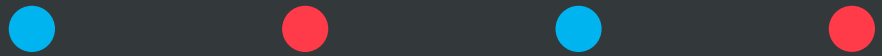
QUICK RECAP

A reminder of what we have done during Lecture 03



LAST TIME...

- Vaadin Data Model
  - What are the key elements?
  - How to link it to components?
- Events
  - How to catch events?
- Notifications and windows
  - How to display annoying popups?
- Server Push

VAADIN FRAMEWORK: STYLING, LAYOUTS AND NAVIGATION

Development of Modern Web Applications (with Vaadin)

Lecture 04

vaadin

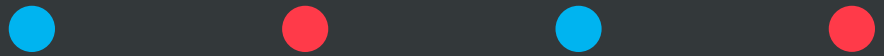
OVERVIEW

- Layouts
- Styling
- CSS essentials
- Navigation

- Shoutbox app continues

BASIC LAYOUTS

Suitable in most of the (simple) cases



Overview

Layouts

- Arrange components
  - No other purpose or functionality
- No user interaction

Linear arrangement

- com.vaadin.ui.HorizontalLayout
  - From left to right
- com.vaadin.ui.VerticalLayout
  - From top to bottom

Layouts are components

- Common properties
- Layout-specific properties
- Cell-based rendering
  - For those listed here, that is

Arrangement in grid

- com.vaadin.ui.GridLayout
- No support for overlapping
  - But supports cell span

PROPERTY: SPACING

- Component separator
  - HorizontalLayout – horizontal spacing
  - VerticalLayout – vertical spacing
  - GridLayout – both
- Boolean flag
  - Size of spacing cannot be decided in the code
- CSS rules
  - .v-{layout-name}-spacing-on

PROPERTY: MARGIN

- Space around the layout
- Boolean flag
  - Possible to set individual margins as well
  - Default values exist in themes
- CSS rules
  - In which you should set up **padding**
  - .v-{layout-name}-margin
  - .v-{layout-name}-margin-{side}

PROPERTY: ALIGNMENT

- Location of a component in its cell
- com.vaadin.ui.Alignment

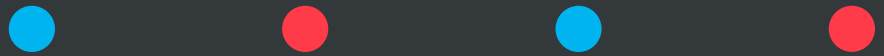
TOP_LEFT	TOP_CENTER	TOP_RIGHT
MIDDLE_LEFT	MIDDLE_CENTER	MIDDLE_RIGHT
BOTTOM_LEFT	BOTTOM_CENTER	BOTTOM_RIGHT

PROPERTY: EXPAND RATIO

- Specifies rate of cell expansion
  - It is not ~~the spoon that bends~~ component that expands
    - It is the cell
- When component size is relative
  - → Component expands
  - Best results with 100%
- When component size is absolute / undefined
  - → Empty space expands
  - Combine with alignment

ADVANCED LAYOUTS

Fine-tune your layout



OVERVIEW

- Arranging components
  - No other purpose or functionality
  - No user interaction
- CSS-based rendering
  - Less options in the API
  - More flexibility and possibilities
- Components
  - Common properties
  - Layout-specific properties

COM.VAADIN.UI.
ABSOLUTELAYOUT

- Absolute positioning
  - CSS-like string
    - left, right, top, bottom
  - Overlapping supported
    - z-index
- Supports relative values
  - Reference point – layout size
- Displays captions above components

```
Image img = new Image(
  "Ya rly!",
  new ExternalResource(
    "http://bit.ly/9o3Qd2")
);

AbsoluteLayout layout =
      new AbsoluteLayout();
layout.addComponent(
  img,
 "left: 30px;"+
 "right: 10px;"+
 "top: 10%;"+
 "bottom: 15%;");
```

COM.VAADIN.UI. ABSOLUTELAYOUT

CSS rules:

.v-absolutelayout

.v-caption

.v-absolutelayout-wrapper

(for each component)

COM.VAADIN.UI.CSSLAYOUT

- Horizontal alignment with wrapping
- Fastest layout
  - No built-in dynamic logic
- Support for custom CSS for each component
  - Not straightforward
  - Inline style
  - Overrides any other styling
- Relies on browsers
  - All browsers are compatible with CSS standard
  - Some of them are more compatible than others

```
Label label = new Label("Bold4life");

// getCss() is protected
// do not abuse this hack
CssLayout layout =
  new CssLayout() {
    protected String getCss(Component c)
    {
      return "font-weight: bold;";
    }
  };

// each component added to the layout
// will have bold font
// and it is impossible to change it
layout.addComponent(label);
```

COM.VAADIN.UI. CSSLAYOUT

CSS rules:

.v-csslayout

.v-csslayout-margin (always present)

.v-csslayout-container (contains s)

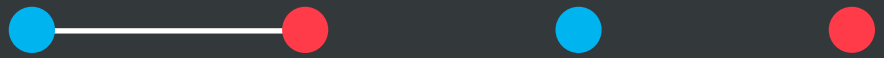
COM.VAADIN.UI.
CUSTOMLAYOUT

- XHTML templates
  - Name- and theme-based template location
    - `WebContent/VAADIN/themes/{theme}/{layout}.html`
  - Anything from an `InputStream`
- Placeholders
  - XHTML
    - `<div location="component_id"></div>`
  - Code
    - `addComponent(component, "component_id");`
- No CSS rules
  - Use template and style directly

DEMO!

Shoutbox step 5
http://github.com/vaadin-miki/shoutbox

end branch: `step-05`



THE PLAN

- Modify the layout
  - Text box and button horizontally at the top
  - Everything else in a panel at the bottom
  - Reverse the order of labels (newest first)
- No styling at this point

THEMES

Reusing the look-and-feel

THEME CONTENTS

- Styles
  - SASS
    - The CSS that should have been
    - Needs compilation to plain CSS
- Any resources
- Layout templates

THE STYLE

- `WebContent/VAADIN/themes/{theme}/{theme}.scss`
- File name and path are fixed
- Default based on Valo
  - https://vaadin.com/valo
- Customisation
  - Use variables mentioned in the generated .scss
  - Write custom rules

Other contents

Resources

- `WebContent/VAADIN/themes/{theme}/{resource-type}/*`
- Name of the directory is recommended
  - `img` for images and icons

Templates

- `WebContent/VAADIN/themes/{theme}/layouts/*`
- Name of the directory is recommended
- Templates to be used with `CustomLayout`

LINKING THEME TO THE APP

- Annotate the UI class
  - `@Theme("theme-name")`
- Add styles to elements
  - `foo.addStyleName("my-style");`
- Add formatting rules in SCSS
  - ```
    .my-style {
        text-decoration: underline;
    }
    ```
- Use built-in style name helpers
  - `foo.addStyleName(ValoTheme.BUTTON_LINK)`
    ```

CSS

A very basic introduction to Cascading Style Sheets

OVERVIEW

Cascading Style Sheets (CSS) is
a style sheet language
used for
describing the presentation semantics
(the look and formatting)
of a document written in a markup language.

from Wikipedia

WHAT DOES THAT MEAN?

- Presentation
  - Fonts
  - Colours
  - Images
  - Layouts
- Goal
  - Separate styling from contents
  - Different screen readers
    - Better accessibility
  - Different markup languages
    - XHTML is a subset of XML
    - Any XML can be styled
- Cross-browser standard, in theory

BASIC SYNTAX

```
rule, rule, ... {
  property: value;
  property: value;
  ...
}
...

/* SCSS allows $variables
   and {nested: rules;}          */
```

Selector rules

Css	XHTML
`element`	`<element>...`
`.klass`	`<a class="klass">...`
`#myId`	`<div id="myId">...`
`foo.klass#myId`	`<foo class="klass" id="myId">...`
`parent child`	`<parent><a><child>...`
`parent > child`	`<parent><child>...`
`a + b`	`<a></a><b>...`
`p[title]`	`<p title="text">...`
`p[title="foo"]`	`<p title="foo">...`
`p[title~="foo"]`	`<p title="foo bar">...`

Selector pseudo-class

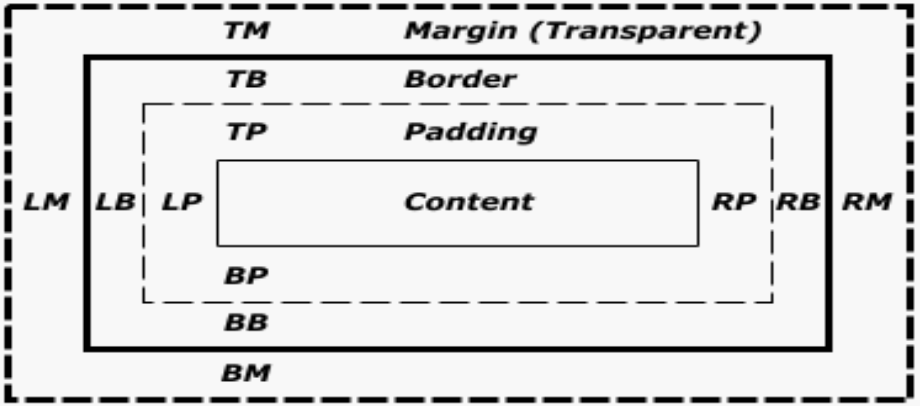
CSS	Explanation
`:link`	• Unvisited hyperlink
`:visited`	• Visited hyperlink
`:active`	• Active hyperlink
`:hover`	• Element under the cursor
`:focus`	• Element accepting keyboard input

Property values

CSS	Explanation
em	Font-size of current font
ex	Height of letter ‘x’ in current font
px	Pixels, device-dependent
in	Inches, 1” = 25.4mm
cm	Centimetres
mm	Millimetres
pt	Points, 1/72in
pc	Picas, 12pt = 1/6in
(no unit)	Zero or any number
%	Relative value
"text" / 'text'	String value
ms	Milliseconds
s	Seconds
url("url" / 'url' / url)	External resource given by URL
deg / grad / rad	Angle in degrees, grads or radians
(colour)	Keyword, rgb(r, g, b) or #RRGGBB

THE BOX MODEL

- W3C CSS2.1 Specification
  - Each edge can be styled separately
  - Margins are always transparent



Must-know properties (1)

Rule	Explanation

`margin`
`padding`
`border`
	• Directly related to the box model
	• Separate for each direction
	  • margin-left
	  • padding-right
	  • border-bottom
	• Various border styles

Must-know properties (2)

Rule	Explanation
`color`	• Colour of main text…
`background-color`	• …and background (and padding)
`font-family`	• Name, name, name…
`font-size`	• Size of the font
`font-weight`	• Bold
`font-style`	• Italic
`text-decoration`	• Underline

Must-know properties (3)

Rule	Explanation
`float`	• Right / left
`clear`	• Clears float, right / left
`width`	• Element dimensions…
`height`	• …include padding and border

```
h1 {
  font-family:
    Consolas, monospace;
  font-size: x-large;
  font-weight: bold;
  color: blue;
}

a {
  text-decoration: none;
}

a:hover {
  text-decoration:
              underline;
}
```

EXAMPLE

Monospaced font (Consolas, if present), extra large (relative to parent font), bold and in blue.

Underlined links are disabled.

Link gets underlined when cursor is hovering.

SELECTOR PRIORITIES

- Inline style
- More specific rule
  - Some properties can be inherited
  - Recent rules are more specific
- Style included in the document
- Style of the document
- Imported style

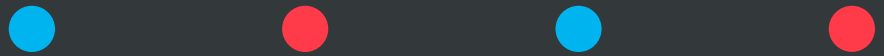
USEFUL RESOURCES ON CSS

- http://www.cssbasics.com
  - Brief presentation of CSS basics
- http://www.xhtml.com/en/css/reference/
  - CSS reference; page also includes XHTML reference
- http://www.csszengarden.com/
  - Downloadable styles for the same page
  - Shows different designs – not only different CSS
- *Eric Meyer on CSS* and *More Eric Meyer on CSS*
  - In-depth tutorials based on examples
  - Highly recommended
- Any good book about UI/UX design
  - CSS is just a tool

DEMO!

Shoutbox step 6
http://github.com/vaadin-miki/shoutbox

end branch: `step-06`

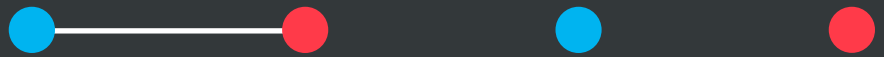


THE PLAN

- Add simple styling ☺
  - Button should be more visible
  - The top bar could use some more bling to it
  - Filtered messages should be grayed out

NAVIGATION

Decomposing the UI



REASONS

- Decomposes the UI into views
  - Better organisation of the code
  - Smaller classes
  - Reusability
- One view = one URL address
  - Bookmarking and sharing
  - Browser history
- Web-oriented
- Part of the core of the framework
  - But optional

COM.VAADIN.NAVIGATOR.NAVIGATOR

- Not a component
  - Requires `ComponentContainer` to display views in
- Registering views
  - `addView(String, View);`
    - Not very efficient, but straightforward
  - `addView(String, Class<? extends View>);`
    - Delays construction of the view until it is first called
  - `addProvider(ViewProvider);`
    - The Factory pattern
- Views are URL-based
  - `http://example.com/app/#!view-name`
- Manual navigation
  - `navigateTo(String);`

COM.VAADIN.NAVIGATOR.VIEW

- Interface that represents a single view
  - May be available under different names, though
  - May be completely different from one call to another
    - Which of course defeats the purpose, but still…
- One method
  - `enter(ViewChangeEvent);`
- A view in an application
  - Extends `Component`
    - Usually a layout
  - Implements `View`
    - May happen that `enter()` is empty

ARRANGING THE CODE

- No precise rules or requirements
  - Only suggestions and guidelines
  - Personal – other people may have other opinions
- Use common sense
  - SOLID (Principles of Object-Oriented Design, Robert C. Martin)
  - KISS
- Use naming patterns
- Try to generalise
  - Do not overgeneralise ☺
  - Think of a possible reuse
  - Anticipate change

vaadin}>

DEMO!

Shoutbox step 7
http://github.com/vaadin-miki/shoutbox

end branch: `step-07`

THE PLAN

- Enable rooms
  - Use view provider
    - Dynamic
    - Room name = view name
  - Allow message to have room name
    - Filter messages with streams
  - Style room name differently from the message

SUMMARY

What did we do today

LESSONS OF TODAY (HOPEFULLY)

- Layouts
  - How do they differ?
- Themes
  - How to use a theme?
  - How to create a theme?
- Navigation
  - What are views?
  - How to navigate between them?

COMING UP NEXT

- Extending Vaadin
- Best practices / Mobile First

THE END

SUGGESTIONS?
QUESTIONS?

miki@vaadin.com
t: @mikiolsz

vaadin}>