QUICK RECAP

A reminder of what we have done during Lecture 07

LAST TIME...

- Web app design principles
  - Why should they be applied?
- Mobile First
  - What is it?
  - What are the benefits and drawbacks?
- Responsive Design
  - What is it?
  - How to use it with Vaadin?

DECLARATIVE UI AND VAADIN DESIGNER

Development of Modern Web Applications (with Vaadin)

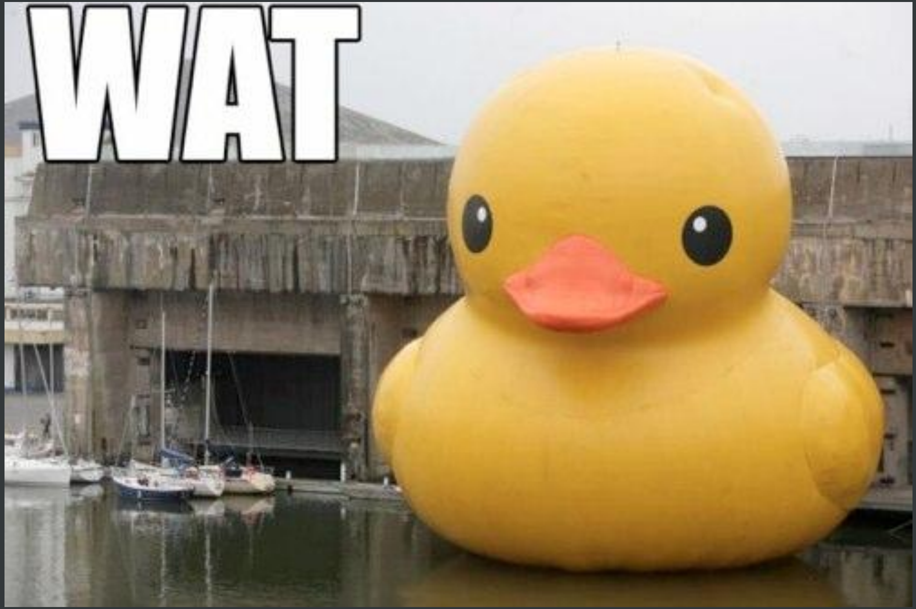
Lecture 08

OVERVIEW

- Declarative UI
- Vaadin Designer

- Shoutbox continues
  - Going declarative

DECLARATIVE UI



PROGRAMMING LANGUAGES

Imperative

- Almost all languages you know
- A program is a **solution** to the problem
  - **How** to do things

Declarative

- Expresses a logic of computations without a control flow
  - Says wiki
- A program is a **description** of a solution to the problem
  - **What** needs to be done

DECLARATIVE UI

- Focuses on what UI looks like
  - UI components and their hierarchy
  - Properties, names, etc.
- Ignores how UI works
  - No event handling
  - No interaction
  - This is left to the real code
- This is not new
  - Visual Basic is from 1991
  - Delphi dates back to 1995
    - And still rocks in 2016 ☺

BENEFITS

- More focus on code
  - Implement behaviour
  - Ignore stuff that is not relevant
- Reusable
  - Declared UIs can be used with a different code
  - Code can use a different UI and still work
- Separation of concerns
  - UI no longer affects the code
  - Can be designed independently
    - Some common interface must still be agreed

DRAWBACKS

- Overhead and performance
  - The declared UI must be generated somehow
- Code pollution
  - The project now features yet another language
    - The declarative UI language
  - Reduces readability of the code

VAADIN DECLARATIVE FORMAT

Declaring UI elements

IN BRIEF

- Custom elements in HTML file
  - Must have a single root
  - `ComponentName` → `vaadin-component-name`
  - `setFoo("bar")` → `foo="bar"`
    - Limited subset of attribute types is supported
- Supported by all `Components`
  - Completely transparent to other components
  - Extend a component that matches the design's root element
  - `@DesignRoot`
    `Design.read("file.html", this);`
    - The file is in the resources

EXAMPLE

Java

```
VerticalLayout vertical = new VerticalLayout();
vertical.addComponent(
                  new TextField("Name"));
vertical.addComponent(
                  new TextField("Street"));
vertical.addComponent(
                  new TextField("Code"));
```

Declarative

```
<vaadin-vertical-layout>
  <vaadin-text-field
       caption="Name"/>
  <vaadin-text-field
       caption="Street"/>
  <vaadin-text-field
        caption="Code"/>
</vaadin-vertical-layout>
```

USING CUSTOM COMPONENTS

Java

```java
package org.vaadin.miki.flatselect;

public class FlatSelect
           extends AbstractSelect {
  public FlatSelect() {
    ...
  }
}
```

Declarative

```html
<!DOCTYPE html>
<html>
  <head>
   <meta name="package-mapping"
      content=
         "fs:org.vaadin.miki.flatselect"
   />
  </head>
  <body>
    <vaadin-css-layout>
      <fs-flat-select/>
    </vaadin-css-layout>
  </body>
</html>
```

PROPERTIES VS ATTRIBUTES

- Inline data (inside the tag) is component-specific
  - https://vaadin.com/api for details
- `setPropertyName` ➔ `property-name`
  - `String`, numbers, boolean or enum
    - Sadly, no objects are supported
- `:property-name` is called on a container
  - Or rather, a containing component
    - Most commonly, a layout
  - Current component is then passed as a parameter

REFERENCING COMPONENTS

Declarative

```
<vaadin-vertical-layout>
  <vaadin-tree
         _id = "mytree"
     caption = "My Tree"
  />
</vaadin-vertical-layout>
```

Java

```
@DesignRoot
public class MyViewDesign
   extends VerticalLayout
{
 protected Tree mytree;
 // note visibility
 ...
}
```

SUMMARY OF VAADIN DECLARATIVE FORMAT

- Event handling is not covered
  - UI is only declared, after all
  - Must be added to the corresponding Java file
- Work overhead
- Work division
- Hides details about UI structure
  - Explicit call to `Design.read`

DEMO!

Shoutbox step 11
http://github.com/vaadin-miki/shoutbox

end branch: `step-11`

THE PLAN

- Let’s add some declarative UI
  - In fact, almost all of it can be declared
- An extension of a CSS Layout

```
<html>
 <head>
  <meta charset="UTF-8"
        name="package-mapping"
        content="fs:org.vaadin.miki.flatselect">
 </head>
 <body>
  <vaadin-css-layout
                 style-name="messages shoutbox"
                 size-full>
   <vaadin-horizontal-layout
                 style-name="card entry-bar"
                 spacing width-full margin
                 _id="top">
    <vaadin-text-field
                 caption="You were saying?"
                 input-prompt="(type something)"
                 style-name="large borderless"
                 width-full _id="text" :expand="0.6">
    </vaadin-text-field>
    <vaadin-button
              style-name="large friendly shout-button"
              width-full plain-text _id="button"
              :middle :center :expand="0.2">
      Shout!
    </vaadin-button>
   </vaadin-horizontal-layout>
   <fs-flat-select caption="Rooms:"
                  style-name="rooms"
                  width-full _id="roomSelect">
   </fs-flat-select>
   <vaadin-panel style-name="viewport" size-full
                 _id="placeholder"></vaadin-panel>
  </vaadin-css-layout>
 </body>
</html>
```

← SHOULD LOOK LIKE THIS, MORE OR LESS

Easy to read, isn't it?
    No sarcasm here. It is easy to read.

Now you know how it should look like, let's not do it.

Surely there must be a better way to do it.

VAADIN DESIGNER

Drag and drop your views

WHAT IS IT?

- Drag-and-drop UI composer
  - WYSIAWYG editor
  - Backend is the HTML file
    - Java file is automatically generated
      - Do not edit!
- An extra (paid) feature of Vaadin
  - https://vaadin.com/designer
    - Trial version available
- A plugin to Eclipse and IntelliJ

WHY IS IT GOOD?

- Instant feedback on the design
  - Works with non-technical people
  - Sets up a local preview server
    - Reflects changes on the fly
- Easy to create
- Easy to include in the development process
  - Wireframes

(SOME OF THE) LIMITATIONS

- Reasonable amount of features
  - Covers only the simplest cases
- No real support for custom components
- No support for data
- Only simple properties are editable
- No support for events
  - Design decision

IS IT NEEDED?

- In a simple project, most likely not
- In an individual project, most likely not
- In a single-view project, most likely not

DEMO!

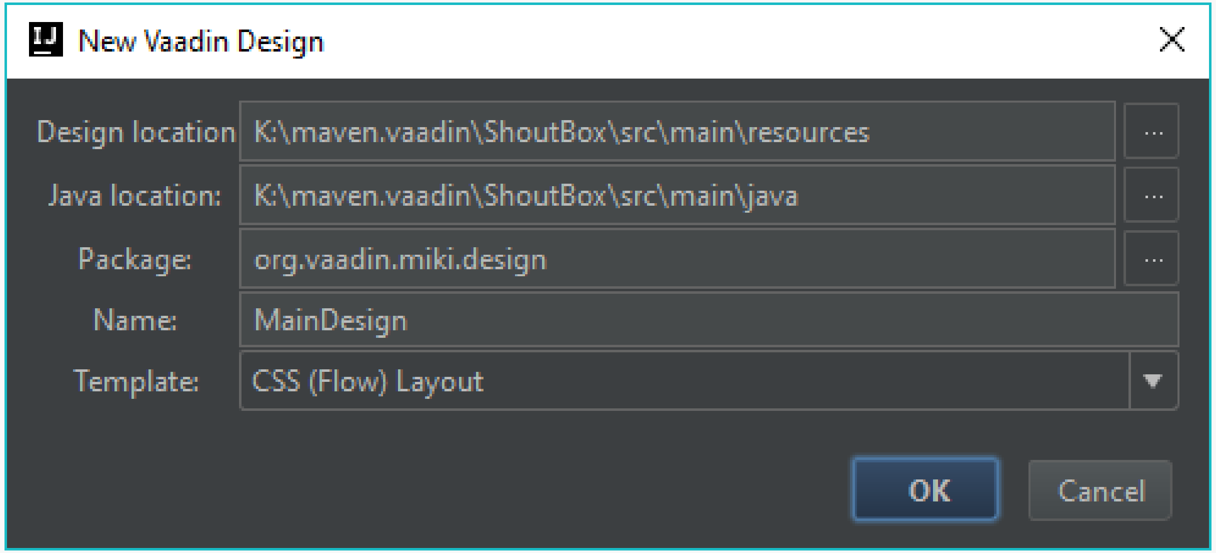
Shoutbox step 11
http://github.com/vaadin-miki/shoutbox

end branch: `step-11`

THE PLAN

- Let’s ~~add~~ create a declarative UI with Designer
  - And add event binding manually

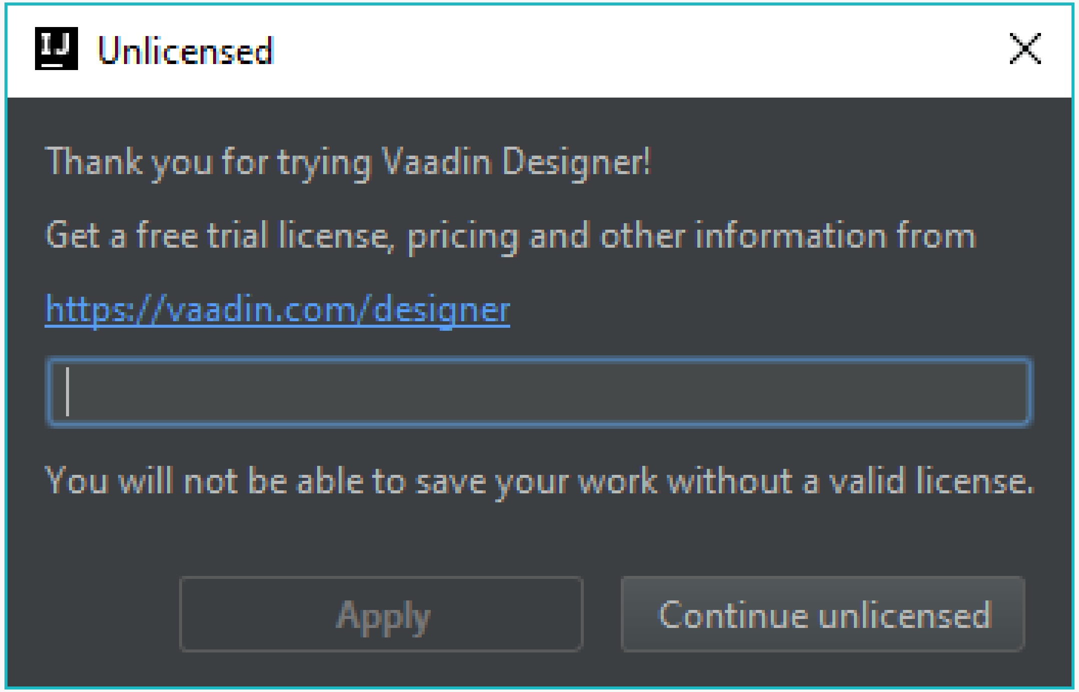
NEW → VAADIN DESIGN



New Vaadin Design

Design location	K:\maven.vaadin\ShoutBox\src\main\resources
Java location:	K:\maven.vaadin\ShoutBox\src\main\java
Package:	org.vaadin.miki.design
Name:	MainDesign
Template:	CSS (Flow) Layout

REGISTER


Unlicensed

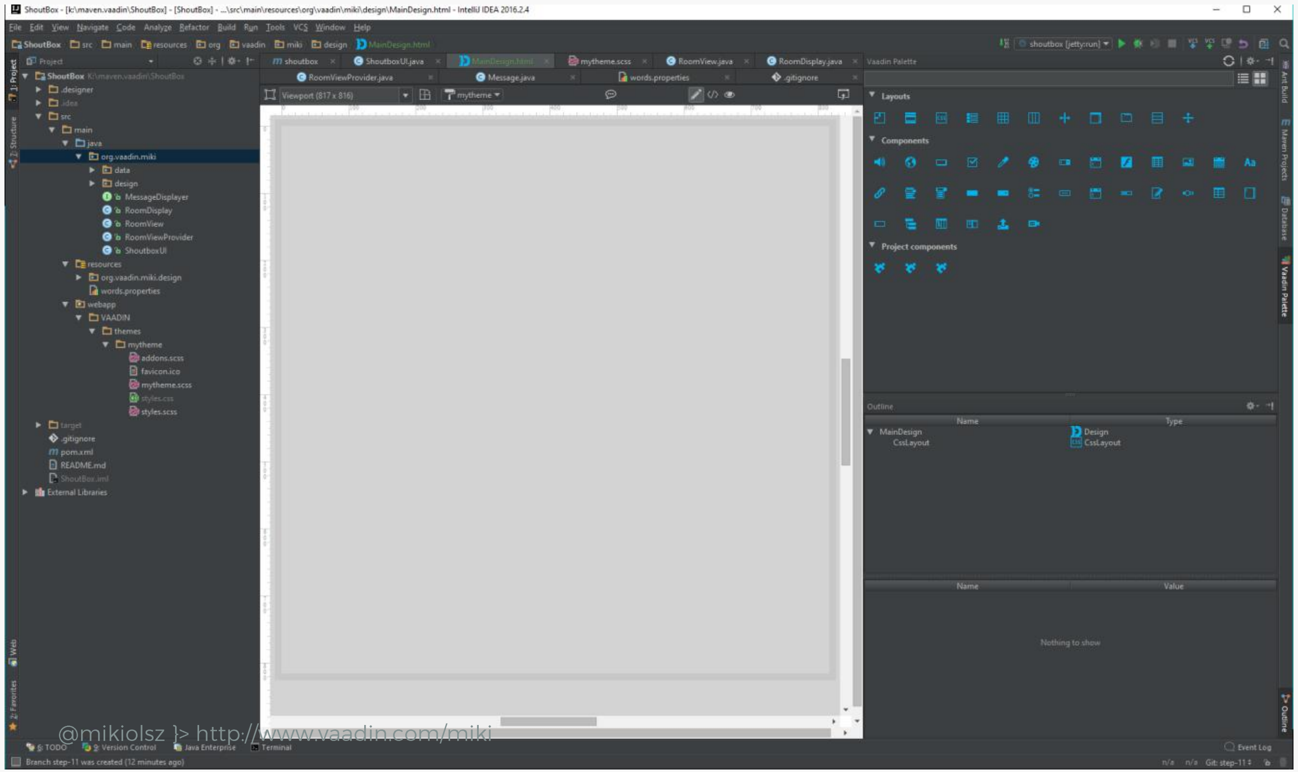
Thank you for trying Vaadin Designer!

Get a free trial license, pricing and other information from

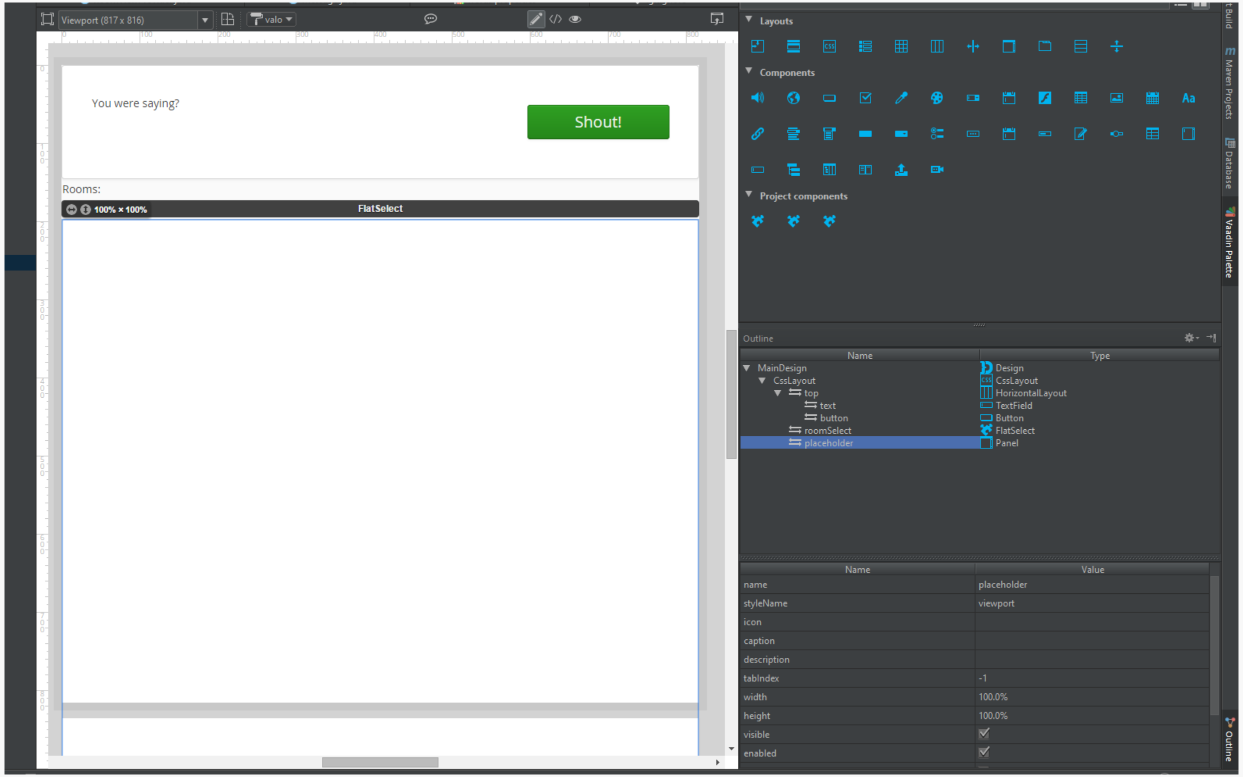
https://vaadin.com/designer

You will not be able to save your work without a valid license.

DESIGN FOR NO DATA? ☺



AFTER A WHILE

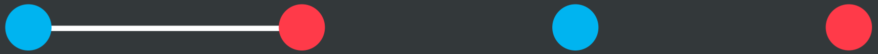


Name	Type
MainDesign	Design
CssLayout	CssLayout
top	HorizontalLayout
text	TextField
button	Button
roomSelect	FlatSelect
placeholder	Panel

Name	Value
name	placeholder
styleName	viewport
icon	
caption	
description	
tabIndex	-1
width	100.0%
height	100.0%
visible	✓
enabled	✓

BINDING DESIGNS TO CODE

Coupling declarations with actions



EXTEND THE DESIGN

- Design’s constructor reads the html file
  - And creates the UI components
- Components with _id are protected
  - Thus available in the subclass
- Implement extra stuff if needed

THE DESIGNER APPROACH

Pros

- Clearly separated UI from code
  - Parallel design
  - Separation of concerns
- Reusable designs
- Reusable code
  - To some extent

Cons

- More files to keep track of
- Extra language
- Yet another tool
- Not everything can be done in parallel
- Fields are not private
  - Workarounds possible

SUMMARY

What did we do today

LESSONS OF TODAY (HOPEFULLY)

- Declarative UI
  - What is it?
- Vaadin Declarative Syntax
  - How to declare and use designs?
  - What are the advantages and drawbacks?
- Vaadin Designer
  - Do I need it?

COMING UP NEXT

- Web Components and Vaadin Elements
- Quality, debugging and testing
- Progressive Web Applications

THE END

SUGGESTIONS?
QUESTIONS?

miki@vaadin.com
t: @mikiolsz