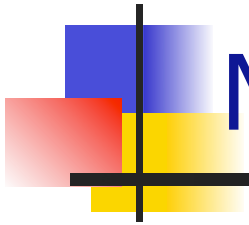


Mastering JavaServer Faces

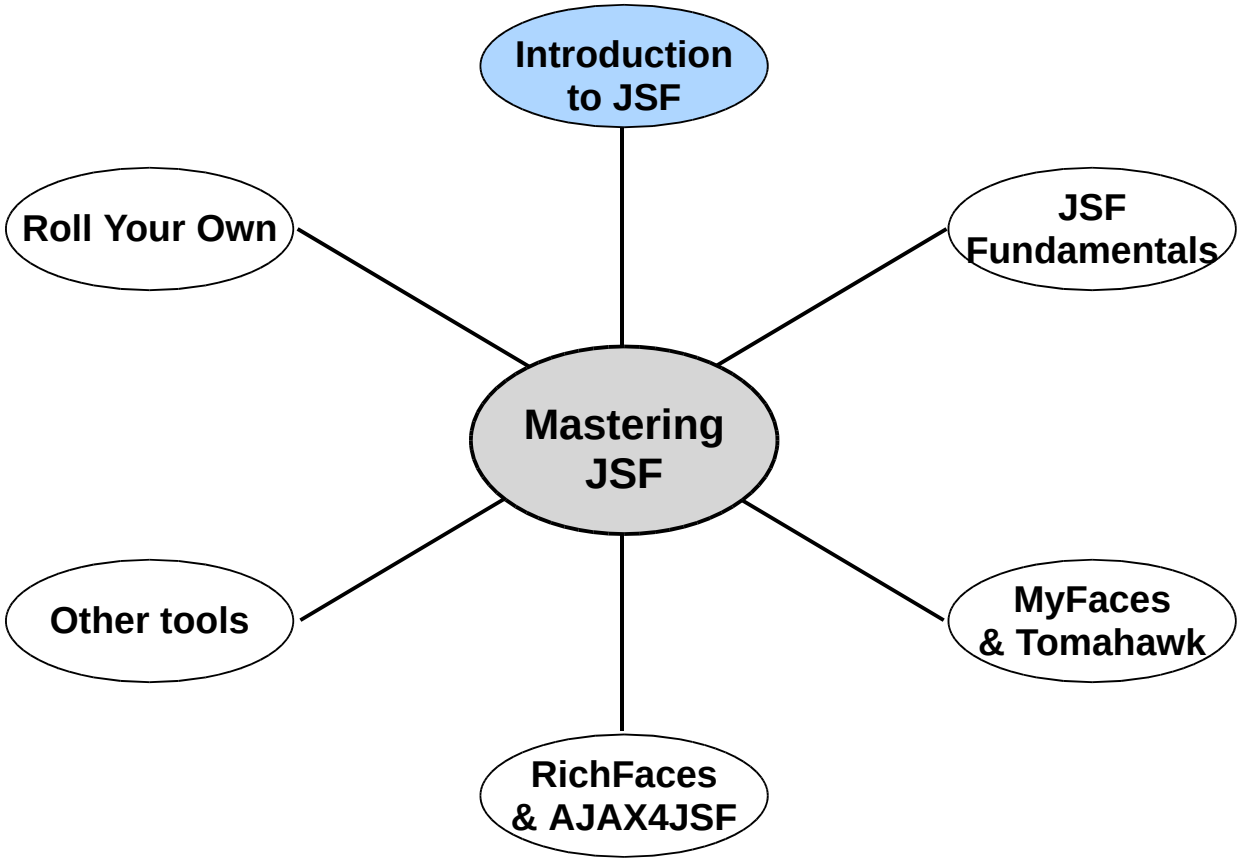
Bryan Basham

Software Alchemist

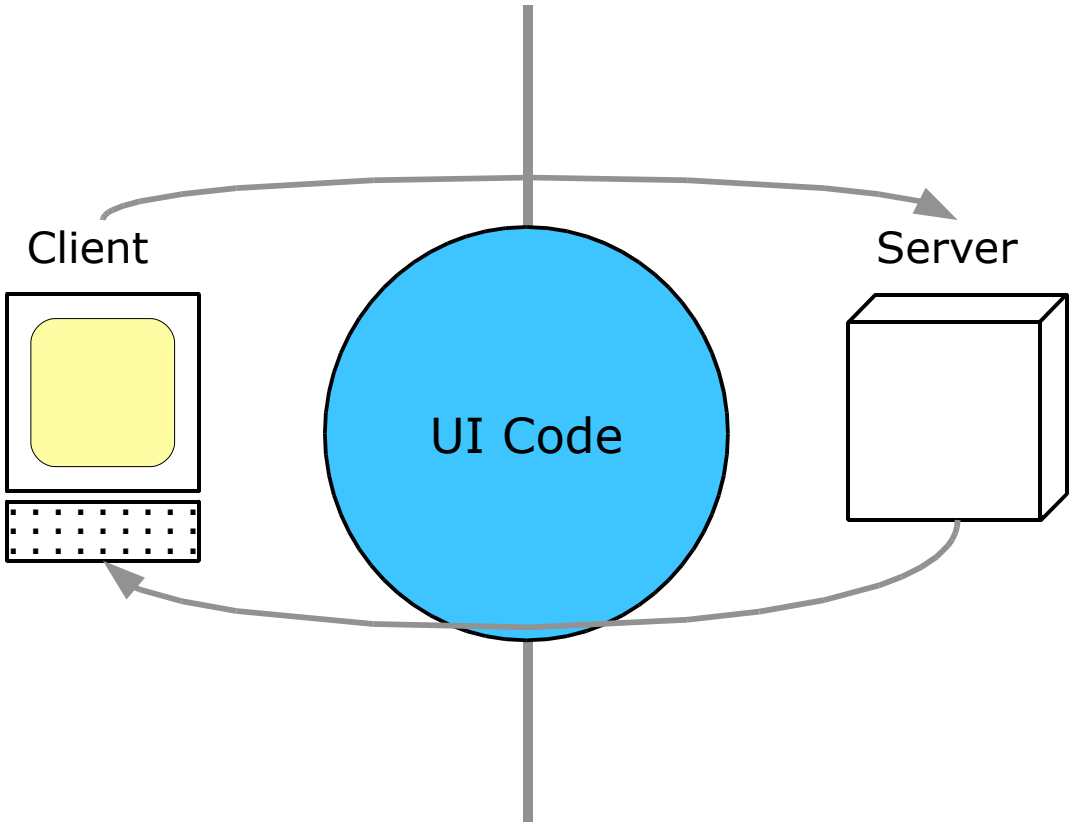
basham47@gmail.com

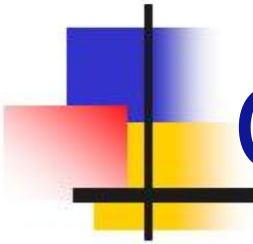
http://www.linkedin.com/in/SoftwareAlchemist

Topics Mind Map



Web UI Architecture Concerns





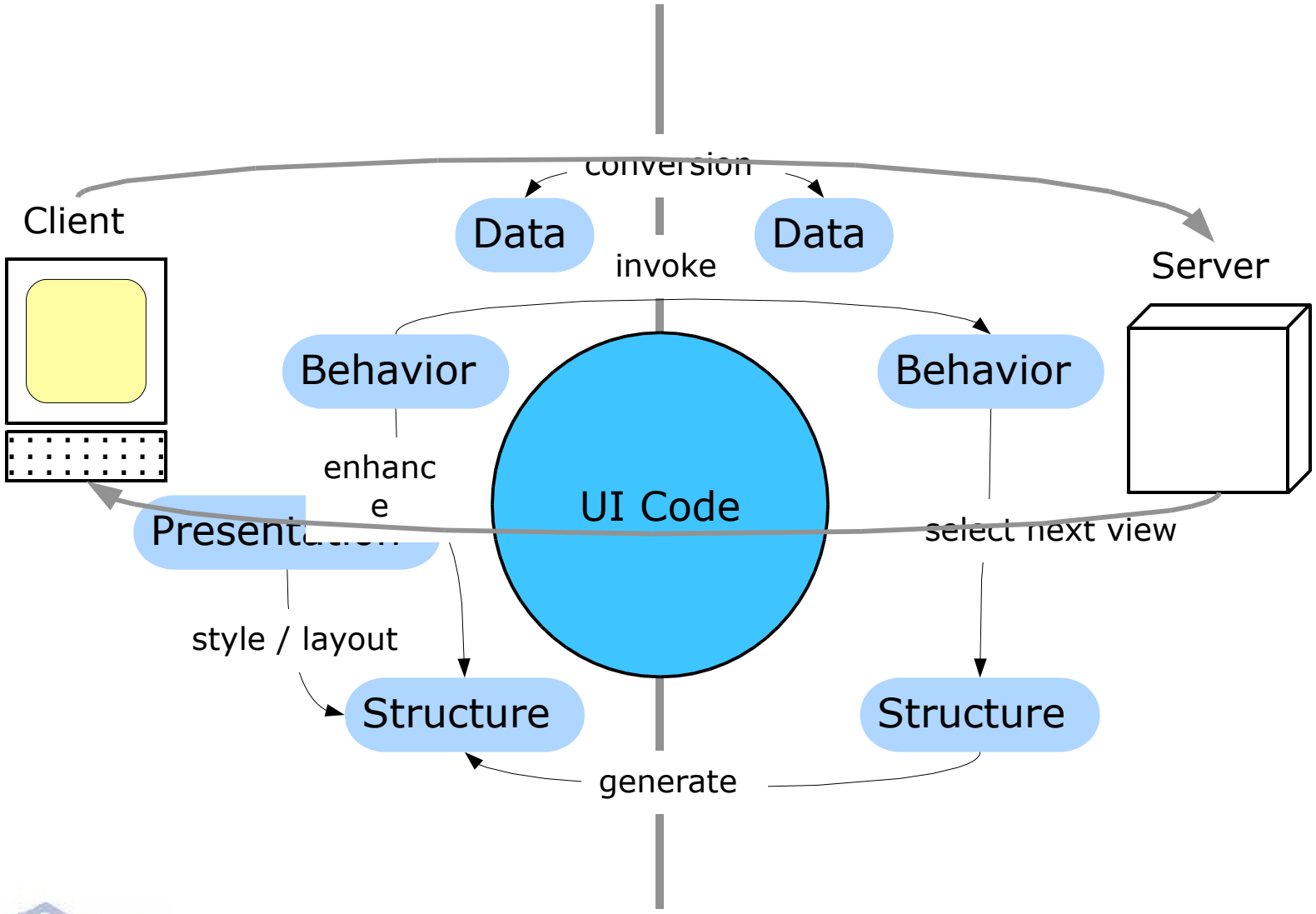
Client-side Technologies

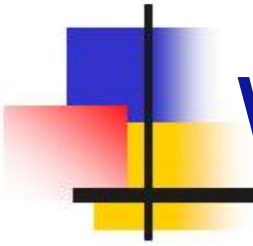
- Structure
  - XHTML (or HTML v4.01)
- Presentation
  - Cascading Style Sheets (CSS)
- Behavior
  - JavaScript (formally: ECMAScript 262 v3)
  - DOM API specification
  - JS objects (and JSON) for data models
  - Open source frameworks

Server-side Technologies

- Structure
  - JSF for components (UI widgets)
  - JSP (or other) for layout of component tree
- Behavior
  - JSF component event model
  - OO programming for data model
  - Open source implementations, frameworks, and component libraries

Separation of Concerns

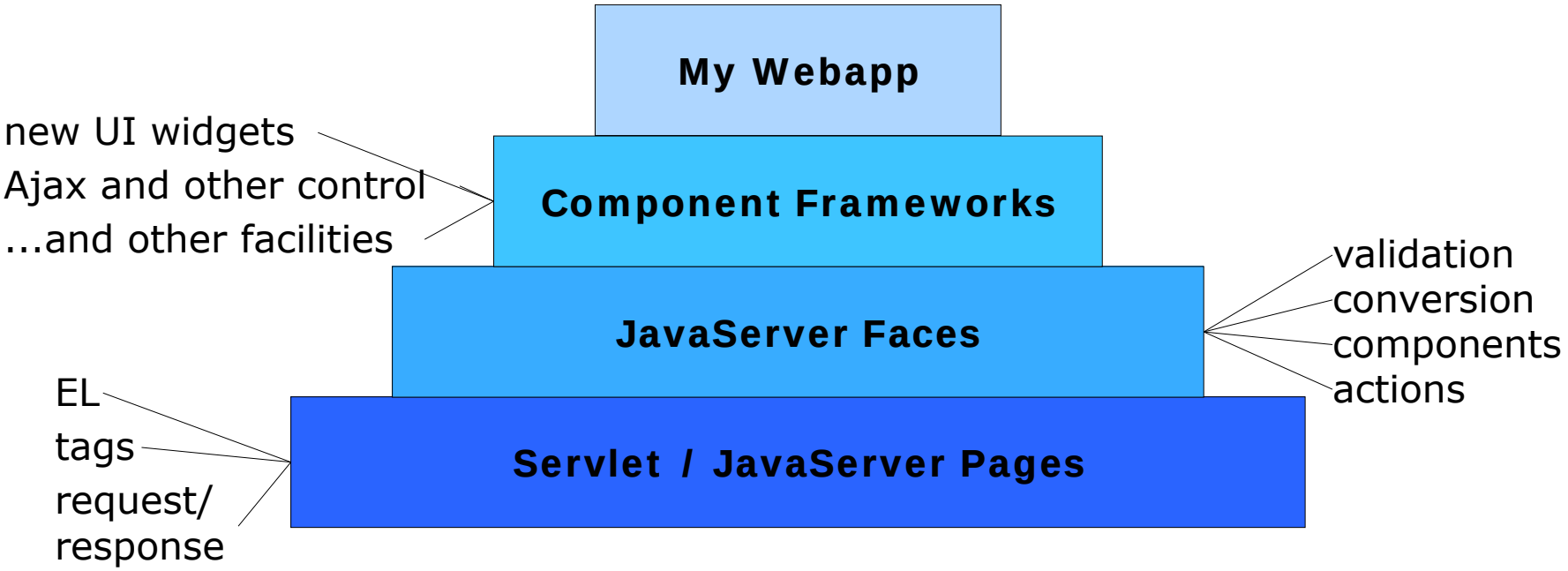


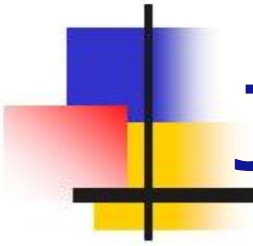


Why JavaServer Faces?

- Third generation Java-Web technology.
- Closer to traditional UI development:
  - component-based
  - event-driven
- Standards-based architecture
- Open source community

The JSF Stack

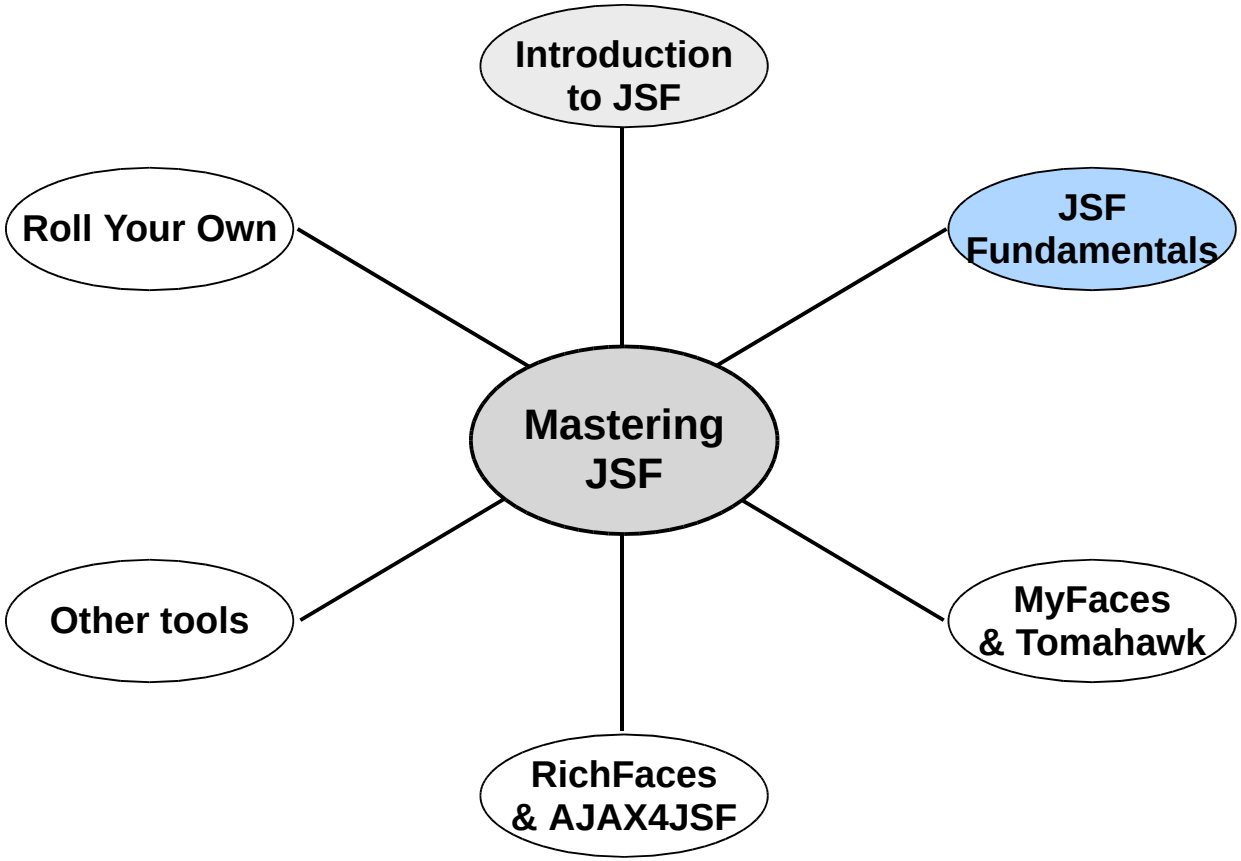




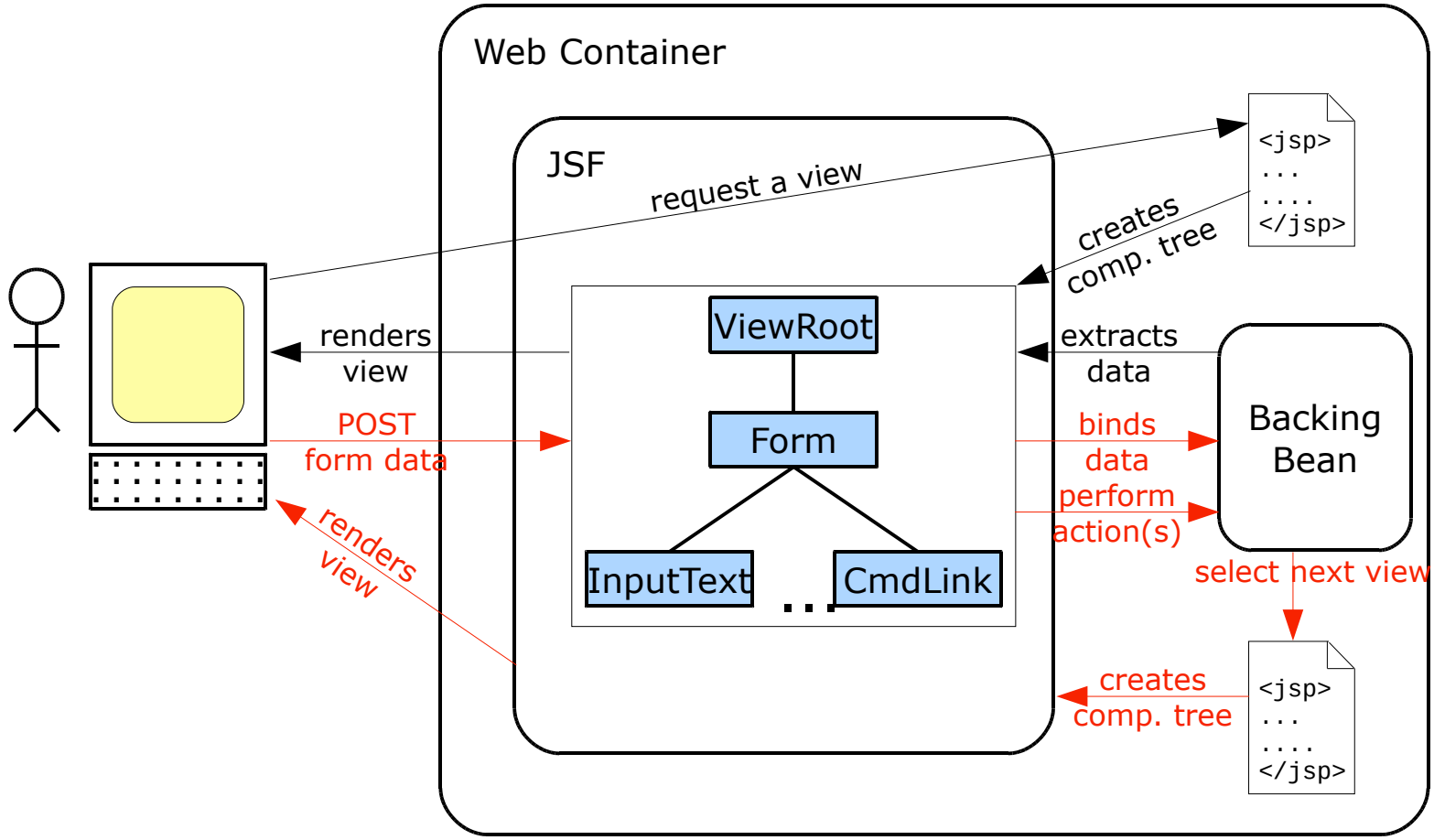
JSF and Web Standards

- JSF supports other Web standards
  - XHTML, for structure
  - CSS, for presentation
  - JavaScript, for client-side behavior
- Use JSF libraries that facilitate client-side Web standards
- The gotcha... JSF embeds JavaScript into the page

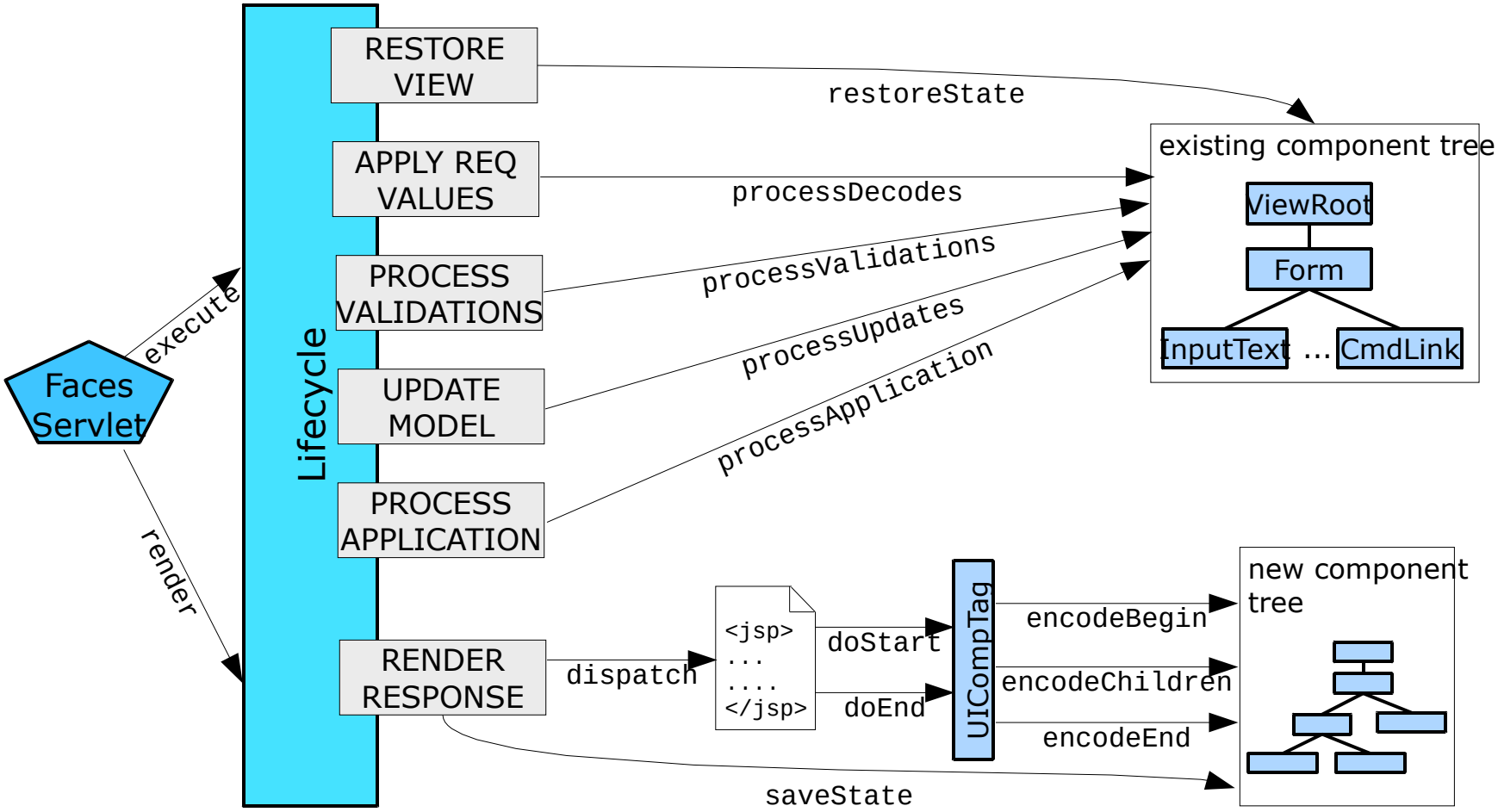
Topics Mind Map

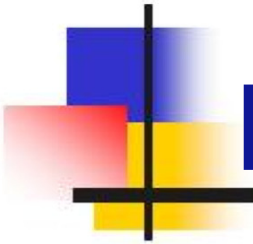


JSF Life Cycle



JSF Life Cycle

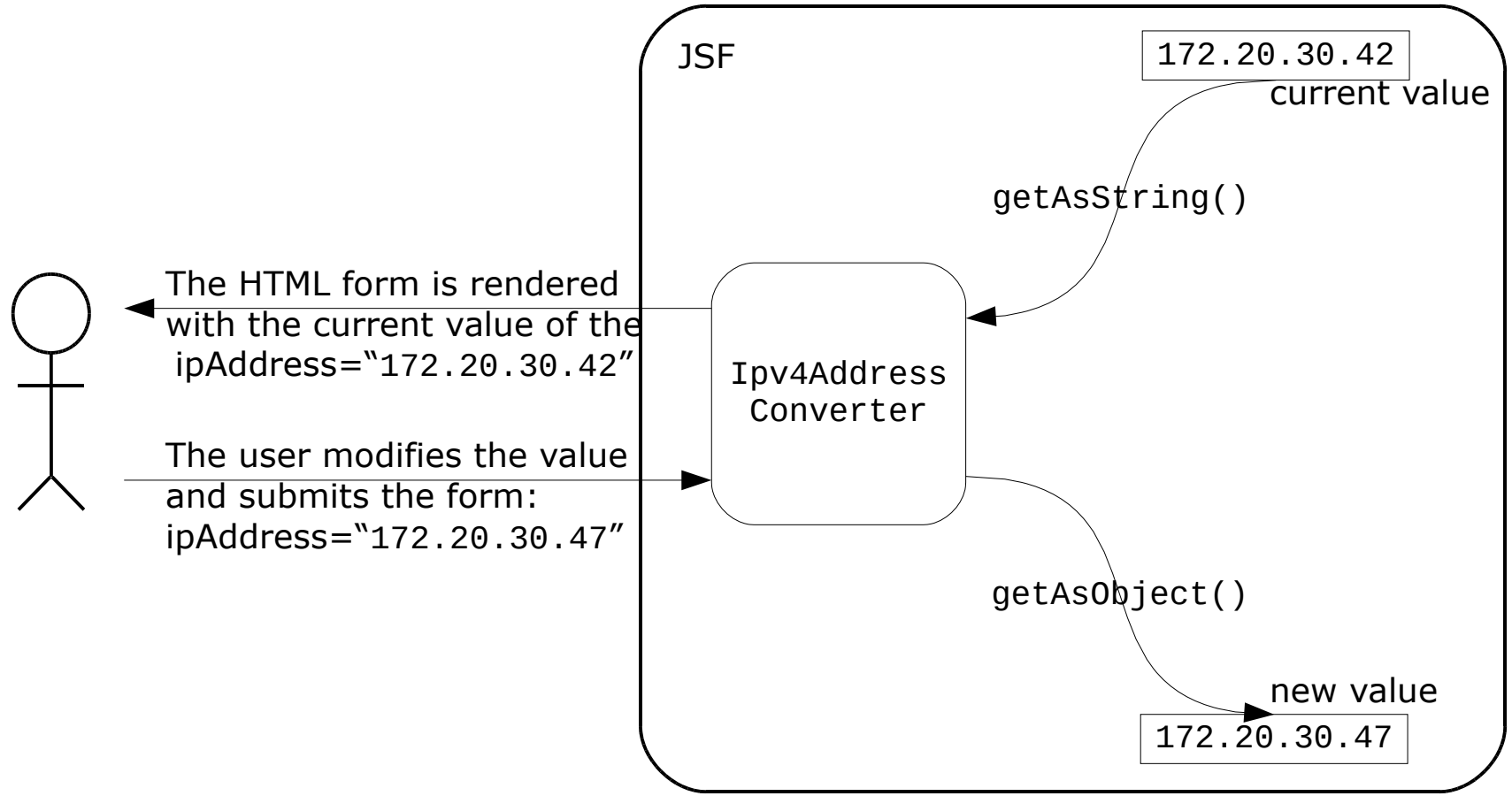


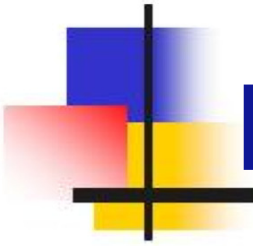


Data Conversion

- All HTTP request data are strings.
- Rich apps require rich data
  - simple non-string data: numbers, Boolean values, IP addresses, money, credit card #
  - complex data: structures, objects, entities with relationships
- The JSF `Converter` interface:
  - The `getAsObject` method
  - The `getAsString` method

Data Conversion





Field Validation

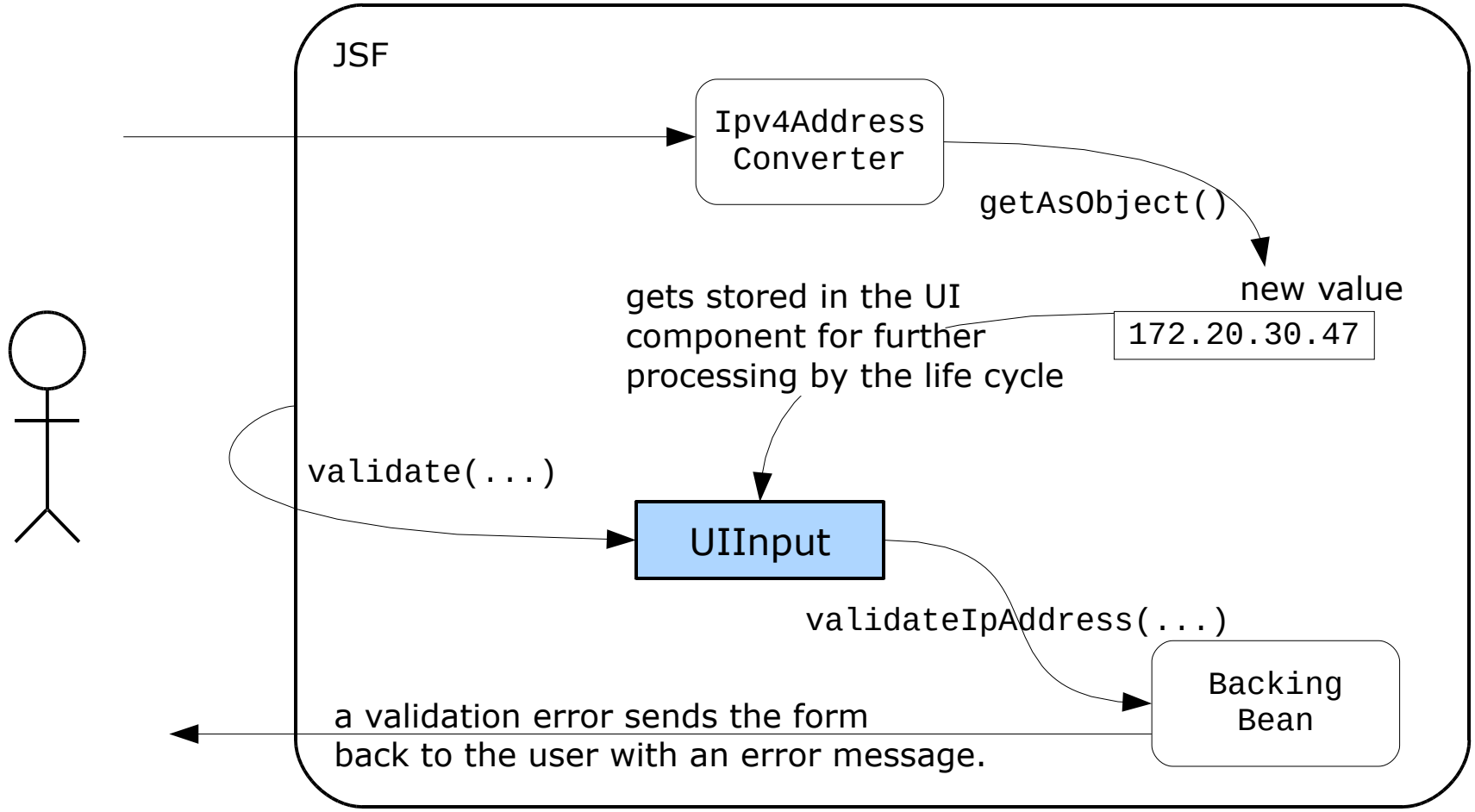
- Allows field validation checks after data conversion
- Option #1: call a method on a backing bean

```
<c:textField id="ipAddress" value="#{editIntf.interface.ipAddress}" required="true"
    converterId="converter.Ipv4Address" validator="#{editIntf.validateIpAddress}" />
```

- Option #2: invoke a validator object

```
<c:textField id="number" value="#{bean.myNumber}" required="true">
       <f:validateLongRange minimum="10" maximum="100" />
</c:textField>
```

Field Validation



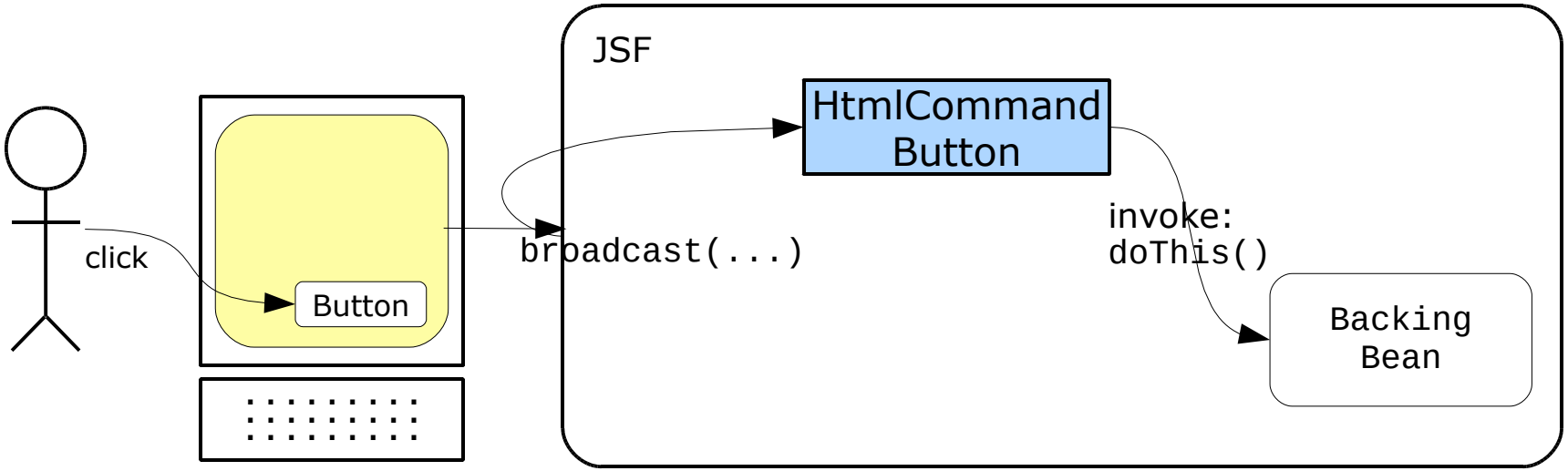
Application Actions

- Action components
  - UICommand (abstract), HtmlCommandLink, and HtmlCommandButton
- Example:

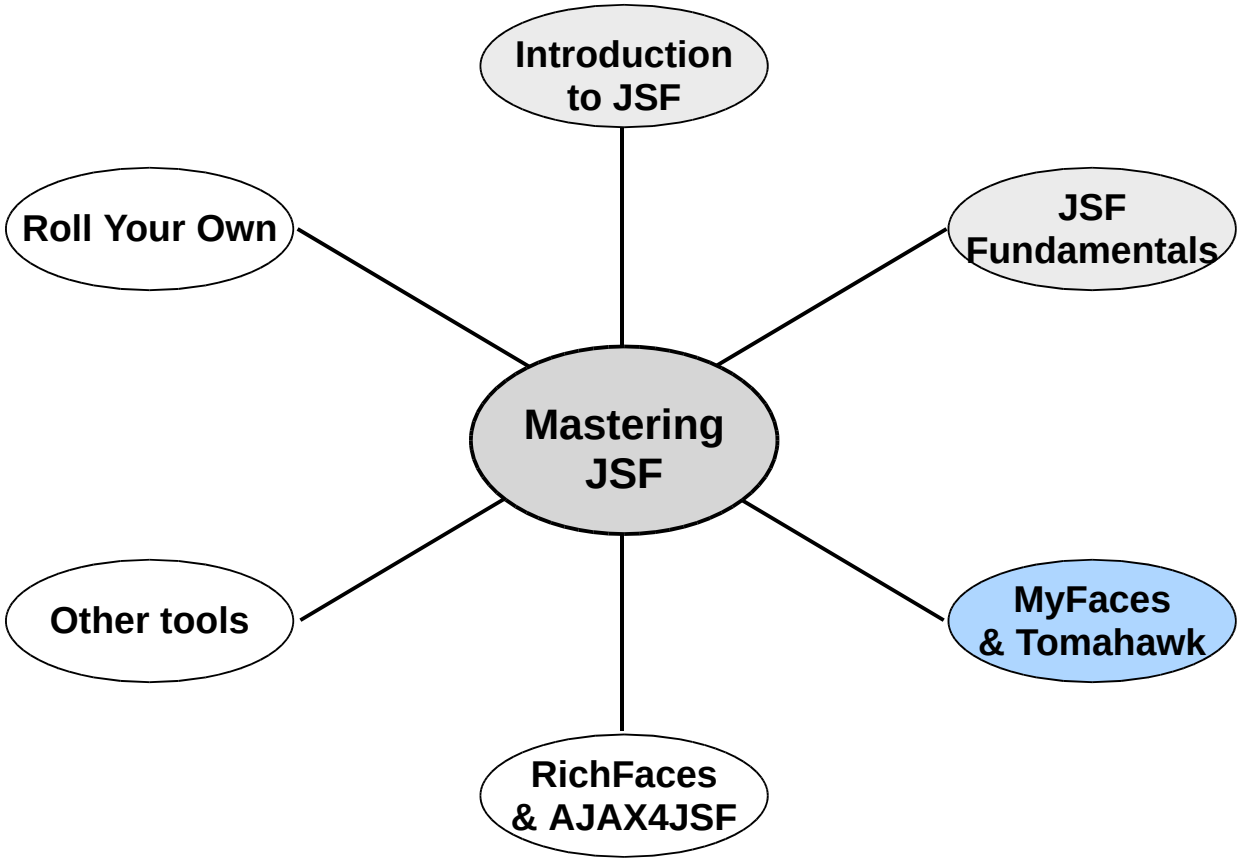
```
<h:commandButton value="Submit" actionListener="#{myBean.doThis}" />
```

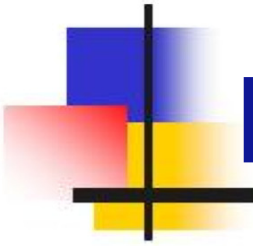
- Invokes the `doThis` method of your backing bean when the user clicks the button

Application Actions



Topics Mind Map





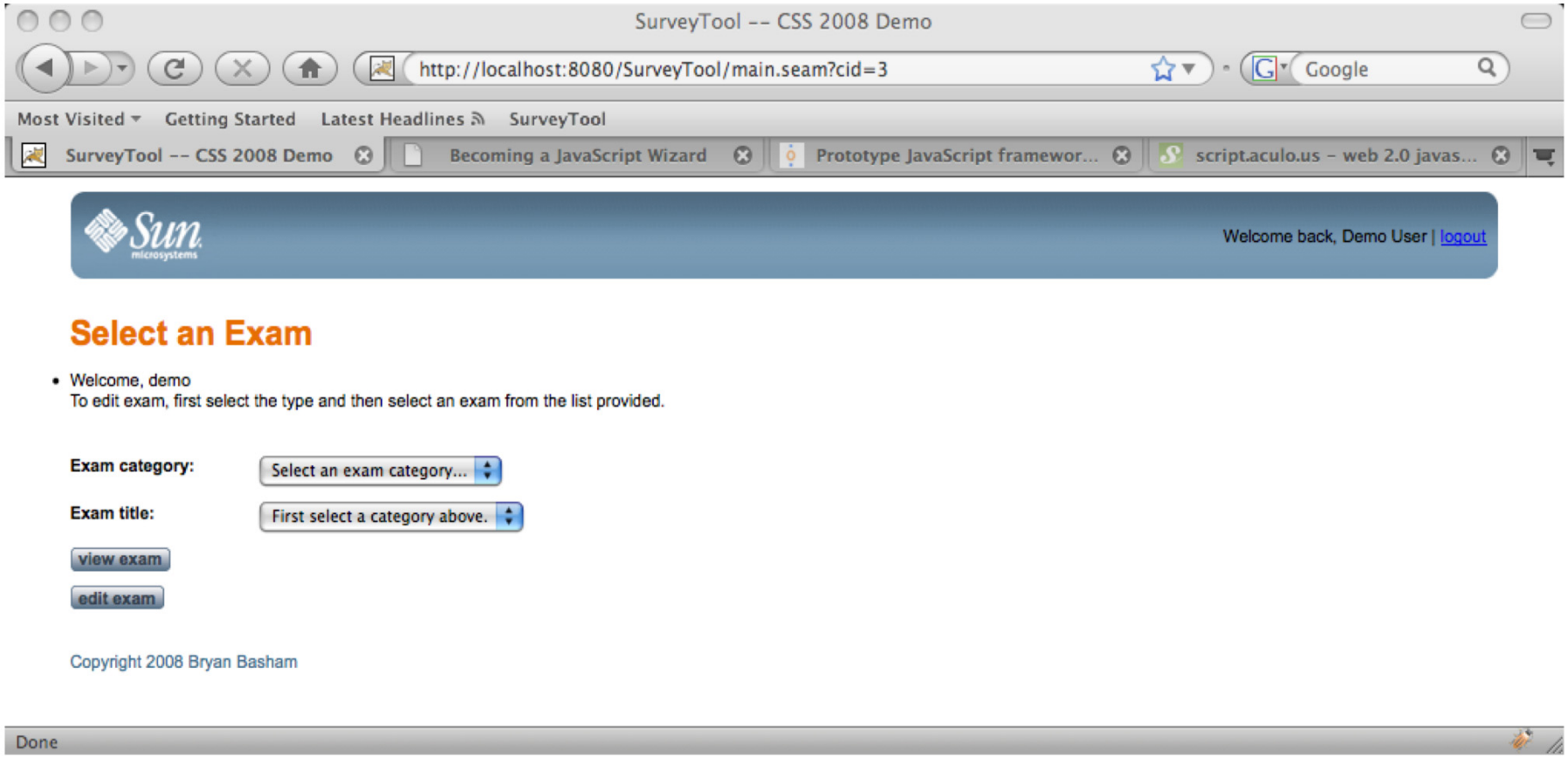
MyFaces

- is an implementation of the JSF spec.
- is an open source tool from Apache
- provides the standard set of UI widgets
- also includes several additional component libraries:
  - Tomahawk
  - Trinidad
  - Tobago
  - Orchestra

Standard Widget Set

- Output: f:view, h:outputText, h:graphicImage,
- Table: h:dataTable, h:column
- Forms:
  - h:form, h:outputLabel, h:inputText/Secret/Textarea/...
  - h:selectBooleanCheckbox
  - h:selectOneListbox/Menu/Radio
  - h:selectManyListbox/Menu/Checkbox
- Layout: h:panelGrid, h:panelGroup

SelectOneMenu: Screen Shot

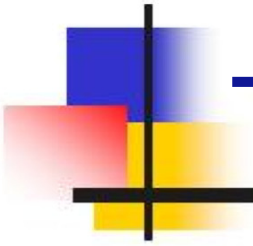


SelectOneMenu: JSP

```
<div id="exam_select">
    <div>
       <h:outputLabel for="exam_type_select">Exam category:</h:outputLabel> 
       <h:selectOneMenu value="#{examEditor.categoryOID}" id="exam_type_select">
          <f:selectItems value="#{examEditor.categorySelectItems}" />
       </h:selectOneMenu>
    </div>
    <div>
       <h:outputLabel for="exam_name_select">Exam title:</h:outputLabel> 
       <h:selectOneMenu value="#{examEditor.examOID}" id="exam_name_select">
          <f:selectItems value="#{examEditor.examSelectItems}" />
       </h:selectOneMenu>
    </div>
    <div class="button">
       <h:commandLink action="#{examEditor.viewExam}"><span>view
    exam</span></h:commandLink>
    </div>
    <div class="button">
       <h:commandLink action="#{examEditor.editExam}" immediate="false"><span>edit
    exam</span></h:commandLink>
    </div>
 </div>
```

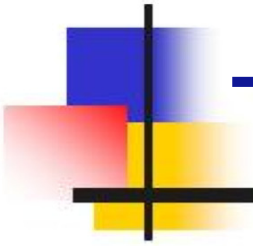
SelectOneMenu: Backing Bean

```
public Long getCategoryOID()
{
    return categoryOID;
}
public void setCategoryOID(Long oid)
{
    this.categoryOID = oid;
}
public List<SelectItem> getCategorySelectItems()
{
    List<Category> categories = this.catSvc.getAllCategories(this.em);
    List<SelectItem> result;
    result = new ArrayList<SelectItem>(categories.size() + 1);
    result.add(new SelectItem("0", "Select an exam category..."));
    for ( Category cat : categories )
    {
        SelectItem item = new SelectItem(cat.getId(), cat.getTitle());
        result.add(item);
    }
    return result;
}
```



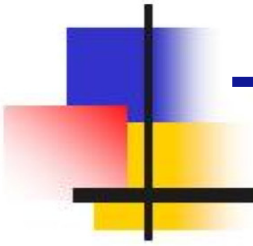
Tomahawk Library

- Compatible with MyFaces and Mojarra (RI)
- Includes the `forceId` attribute
- Provides a Tiles-like facility
- Provides additional validators: email, credit card, regexp matching
- Additional UI widgets



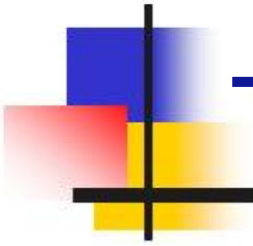
Tomahawk Components

- List: t:dataList
- Table: t:dataTable, t:dataScroller, t:newspaperTable, t:schedule
- Input: t:inputDate, t:inputFileUpload, t:inputHtml, ...
- Layout: t:panelLayout, t:panelTabbedPane, t:panelStack, t:collapsiblePanel
- Document: t:document/Body/Head, t:stylesheet



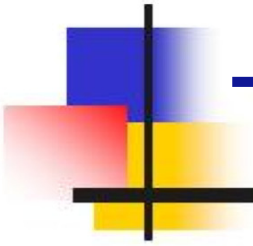
Trinidad Library

- Partial page refresh (PPR)
- Client-side converters and validators
- Advanced client-side dialog boxes
- Advanced navigation tools/components
- File upload
- Additional UI widgets



Trinidad Components

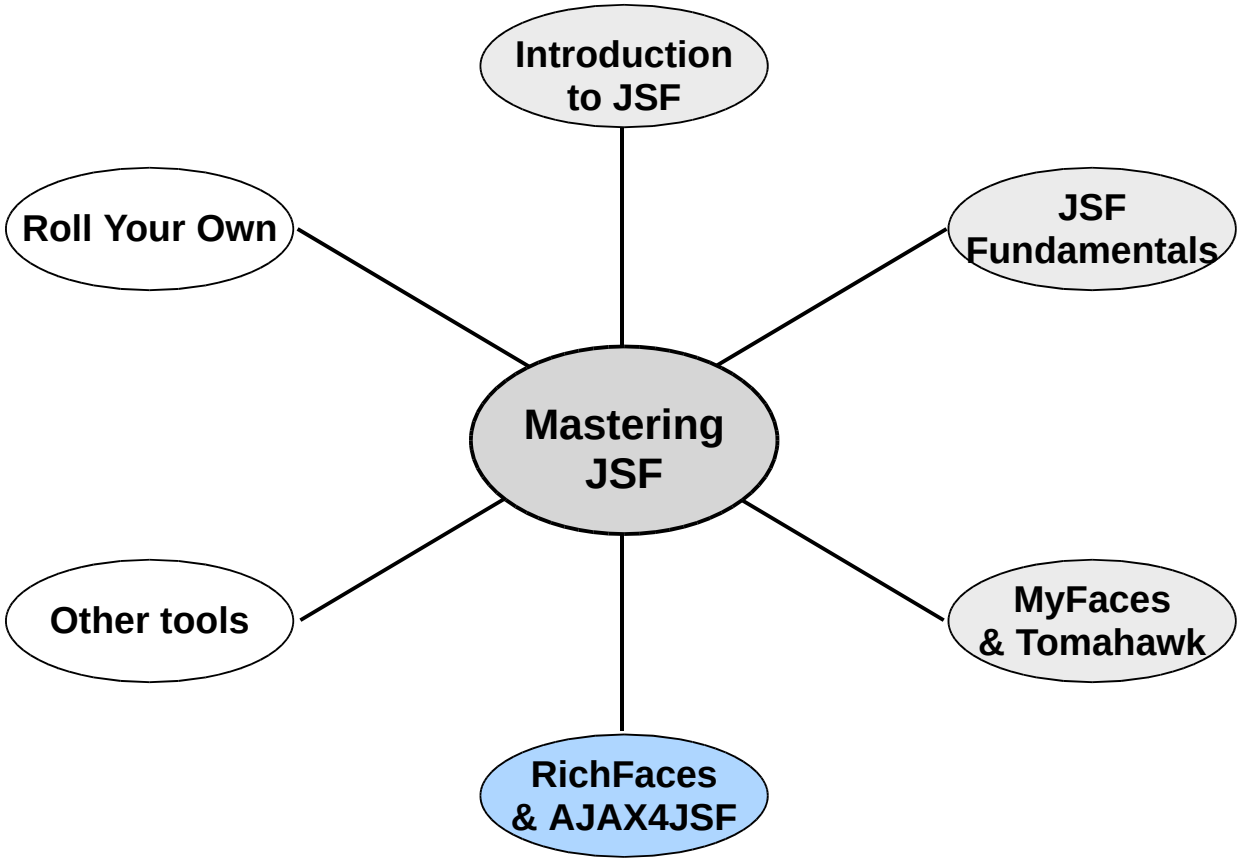
- Table:
  - similar to UIData
  - supports paging, sorting, multiple item selection
- Tree:
  - simple tree and tree-table
- Charts:
  - many types: pie, bar, scatter plot, and so on
  - generates SVG graphics

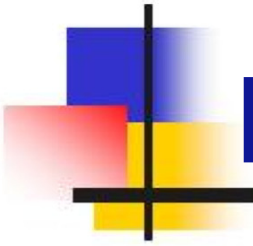


Tobago Library

- Styling by use of “themes”
- Built-in I18N / L10N support
- Controls:
  - table
  - tree
  - sheet
  - tab panel

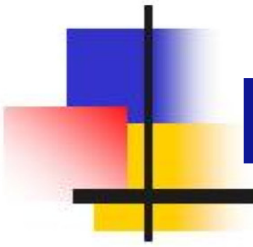
Topics Mind Map





RichFaces Library

- Built-in support for Ajax
- Skinnable styles
- Rich component development kit
- Resource packaging
- Dynamic resources (*e.g.* building button icons on the fly)



RichFaces Components

- Layout: a4j:page, a4j:loadStyle, a4j:loadScript, a4j:include
- UI widgets: rich:calendar, rich:comboBox,
- Tables: rich:dataTable, rich:datascroller, rich:dataFilterSlider, sorting, filtering
- Lists: ordered, unordered, definitions
- Other: drag-n-drop support, effects, panels, trees, and a whole lot more

Ajax Support

- Forms: form submittal via Ajax; rerender portion of screen
- Triggers:
  - Periodic: a4j:poll, a4j:repeat
  - Events: a4j:support
  - JSF: rich:ajaxValidator
- Most components have built-in Ajax support

OnChange: JSP

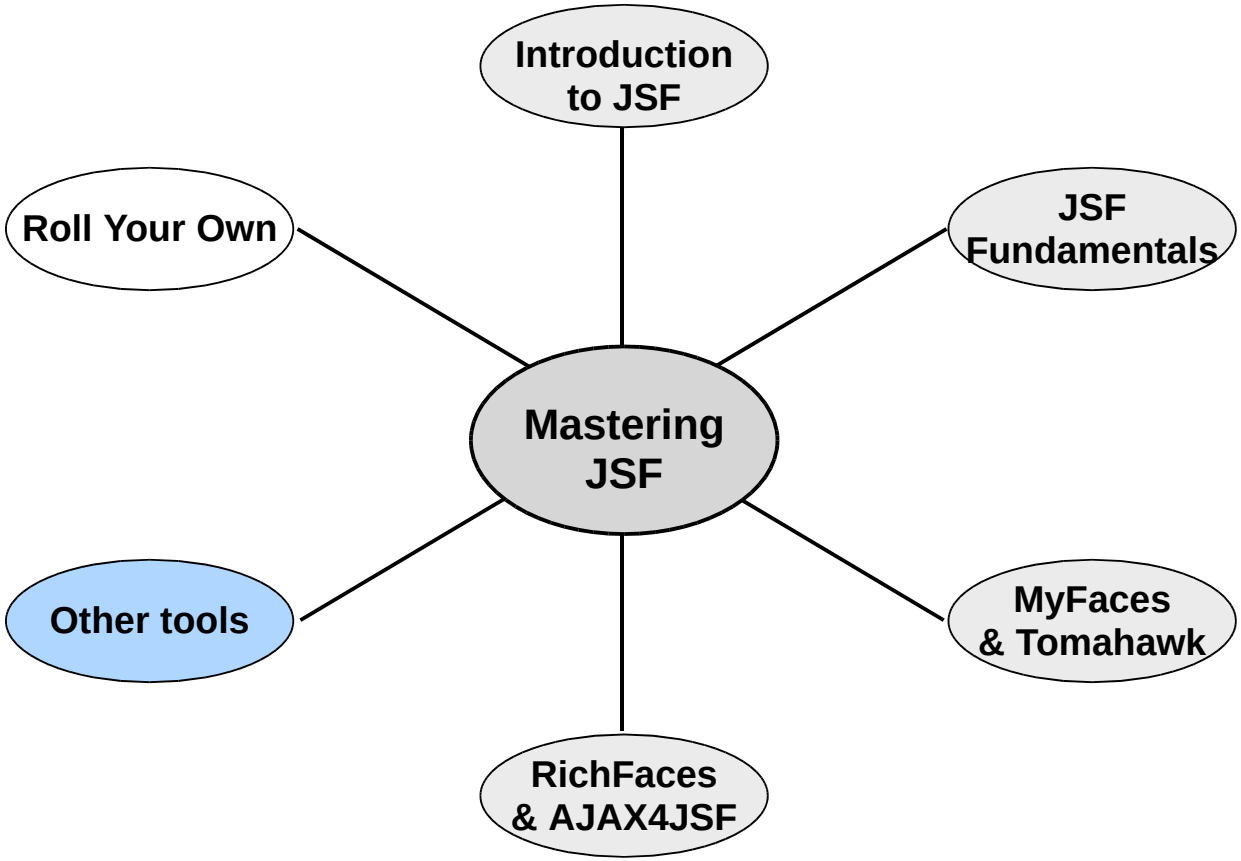
```
<div id="exam_select">
    <div>
       <h:outputLabel for="exam_type_select">Exam category:</h:outputLabel> 
       <h:selectOneMenu value="#{examEditor.categoryOID}" id="exam_type_select">
          <f:selectItems value="#{examEditor.categorySelectItems}" />
          <a4j:support event="onchange" actionListener="#{examEditor.findExams}"
                       reRender="exam_name_select" />
       </h:selectOneMenu>
    </div>
    <div>
       <h:outputLabel for="exam_name_select">Exam title:</h:outputLabel> 
       <h:selectOneMenu value="#{examEditor.examOID}" id="exam_name_select">
          <f:selectItems value="#{examEditor.examSelectItems}" />
       </h:selectOneMenu>
    </div>
    ...
 </div>
```

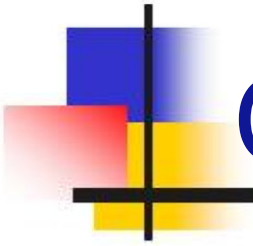
OnChange: Backing Bean

```
public void findExams() {
    this.category = catSvc.getCategoryById(this.categoryOID, this.em);
    if ( this.category != null ) {
        this.exams = this.category.getExams();
    }
}

public List<SelectItem> getExamSelectItems() {
    List<SelectItem> result;
    if ( this.exams == null || this.exams.size() == 0 ) {
        result = new ArrayList<SelectItem>(1);
        result.add(new SelectItem("0", "First select a category above."));
    } else {
        result = new ArrayList<SelectItem>(this.exams.size() + 1);
        result.add(new SelectItem("0", "Select an exam..."));
        for ( CertificatonExam e : this.exams ) {
            SelectItem item = new SelectItem(e.getId(), e.getTitle());
            result.add(item);
        }
    }
    return result;
}
```

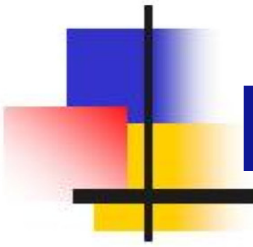
Topics Mind Map





Other component libraries

- JSF has broad support and rich community
  - See JSFCentral.org
  - See JSF Forum on java.net
- There are many open source and COTS libraries



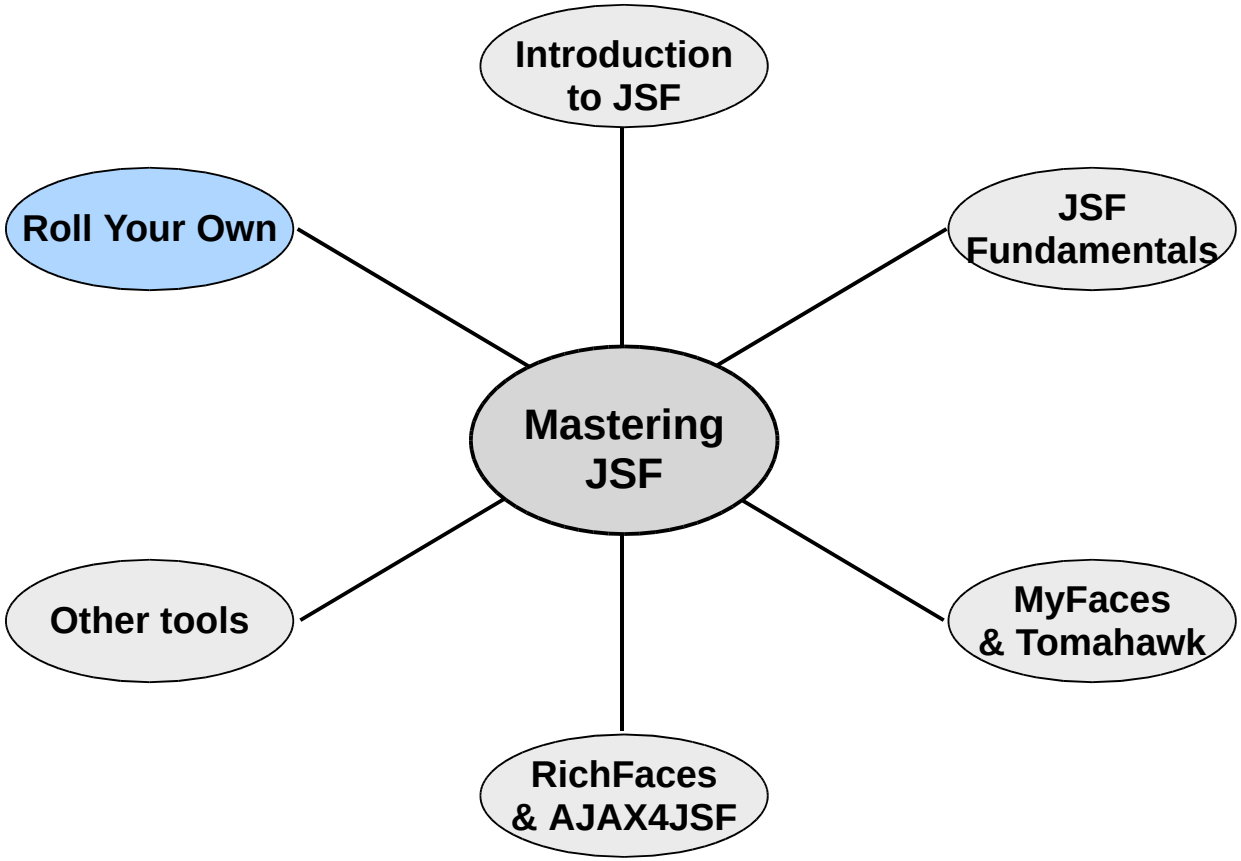
Facelets

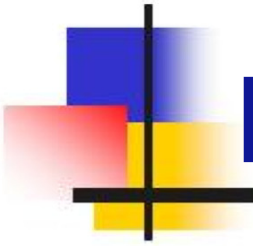
- Built as a replacement for JSP rendering, due to limitations in JSF v1.1
  - eliminates the need for f:verbatim blocks
  - not really needed in JSF v1.2
- Support for tiling, ie. “view composition”
- Support for template-based component creation

Seam

- Integration between UI and Business tiers: JSF <=> EJB3
- Use of annotations simplifies complex deployment descriptors
- Works well with Ajax and Ajax-enabled JSF libraries such as RichFaces
- Conversational scope

Topics Mind Map

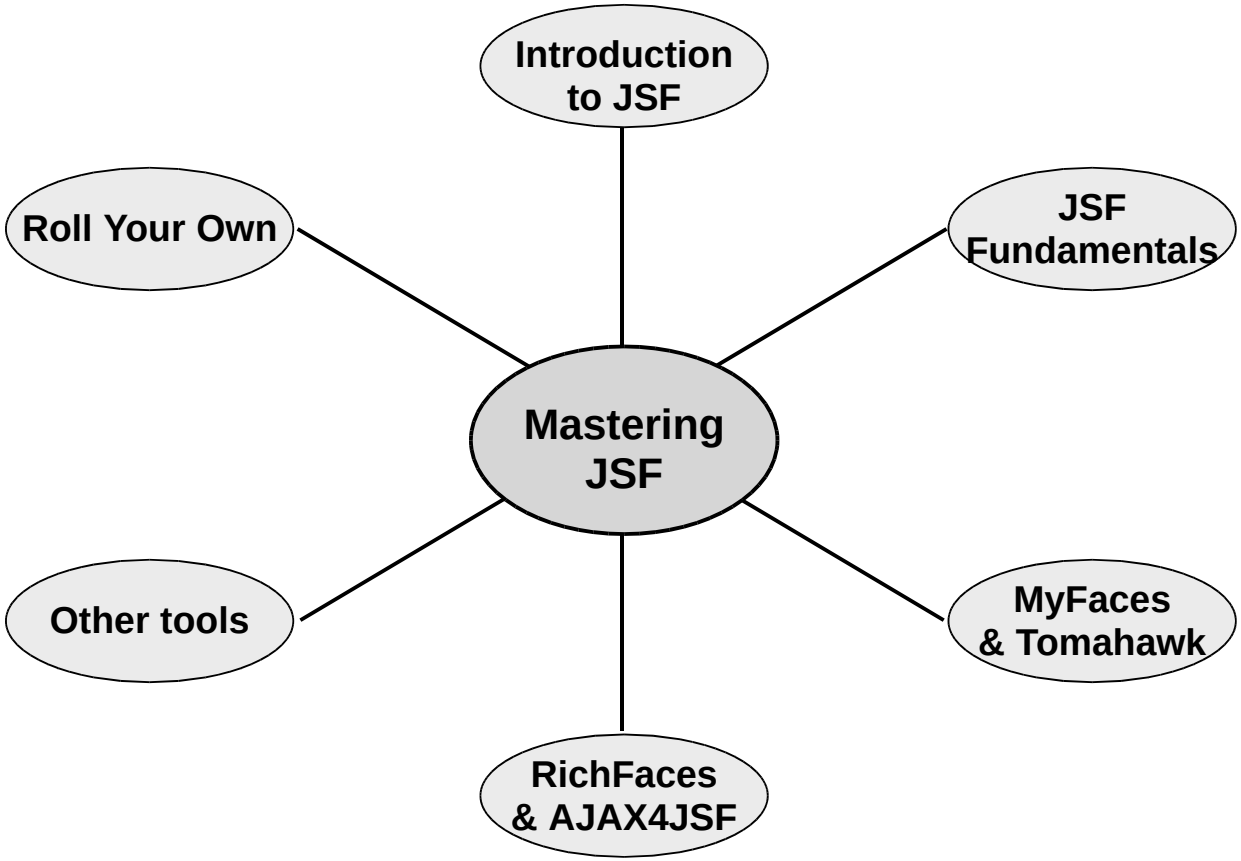


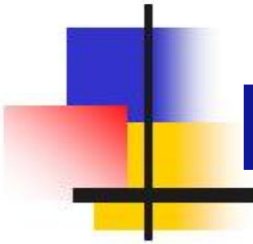


Building Components

- Several open source libraries provide rich APIs for building components
  - MyFaces
  - RichFaces
- Consider the need to inject component-specific resources
  - styles
  - JavaScript
  - media

Q & A





Resources

- JavaServer Faces in Action (Manning)
- JavaServer Faces:
  - java.sun.com
  - java.net
  - JSF Spec (JSR #252)
  - JSFCentral.org
  - JSF tutorials
- Component Libraries:
  - MyFaces (for JSF v1.2)
  - RichFaces
- Other tools:
  - Facelets
  - Seam Framework