JSF + Bootstrap 3
Building a more responsive design
Where are we headed? Why?

http://www.wired.com/2015/05/google-now-io/
Agenda

Introduction

- Java Server Faces
- Bootstrap 3
  - Bootstrap CSS grid and other form classes
- Font Awesome
  - Best practice to implement in JSF

Leveraging Bootstrap with JSF

- Bootstrap validation using component bindings
- Responsive Bootstrap modals

What are they? How do you use them together?
Example App

KEYHOLE SOFTWARE  Agility. Delivered.
## Create Tax

**Example Content**


Donec posuere, lacinus sed ornare consequat, mi mi pretium elit, sed gravida enim felis vitae leo. Nulla diam libero, lacinia vel consectetur a, dapibus in lorem. Quisque malesuada nec at sapien eleifend scelerisque.

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associated State</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

[Create](#)
Cover best practices

- Bootstrap validation using component bindings
- Font awesome icons usage
- Bootstrap CSS grid and other form classes
- Responsive Bootstrap modals

Lots of other JSF component libraries

- Primefaces
- Icefaces
- Richfaces
Example App

Software Used

- Netbeans 8
- TomEE 1.7
- MySQL
- JSF 2.0
- Deltaspike 1.4 (used for viewscoped)
- Bootstrap 3
- Font Awesome 4.2
Java specification for building component-based user interfaces for web applications. It gives you the tools to communicate with java backing beans. JSF has all the javascript behind the scenes, creates the endpoints from JSF managed beans, and wires it all together.

A brief history

○ JSF 1.0 - 2004
  ■ Starting point, major drawback that got alot of criticism was no inline html tags. In the next release JSF 1.1 bug release the ability was added using the `<f:verbatim>`

○ JSF 1.2 - 2006
  ■ allowed for plain HTML tags to be inline without the verbatim tag
  ■ JSF from JSP completely detached
  ■ lacked conversation scope, which caused many developers to abuse session scope

○ JSF 2.0 - 2009
  ■ ajax support, what all modern applications are using now (that or web sockets)
  ■ facelets introduced to create composite components - reusable code

The weight is put on the server “Java SERVER faces” whereas newer javascript libraries put the weight on the client.
What is Bootstrap?

Bootstrap is a out of the box, ready to use CSS framework. Very little customization should be needed. The framework makes core decisions for you by bundling helpful premade css components; forms, buttons, menus, etc. It was built with mobile-first in mind.

What are some benefits?

- Flat trendy design
  - Makes design simpler, no longer have to fill your time with complex layouts
  - Improves performance when browser has to draw all elements to the screen

- Grid system
  - It allows you to target different screen densities using a 4 level grid system: http://getbootstrap.com/css/#grid
<table>
<thead>
<tr>
<th></th>
<th>Extra small devices Phones (&lt;768px)</th>
<th>Small devices Tablets (≥768px)</th>
<th>Medium devices Desktops (≥992px)</th>
<th>Large devices Desktops (≥1200px)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid behavior</td>
<td>Horizontal at all times</td>
<td>Collapsed to start, horizontal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>above breakpoints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container width</td>
<td>None (auto)</td>
<td>750px</td>
<td>970px</td>
<td>1170px</td>
</tr>
<tr>
<td>Class prefix</td>
<td>.col-xs-</td>
<td>.col-sm-</td>
<td>.col-md-</td>
<td>.col-lg-</td>
</tr>
<tr>
<td># of columns</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column width</td>
<td>Auto</td>
<td>~62px</td>
<td>~81px</td>
<td>~97px</td>
</tr>
<tr>
<td>Gutter width</td>
<td>30px (15px on each side of a column)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nestable</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offsets</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column ordering</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Responsive Design

Row with 2 columns divided equally

```html
<div class="row">
  <div class="col-md-6">left column</div>
  <div class="col-md-6">right column</div>
</div>
```

Row with 2 columns divided in a 1:3 ratio

```html
<div class="row">
  <div class="col-md-3">left column</div>
  <div class="col-md-9">right column</div>
</div>
```
Create Tax

Example Content

Donec posuere, lacus sed ornare consequat, mi mi pretium elit, sed gravida enim feles vitae leo. Nulla diam libero, iaculis vel consectetur a, dapibus in lorem. Quisque malesuada neque at sapien eleifend scelerisque.

Status
 UserType Single

Min
$0

Max
$10000

Rate
0

Associated State
State Name

Create
Salary After Taxes

Create Tax

COL-MD-3

COL-MD-9

Example Content


Donec posuere, lacus sed ornare consequat, mi mi pretium elit, sed gravida enim feles vitae leo. Nulla diam libero, lacus vel consectetur a, dapibus in lorem. Quisque malesuada neque at sapien eleifend scelerisque.

Status

Single

Min

$
Font Awesome

What is FA FA?

An open source icon font kit. It does not have to be used with Bootstrap, but it most commonly is. It is used so much, that the standard glyphicons that come with Bootstrap are being removed in Bootstrap 4 because they are not being used.

**Facts**

- 519 pictographic icons
- Easily scalable vector graphics

**Traditional Usage**

```html
<i class="fa fa-check"></i>
```

This becomes annoying when using JSF

**Ugly Code**

```html
<h:commandLink action="#{myBean.backing}" styleClass="btn btn-default"><i class="fa fa-check"></i> Create</h:commandLink>
```
Cheat Sheet

Using the content property generated content can placed :before and :after pseudo-elements

http://fortawesome.github.io/Font-Awesome/cheatsheet/
What is FA FA?

CSS Submit Class

```css
.act-submit:before {
    font-family: FontAwesome;
    display: inline-block;
    content: "\f00c";
    padding-right: .4em;
}
```

Ugly Code

```xml
<h:commandLink action="#{myBean.backing}" styleClass="btn btn-default"><i class="fa fa-check"></i> Create</h:commandLink>
```

New Code

```xml
<h:commandLink value="Create" action="#{myBean.backing}" styleClass="btn btn-default act-submit"/>
```
Agenda Update

Introduction

- Java Server Faces
- Bootstrap 3
  - Bootstrap CSS grid and other form classes
- Font Awesome
  - Best practice to implement in JSF

Leveraging Bootstrap with JSF

- Bootstrap validation using component bindings
- Responsive Bootstrap modals
JSF provides a simple way to access component properties. Gives you access #{mybinding.clientId}, #{mybinding.value}, #{mybinding.valid}, #{mybinding.submittedValue} properties in the EL scope.

2 ways to use component bindings
1. Bind the component to the current view scope
2. Bind the component to a UIComponent object on the backing bean, then perform validation / manipulation

Typical Usage (Primefaces)

```
<p:outputLabel for="firstname" value="Firstname:" /></p:inputText

<p:inputText id="firstname" value="#{userView.firstname}" required="true" label="Firstname">
  <f:validateLength minimum="2" />
</p:inputText>
```

>This has too much magic for me...<
Using Bootstrap 3 classes

```
<div class="form-group #{!name.valid ? 'has-error' : 'none'}">
  <label for="name">Enter name</label>
  <h:inputText value="#{myController.name}" binding="#{name}" id="name" styleClass="form-control">
    <f:validateLength minimum="4" maximum="6" />
  </h:inputText>
</div>
```

Bootstrap CSS

```
.form-group: Provides optimum spacing for label / inputs
.has-error: validation style for a error. Also has other validation states such as .has-warning, and .has-success
.form-control: Provides formatting to <input>, <textarea>, and <select> elements.
```
Example: Binding to UIComponent

Location.xhtml page

```xml
<h:selectOneMenu binding="#{state}" value="#{myBean.selectedState}"
    >
    <f:selectItems value="#{myBean.states}" />
</h:selectOneMenu>
<h:inputText value="#{myBean.city}"
    >
    <f:attribute name="state" value="#{state.value}" />
    <f:validator validatorId="locationValidator" />
</h:inputText>
```

LocationValidator.java

```java
@FacesValidator("locationValidator")
public class LocationValidator implements Validator {
    @Override
    public void validate(FacesContext context, UIComponent component, Object value) {
        Object item = component.getAttributes().get("state");
    }
}
```
A composite component is a special type of template that acts as a component. It allows for code to easily be reused.

1. Define a namespace with the `xmlns:

   $$\text{xmlns:sat=\"http://java.sun.com/jsf/composite/salaryaftertaxes\"}$$

2. Create composite component in resources folder

3. Implement the composite component

4. Use the composite component on the view

   $$\text{<sat:statesModal id=\"statePreview\" contentId=\"content\" modalTitle=\"Preview State\" selectedState=\"#{taxesViewController.current.state}\"/>}$$
Implement the composite component

```xml
<composite:interface>
    <composite:attribute name="contentId" required="true"/>
    <composite:attribute name="selectedState" required="true"/>
    <composite:attribute name="modalTitle" required="true"/>
</composite:interface>

<composite:implementation>
    // .. Modal code omitted
</composite:implementation>
```
Bootstrap HTML Code

```html
<div class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        <p>One fine body &hellip;</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

---

Creating a Modal

Bootstrap + JSF

Agility. Delivered.
Bootstrap Usage

```html
<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
  Launch demo modal
</button>
```

Bootstrap uses custom data attributes `data-toggle` or `data-target`.
- `Data-toggle` tells Bootstrap JS what it is going to do
- `Data-target` tells Bootstrap JS what element or ID to perform on

Javascript

```
$('modal-content').modal('show');
```
Why would I use JSF?

**Pros:**
- Large community
- Integrates well with Java EE
- Ajax component re rendering is simple and easy

**Cons:**
- Server side presentation layer
- Component design hides functionality from developers; creates a layer of abstraction which can lead to overhead.

Side Note: Client Side Validation based on JSR-303 Bean Validation specification has been implemented in Primefaces and Richfaces.

Bottom Line: JSF provides a quick and easy way to implement a UI given your requirements fit within the abilities of the provided components.
Questions?