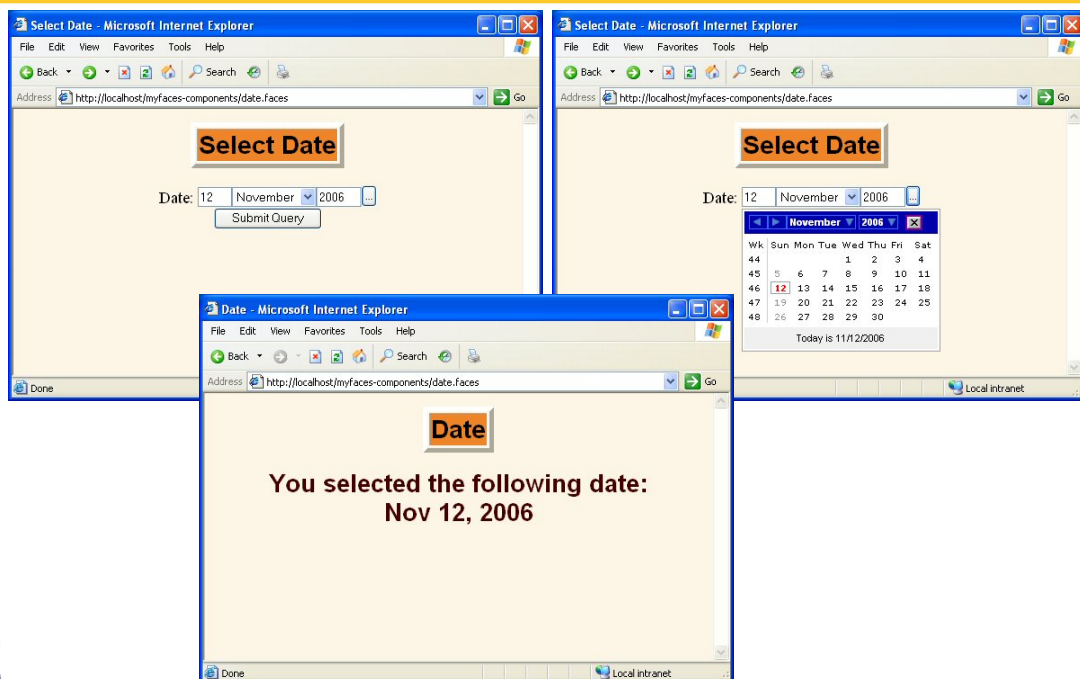
```
<managed-bean>
  <managed-bean-name>sample</managed-bean-name>
  <managed-bean-class>
    coreservlets.SampleBean
  </managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
<navigation-rule>
  <from-view-id>/date.jsp</from-view-id>
  <navigation-case>
    <from-outcome>show-date</from-outcome>
    <to-view-id>/WEB-INF/results/show-date.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

- **Results Page**

```
<H2>You selected the following date:<BR>
<h:outputText value="#{sample.date}"/></H2>
```

14

# t:inputDate: Results



15

# t:panelTabbedPane

- **Purpose**
  - To use CSS layers to create tabbed panes
- **Attributes**
  - bgColor
    - The background color of the active tab
  - activeTabStyleClass, inactiveTabStyleClass, etc.
    - Many attributes for giving CSS styles
- **Notes**
  - Per-tab content goes within t:panelTab
    - Regular HTML must be inside f:verbatim
  - Shared content goes outside of t:PanelTab but within t:panelTabbedPane
  - Non-tab content goes outside of t:panelTabbedPane
  - Entire thing must be inside h:form
    - Old examples omitted this and are technically illegal

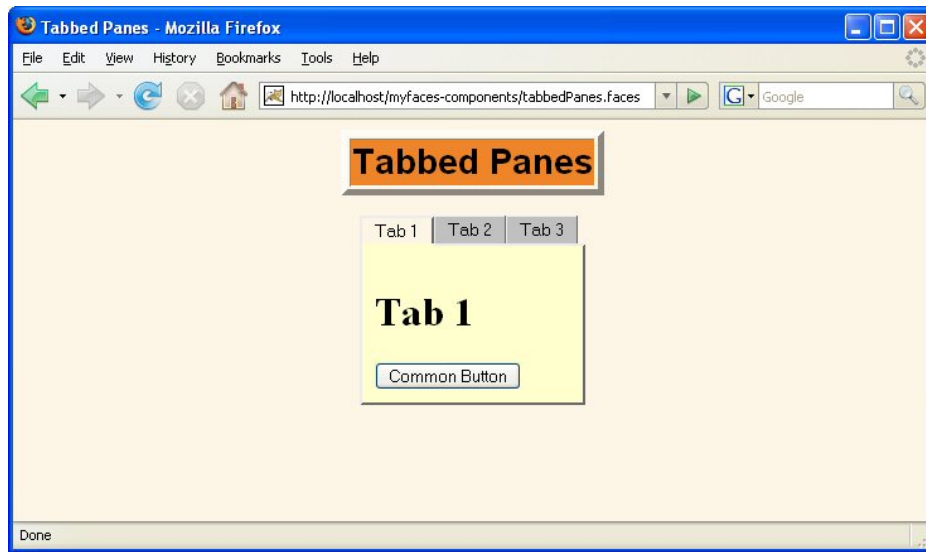
16

# t:panelTabbedPane: Example Code

```
<h:form>
<t:panelTabbedPane bgcolor="#FFFFCC">
  <t:panelTab label="Tab 1">
    <f:verbatim><H1>Tab 1</H1></f:verbatim>
  </t:panelTab>
  <t:panelTab label="Tab 2">
    <f:verbatim><H1>Tab 2</H1></f:verbatim>
  </t:panelTab>
  <t:panelTab label="Tab 3">
    <f:verbatim><H1>Tab 3</H1></f:verbatim>
  </t:panelTab>
  <h:commandButton value="Common Button" action="..." />
</t:panelTabbedPane>
</h:form>
```

17

# t:PanelTabbedPane: Result



18

# t:popup

- **Purpose**
  - To make HTML text that pops up (using layers) when the user moves the mouse over the specified text or images
- **Attributes**
  - displayAtDistanceX, displayAtDistanceY
    - Offset relative to where mouse entered the text
  - closePopupOnExitingElement, closePopupOnExitingPopup
    - Flag for whether popup should close automatically (true by default)
  - Many JavaScript and CSS entries
    - styleClass refers to style of the popup, not the main text
- **Notes**
  - Use `<f:facet name="popup">` inside `t:popup` for actual popup text
  - The offsets are relative to where mouse entered text, so popup will appear in different places when cursor comes onto text from different directions.

19

## t:popup: Typical Usage

```
<t:popup displayAtDistanceX="10"  
        displayAtDistanceY="10"  
        styleClass="popupStyle">  
  <h:outputText value="Main text"/>  
  <f:facet name="popup">  
    <h:outputText value="Popup Text"/>  
  </f:facet>  
</t:popup>
```

20

## t:popup: Example Code (JSP)

```
<H2>  
<t:popup displayAtDistanceX="10"  
        displayAtDistanceY="10"  
        styleClass="popupStyle">  
  <h:outputText value="This is a test"/>  
  <f:facet name="popup">  
    <h:outputText value="Some Sample Popup Text"/>  
  </f:facet>  
</t:popup>  
</H2>
```

21

## t:popup: Example Code (JSP Continued)

```
<H2>Some people attending the coreservlets.com party</H2>
<UL CLASS="larger">
  <LI><B><t:popup displayAtDistanceX="10"
    displayAtDistanceY="10"
    styleClass="popupStyle">
      <h:outputText value="#{sample.names[0].name}"/>
      <f:facet name="popup">
        <h:outputText
          value="#{sample.names[0].address}"/>
        </f:facet>
      </t:popup></B>
  <LI><B><t:popup displayAtDistanceX="10"
    displayAtDistanceY="10"
    styleClass="popupStyle">
      <h:outputText value="#{sample.names[1].name}"/>
      <f:facet name="popup">
        <h:outputText
          value="#{sample.names[1].address}"/>
        </f:facet>
      </t:popup></B>
  ...
</UL>
```

22

## t:popup: Example Code (Bean)

```
public class SampleBean {
  ...
  public NameBean[] getNames() {
    NameBean[] names =
      { new NameBean
        ("Marty Hall",
         "6 Meadowsweet Ct., Reisterstown MD 21136"),
        new NameBean
        ("Bill Gates",
         "One Microsoft Way, Redmond WA 98052"),
        new NameBean
        ("George W. Bush",
         "1600 Pennsylvania Avenue, Washington DC 20500") };
    return(names);
  }
}
```

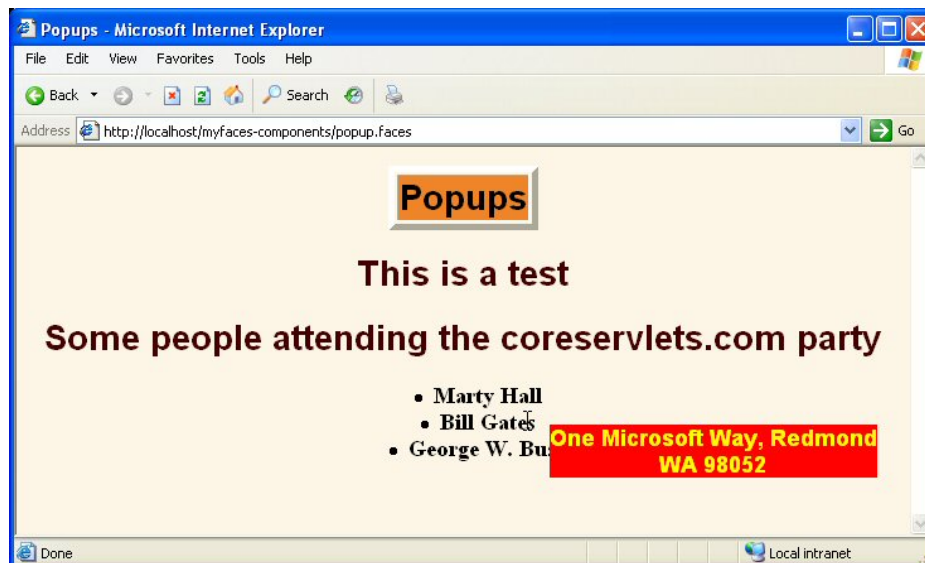
23

## t:popup: Example Code (Style Sheet)

```
...  
.popupStyle { background-color: red;  
               color: yellow;  
               font-size: 18px;  
               font-family: Arial, Helvetica, sans-serif;  
             }  
...
```

24

## t:popup: Results



25

## t:dataList

- **Purpose**

- To iterate through a collection (ala h:dataTable), but with choices in what HTML gets produced
- Each subelement can be put into an LI element in a list, or not wrapped with HTML at all
  - Letting JSF page author choose the markup

- **Attributes**

- value: the collection
- var: the local variable (as in h:dataTable)
- layout
  - One of simple, unorderedList, or orderedList
    - No markup, UL, OL, respectively. Default is simple.
- styleClass and itemStyleClass
- Many JavaScript attributes

26

## t:dataList: Example Code (JSP)

```
<t:dataList value="#{sample.states}" var="state"
           layout="unorderedList">
  <t:popup displayAtDistanceX="10"
          displayAtDistanceY="10"
          styleClass="popupStyle">
    <h:outputText value="#{state[0]}" />
    <f:facet name="popup">
      <h:outputText value="#{state[1]}" />
    </f:facet>
  </t:popup>
</t:dataList>
```

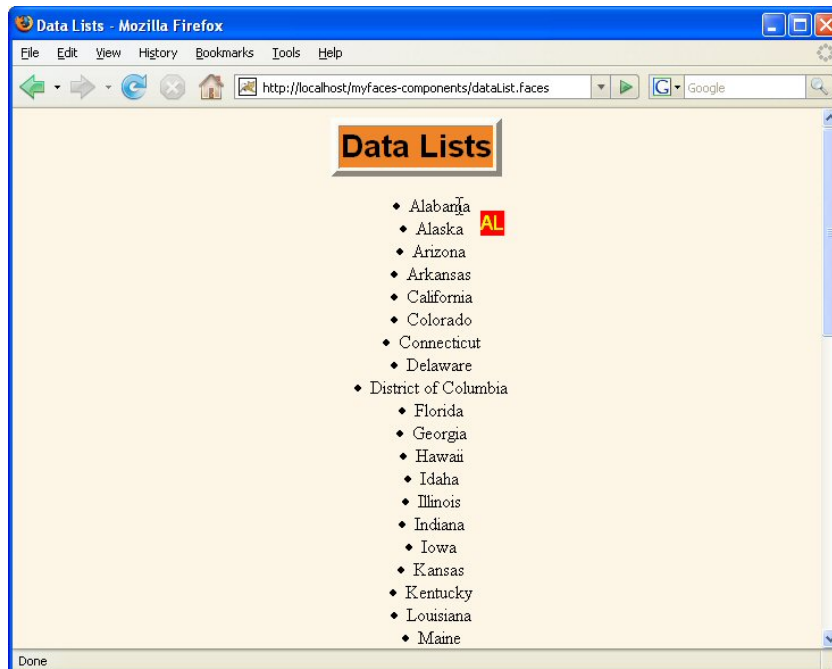
27

## t:dataList: Example Code (Bean)

```
public class SampleBean {  
    ...  
  
    private String[][] states =  
        { { "Alabama", "AL" },  
          { "Alaska", "AK" },  
          { "Arizona", "AZ" },  
          { "Arkansas", "AR" },  
          { "California", "CA" }, ... };  
  
    public String[][] getStates() {  
        { return(states); }  
    }  
}
```

28

## t:dataList: Result



29

# t:newspaperTable

- **Purpose**
  - To take a tall skinny table and turn it into a wider multi-column table with a balanced # of entries per column.
- **Attributes**
  - newspaperColumns
    - The number of columns
  - value
    - The collection containing the values
  - var
    - The local variable set to each entry of the collection
  - Many CSS entries
- **Notes**
  - Usage is similar to h:dataTable; see earlier lecture
  - Use h:column for each sub-piece of "var"

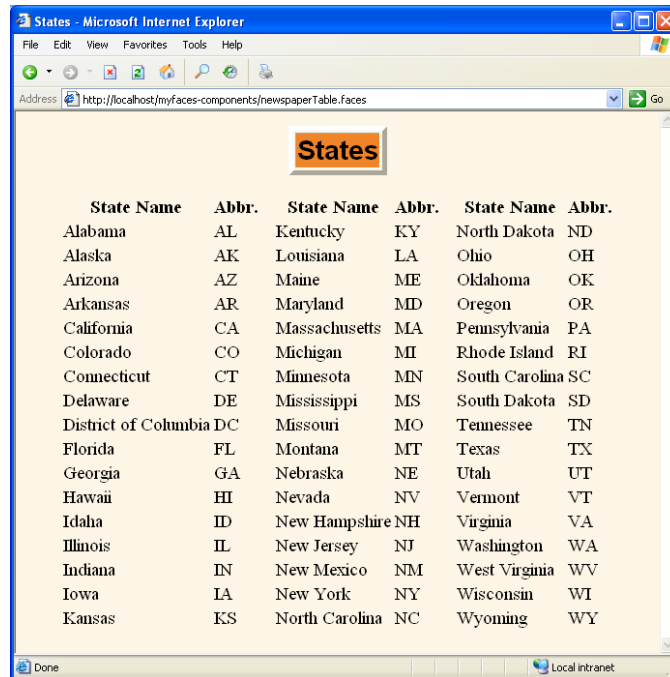
30

# t:newspaperTable: Example Code

```
<t:newspaperTable newspaperColumns="3"
                 value="#{sample.states}"
                 var="state">
  <f:facet name="spacer">
    <f:verbatim>&nbsp;&nbsp;&nbsp;</f:verbatim>
  </f:facet>
  <h:column>
    <f:facet name="header">
      <f:verbatim>State Name</f:verbatim>
    </f:facet>
    <h:outputText value="#{state[0]}" />
  </h:column>
  <h:column>
    <f:facet name="header">
      <f:verbatim>Abbr.</f:verbatim>
    </f:facet>
    <h:outputText value="#{state[1]}" />
  </h:column>
</t:newspaperTable>
```

31

# t:newspaperTable: Result

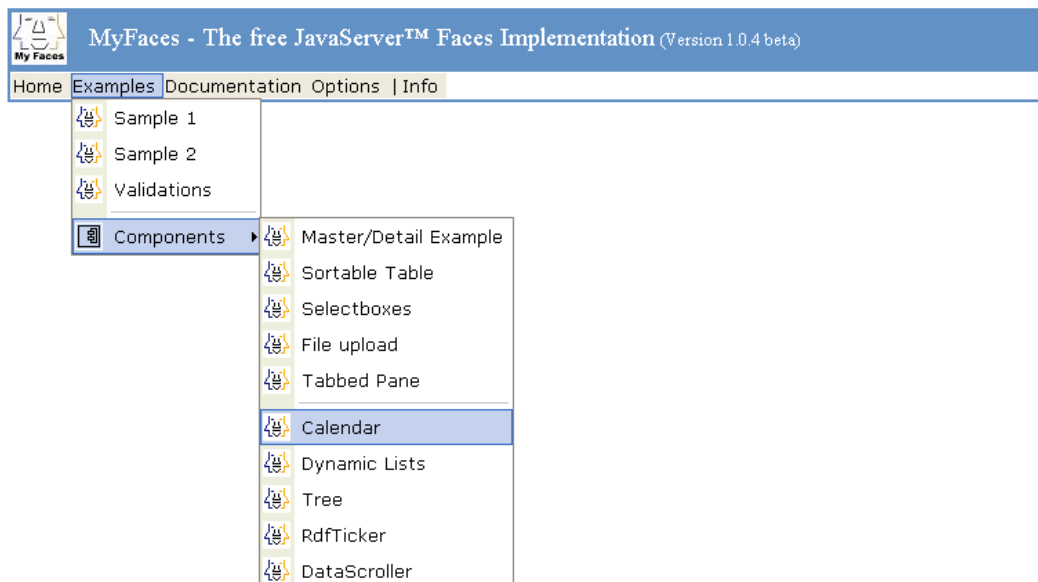


State Name	Abbr.	State Name	Abbr.	State Name	Abbr.
Alabama	AL	Kentucky	KY	North Dakota	ND
Alaska	AK	Louisiana	LA	Ohio	OH
Arizona	AZ	Maine	ME	Oklahoma	OK
Arkansas	AR	Maryland	MD	Oregon	OR
California	CA	Massachusetts	MA	Pennsylvania	PA
Colorado	CO	Michigan	MI	Rhode Island	RI
Connecticut	CT	Minnesota	MN	South Carolina	SC
Delaware	DE	Mississippi	MS	South Dakota	SD
District of Columbia	DC	Missouri	MO	Tennessee	TN
Florida	FL	Montana	MT	Texas	TX
Georgia	GA	Nebraska	NE	Utah	UT
Hawaii	HI	Nevada	NV	Vermont	VT
Idaho	ID	New Hampshire	NH	Virginia	VA
Illinois	IL	New Jersey	NJ	Washington	WA
Indiana	IN	New Mexico	NM	West Virginia	WV
Iowa	IA	New York	NY	Wisconsin	WI
Kansas	KS	North Carolina	NC	Wyoming	WY

32

# JavaScript Menus

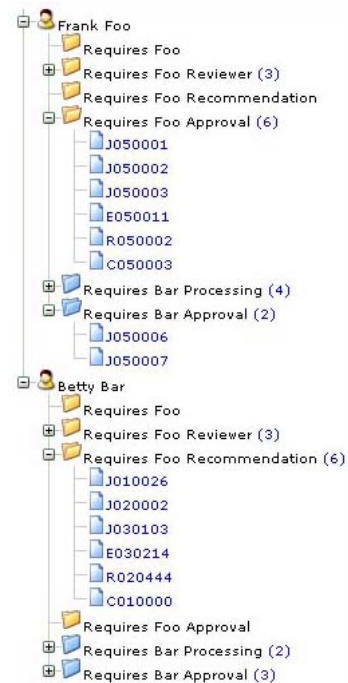
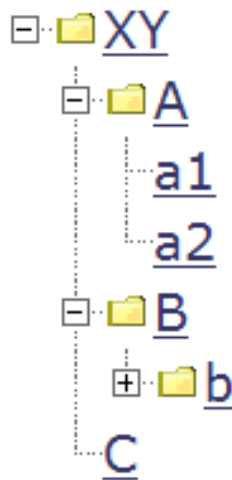
- **t:jsCookMenu**



33

## t:tree and t:tree2

- **Simple and advanced trees**



34

## t:fileUpload

- **Renders a file upload field**
  - Ie `<INPUT TYPE="FILE">`
- **MyFaces examples come bundled with Jakarta Commons File Upload code**
  - But the core Tomahawk download omits it
  - commons-fileupload-x.x-jar must be in WEB-INF/lib
  - Other components need this code also
    - Treat it as a required part of Tomahawk

35

# t:inputCalendar

- **Renders a calendar**
  - But t:inputDate more generally useful

The image shows two side-by-side examples of the t:inputCalendar widget. The left example, titled "Popup", shows a date input field containing "08.07.05" and a "Submit" button. A calendar popup is displayed over the input field, showing the month of July 2005. The popup includes a table with columns for the days of the week (Mo, Di, Mi, Do, Fr, Sa, So) and rows for the weeks. The date "8" is highlighted in red. Below the table, it says "Today is Fri, 8 Jul 2005". The right example, titled "No Popup", shows the same date input field and submit button, but instead of a popup, a calendar is rendered directly within the input field's area. It shows the month of July 2005 with the date "8" highlighted in red.

36

# t:inputHTML

- **An inline HTML-based word processor**

The image shows a screenshot of the t:inputHTML widget. It features a rich text editor interface with a toolbar at the top containing various formatting options like bold, italic, underline, and text color. Below the toolbar is a large, empty text area for editing. To the right of the text area is a "Properties" panel with two input fields: "Title:" and "Description:". The "Title:" field is a simple text input, and the "Description:" field is a text area with a scroll bar.

37

## Other Components

- **Many Additions and Extensions to h:dataTable**
  - Scrolling
  - Columns
  - Sorting
  - Etc.
- **More constantly being added**
  - But documentation always lagging

38

## Tomahawk Validators

- **Tomahawk includes several powerful and useful validators**
  - Mostly based on the Struts validators
  - But still no client-side validation support
- **Most used validators**
  - validateRegExpr
  - validateEmail
  - validateCreditCard
  - validateEquals
- **Attribute common to all Tomahawk validators**
  - message
    - Lets you change the error message text without messing with properties files. Very useful.

39

# validateRegExpr

- **validateRegExpr**

- pattern: the regular expression pattern

- Follows ORO syntax which differs very slightly from JDK 1.4/1.5 syntax

- **Example**

```
<TABLE BGCOLOR="WHITE">
<h:form>
...
<TR><TD>
    <B>ZIP Code:</B>
    <t:inputText value="#{order.zipCode}"
        required="true" id="ZIP">
        <t:validateRegExpr pattern="\d{5}"
            message="ZIP Code must be 5 digits"/>
    </t:inputText>
    <TD><h:message for="ZIP" styleClass="red"/></TD>
</TD></TR>
```

40

# validateEmail

- **Attributes**

- None required, but "message" useful

- **Example**

```
<TR><TD>
    <B>Email:</B>
    <t:inputText value="#{order.email}"
        required="true" id="email">
        <t:validateEmail
            message="Invalid email address"/>
    </t:inputText>
    <TD><h:message for="email" styleClass="red"/></TD>
</TD></TR>
```

41

# validateCreditCard

- **Attribute**

- None required, but "message" useful.
- You can also specify that only Visa, MasterCard, American Express, or Discover should be accepted
- Use 41111111111111 for testing

- **Example**

```
<TR><TD>
    <B>Credit Card:</B>
    <t:inputSecret value="#{order.creditCard}"
        required="true" id="card1">
        <t:validateCreditCard
            message="Invalid credit card number"/>
    </t:inputSecret>
    <TD><h:message for="card1" styleClass="red"/></TD>
</TD></TR>
```

42

# validateEquals

- **Attributes**

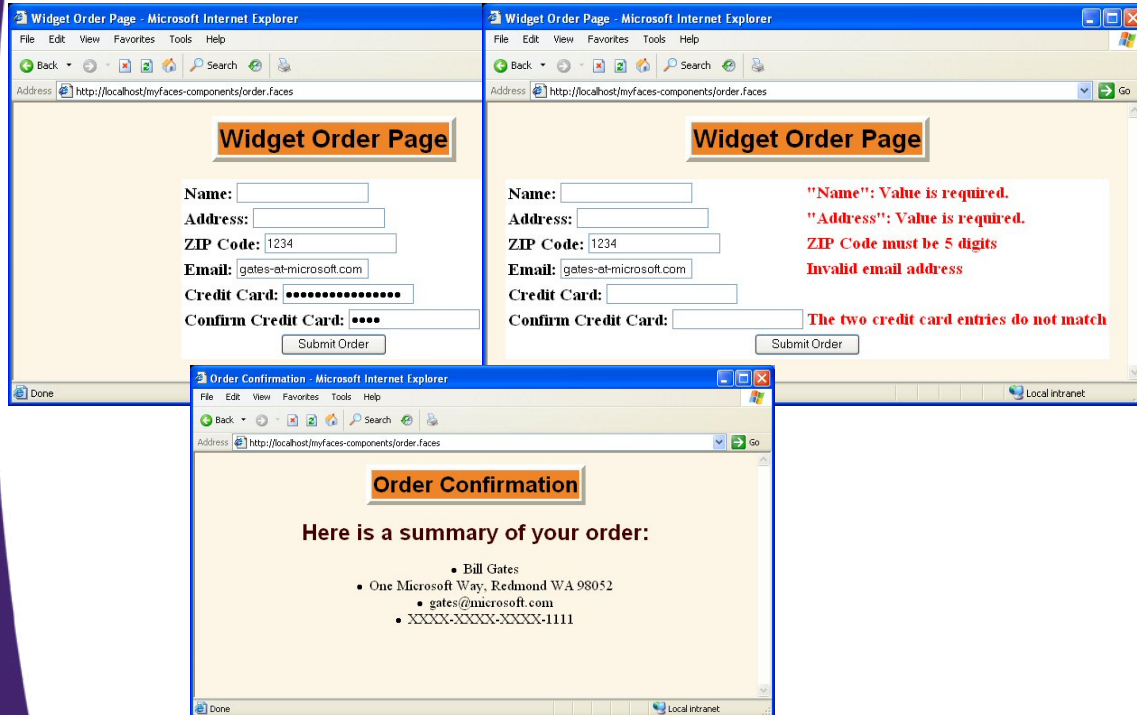
- for
  - Required: the id of the component that must match
- message
  - Optional but useful

- **Example**

```
<TR><TD>
    <B>Credit Card:</B>
    <t:inputSecret value="#{order.creditCard}"
        required="true" id="card1">
        <t:validateCreditCard
            message="Invalid credit card number"/>
    </t:inputSecret>
    <TD><h:message for="card1" styleClass="red"/></TD>
</TD></TR>
<TR><TD>
    <B>Confirm Credit Card:</B>
    <t:inputSecret required="true" id="card2">
        <t:validateEqual for="card1"
            message="The two credit card entries do not match"/>
    </t:inputSecret>
    <TD><h:message for="card2" styleClass="red"/></TD>
</TD></TR>
```

43

# Custom Validators: Results



## Summary

- **MyFaces supplies many powerful custom components and validators**
  - Documentation currently very poor
    - But supposedly this will change soon
  - More components being added
  - MyFaces components and validators can be used in *any* JSF implementation
- **Alternative**
  - Consider Oracle ADF components
    - Now called MyFaces Trinidad
    - Currently nearly impossible to use without reading the source code
    - Soon (?) to be better integrated and documented



# Questions?

**Customized Java EE Training: <http://courses.coreservlets.com/>**

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.