

Chapter 4

Web App Development Platform Via Vaadin

บรรยายโดย ผศ.ดร.ธราวิเชษฐ์ ธิติจรรุญโรจน์ และอาจารย์สัญญาชัย น้อยจันทร์

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Vaadin คืออะไร



คือ เฟรมเวิร์กแบบ Open Source ที่ช่วยให้เราพัฒนาเว็บไซต์ได้รวดเร็วด้วย Vaadin Flow สำหรับนักพัฒนาภาษาจาวาและ Vaadin Fusion สำหรับนักพัฒนาภาษา Typescript



Flow

You build your app from UI components without ever having to touch HTML or JavaScript.



Java Backend & Java UI



Fusion

You build your app in TypeScript using reactive templates with a type safe Java backend.



Java Backend & TypeScript UI



Vaadin Flow คืออะไร

WELLS
FARGO

DELL EMC

NETFLIX

motorola

ENBRIDGE

Bank of America

Volkswagen

PRIMERICA

JPMorganChase



Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Form Inputs

Checked

Unchecked

Checked disabled

Unchecked disabled

Checkbox

Latest v3.0.0

Combo box

List item

List item

List item

List item

Combo Box

Latest v6.0.1

Price

234.95

EUR

Datetime

Jan 1st

12:34

Credit card number

Custom Field

Latest v2.0.0

Date

Choose a date

March 2018

Mon Tue Wed Thu Fri Sat Sun

9 1 2 3 4 5 6 7

10 8 9 10 11 12 13 14

11 15 16 17 18 19 20 21

12 22 23 24 25 26 27 28

13 29 30 31 1 2 3 4

Date Picker

Latest v5.0.0

8/10/2020

08:00

August 2020

Sun Mon Tue Wed Thu Fri Sat

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

Date Time Picker

Latest v2.0.0

Email

team@vaadin.com

Email Field

Latest v3.0.2

List item

List item

List item

List item

List Box

Latest v2.0.0

Label

42

Number Field

Latest v3.0.2

Password

Password

secret1

Password Field

Latest v3.0.2

Checked

Unchecked

Checked disabled

Unchecked disabled

Radio Button

Latest v1.5.4

CLOSED STATE

Opened menu

List item

OPENED STATE

List item

List item

List item

List item

Select

Latest v3.0.0

Description

Write here...

Text Area

Latest v3.0.2

Form Inputs ของ Vaadin จะประกอบไปด้วย 15 Components ได้แก่ Checkbox, Combo Box, Custom Field, Date Picker, Date Time Picker, Email Field, List Box, Number Field, Password Field, Radio Button, Select, Text Area, Text Field, Time Picker และ Upload



Form Inputs - Checkbox

☒ Checked

☐ Unchecked

☒ Checked

☐ Unchecked

☒ Indeterminate

☐ Indeterminate



Form Inputs - Checkbox

```
@Route(value = "index")
public class View extends HorizontalLayout {

    public View() {
        Checkbox checkbox = new Checkbox();
        checkbox.setLabel("Option");
        checkbox.setValue(true);
        add(checkbox);
    }
}
```





Form Inputs - Checkbox

```
@Route(value = "index")
public class View extends HorizontalLayout {
    public View() {

        CheckboxGroup<String> checkboxGroup = new CheckboxGroup<>();
        checkboxGroup.setLabel("Quiz 1: Which is incorrect?");
        checkboxGroup.setItems("1+1 = 2", "1*1 = 1", "1-1 = 1");

        // Horizontal is default if you don't define the theme. In this case, it is set to Vertical
        checkboxGroup.addThemeVariants(CheckboxGroupVariant.LUMO_VERTICAL);
        add(checkboxGroup);
    }
}
```

A screenshot of a web browser window displaying the rendered form. The browser's address bar shows 'localhost'. The form content is as follows:

Quiz 1: Which is incorrect?

- ☐ 1+1 = 2
- ☐ 1*1 = 1
- ☐ 1-1 = 1



Form Inputs - Combo Box

Combo box

List item



Combo box

List item



✓ List item

List item

List item



Form Inputs – Combo Box

```
@Route(value = "index")
public class View extends HorizontalLayout {

    public View() {
        ComboBox<String> labelComboBox = new ComboBox<>();
        labelComboBox.setItems("Option one", "Option two");
        labelComboBox.setLabel("Label");

        ComboBox<String> placeholderComboBox = new ComboBox<>();
        placeholderComboBox.setItems("Option one", "Option two");
        placeholderComboBox.setPlaceholder("Placeholder");

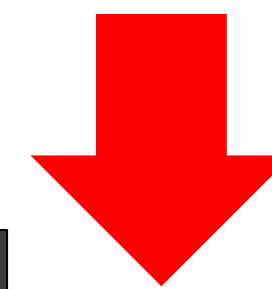
        ComboBox<String> valueComboBox = new ComboBox<>();
        valueComboBox.setItems("Value", "Option one", "Option two");
        valueComboBox.setValue("Value");
        add(labelComboBox, placeholderComboBox, valueComboBox);
    }
}
```




Form Inputs – Combo Box

localhost

Label Placeholder Value



localhost

Label Placeholder Value

Option one
Option two

localhost

Label Placeholder Value

Option one
Option two

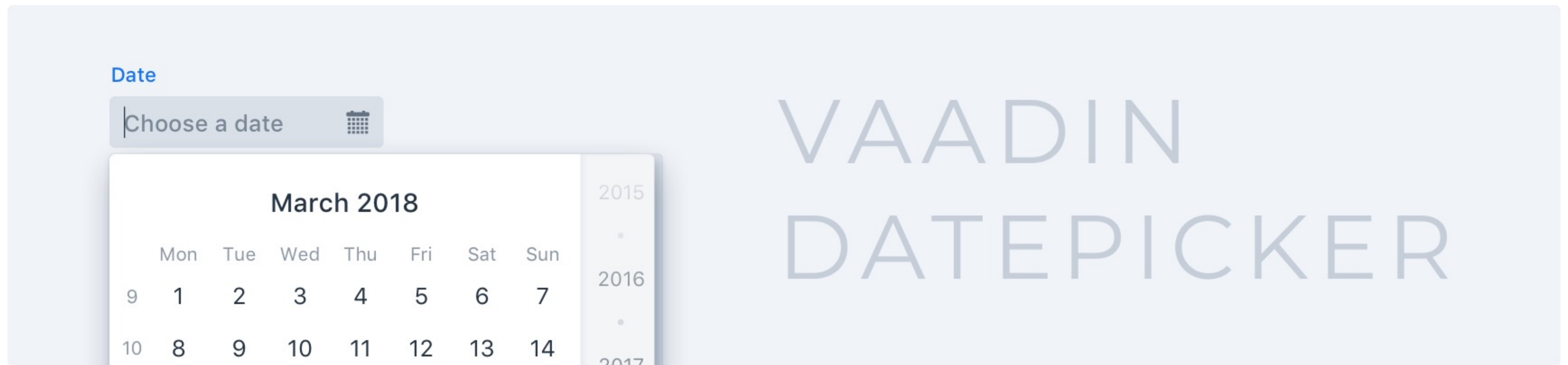
localhost

Value

✓ Value
Option one
Option two



Form Inputs - Date Picker





Form Inputs – Date Picker

```
@Route(value = "index")
public class View extends HorizontalLayout {
    public View() {

        DatePicker labelDatePicker = new DatePicker();
        labelDatePicker.setLabel("Label");

        DatePicker placeholderDatePicker = new DatePicker();
        placeholderDatePicker.setPlaceholder("Placeholder");

        DatePicker valueDatePicker = new DatePicker();
        LocalDate now = LocalDate.now();
        valueDatePicker.setValue(now);
        add(labelDatePicker, placeholderDatePicker, valueDatePicker);

    }
}
```



Form Inputs – Date Picker

Label Placeholder 05/11/2564 BE

Calendar Picker

November 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

December 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4

Today Cancel

2019 2020 2021 2022 2023 2024

นอกจากนี้ ยังมี Date Time Picker
และ Time Picker ให้เลือกใช้งาน



Form Inputs – Field

Label

Value



Label

Placeholder

Label

Placeholder

Password

.....



Disabled

Value

Read-only

Value

Invalid

Value



Error message

Text area

Value



Form Inputs – Field

```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {

        TextField txt = new TextField();
        txt.setLabel("Label");
        txt.setPlaceholder("Placeholder");

        PasswordField pass = new PasswordField();
        pass.setLabel("Password");
        pass.setPlaceholder("Enter password");

        EmailField emailTxt = new EmailField("Email");
        emailTxt.setClearButtonVisible(true);
        emailTxt.setErrorMessage("Please enter a valid email address");

        NumberField numTxt = new NumberField("Years of expertise");
```



Form Inputs – Field

```
TextArea textArea = new TextArea("Description");
textArea.setPlaceholder("Write here ...");

NumberField dollarField = new NumberField("Dollars");
dollarField.setPrefixComponent(new Span("$"));

NumberField euroField = new NumberField("Euros");
euroField.setSuffixComponent(new Span("€"));

add(txt, pass, emailTxt, textArea, numTxt, dollarField, euroField);

    }
}
```



Form Inputs – Field

[illegible]

Description

Write here ...

Years of expertise

Dollars

\$

Euros

€



Form Inputs – List Box

✓ List item

FOCUS STATE

List item

List item

HOVER STATE

List item

List item

VAADIN
LISTBOX



Form Inputs – List Box

```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {
        ListBox<String> listBox = new ListBox<>();
        listBox.setItems("Ant", "Bee", "Cat", "Dog");
        listBox.setValue("Bee");

        add(listBox);
    }
}
```

A screenshot of a web browser window displaying a list box. The browser's address bar shows 'localhost'. The list box contains four items: 'Ant', 'Bee', 'Cat', and 'Dog'. The 'Bee' item is selected, indicated by a blue checkmark to its left. The 'Cat' item is highlighted with a light blue background.



Form Inputs – Radio Button

☒ Checked

☐ Unchecked

☒ Checked disabled

☐ Unchecked disabled



Form Inputs – Radio Button

```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {
        RadioButtonGroup<String> rd1 = new RadioButtonGroup<>();
        rd1.setLabel("Question 1: Which is a correct answer?");
        rd1.setItems("int a;", "INT a;", "Int a;");
        rd1.setHelperText("Footer text ");
        //radioGroupHelperComponent.setHelperComponent(new Span("Helper text"));

        RadioButtonGroup<String> rd2 = new RadioButtonGroup<>();
        rd2.setLabel("Question 2: Which is an animal?");
        rd2.setItems("Ant", "Boat", "Car", "Doll");
        rd2.addThemeVariants(RadioGroupVariant.LUMO_VERTICAL);

        add(rd1, rd2);
    }
}
```




Form Inputs – Radio Button

localhost

Question 1: Which is a correct answer?

☐ int a; ☐ INT a; ☐ Int a;

Footer text

Question 2: Which is an animal?

☐ Ant

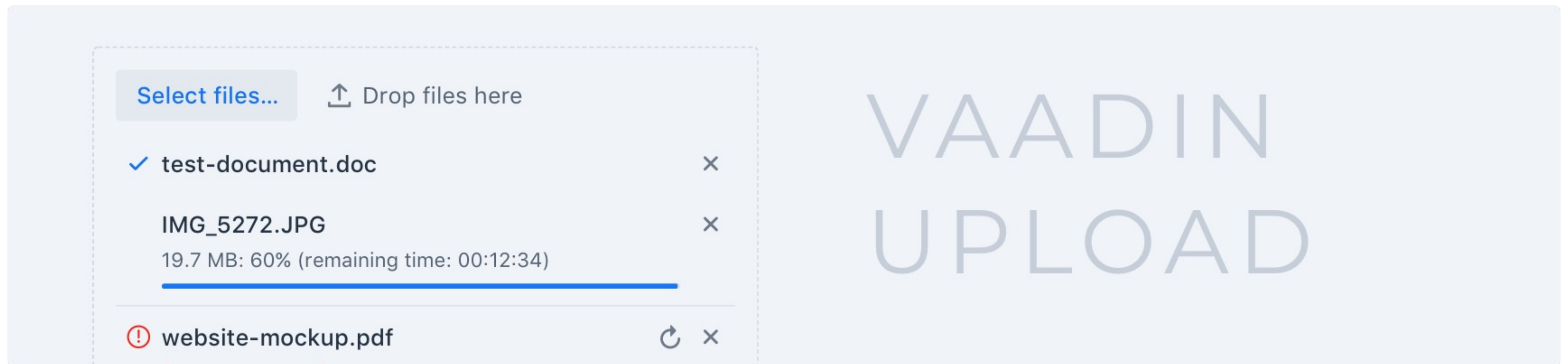
☐ Boat

☐ Car

☐ Doll



Form Inputs – Upload





Form Inputs – Upload

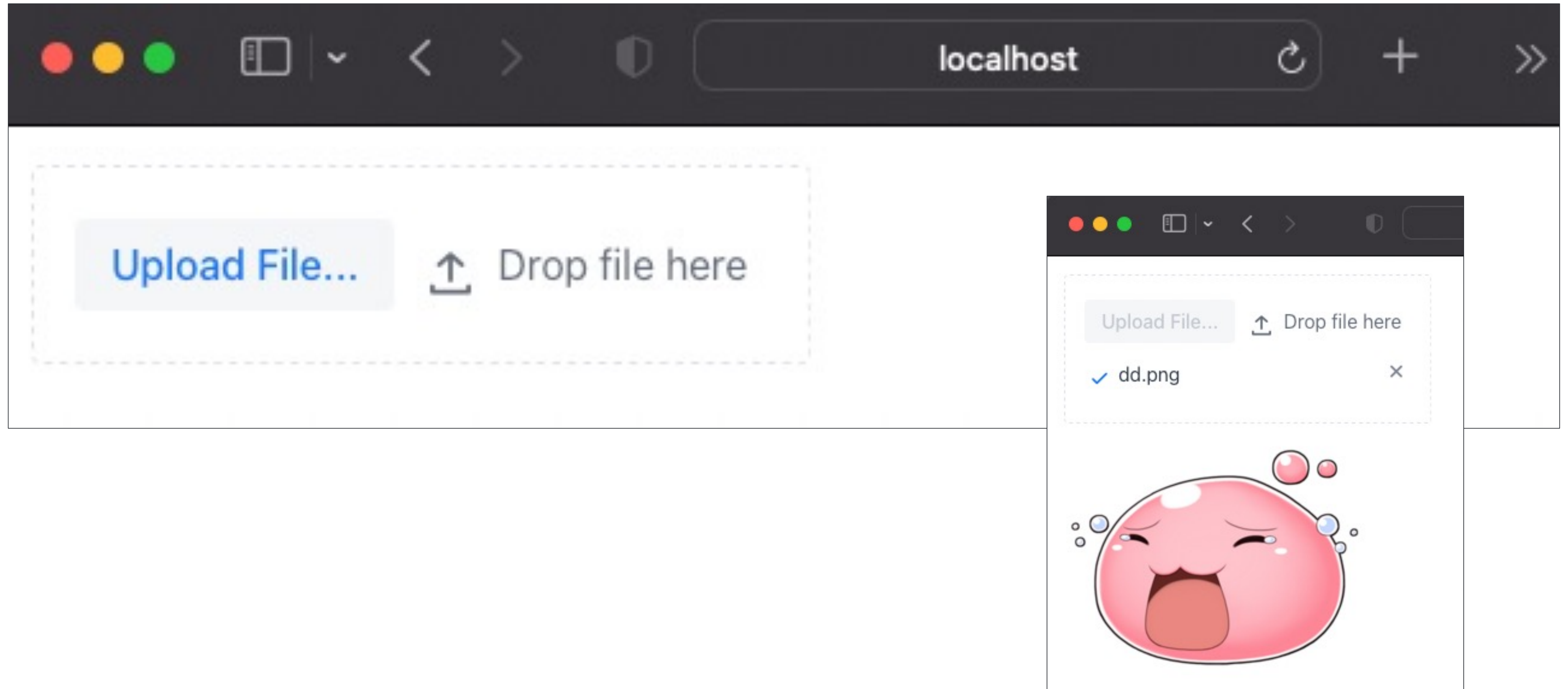
```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {
        MemoryBuffer memoryBuffer = new MemoryBuffer();
        Upload upload = new Upload(memoryBuffer);

        // upload.addFinishedListener(e -> {...insert an upload script...});

        add(upload);
    }
}
```



Form Inputs – Upload



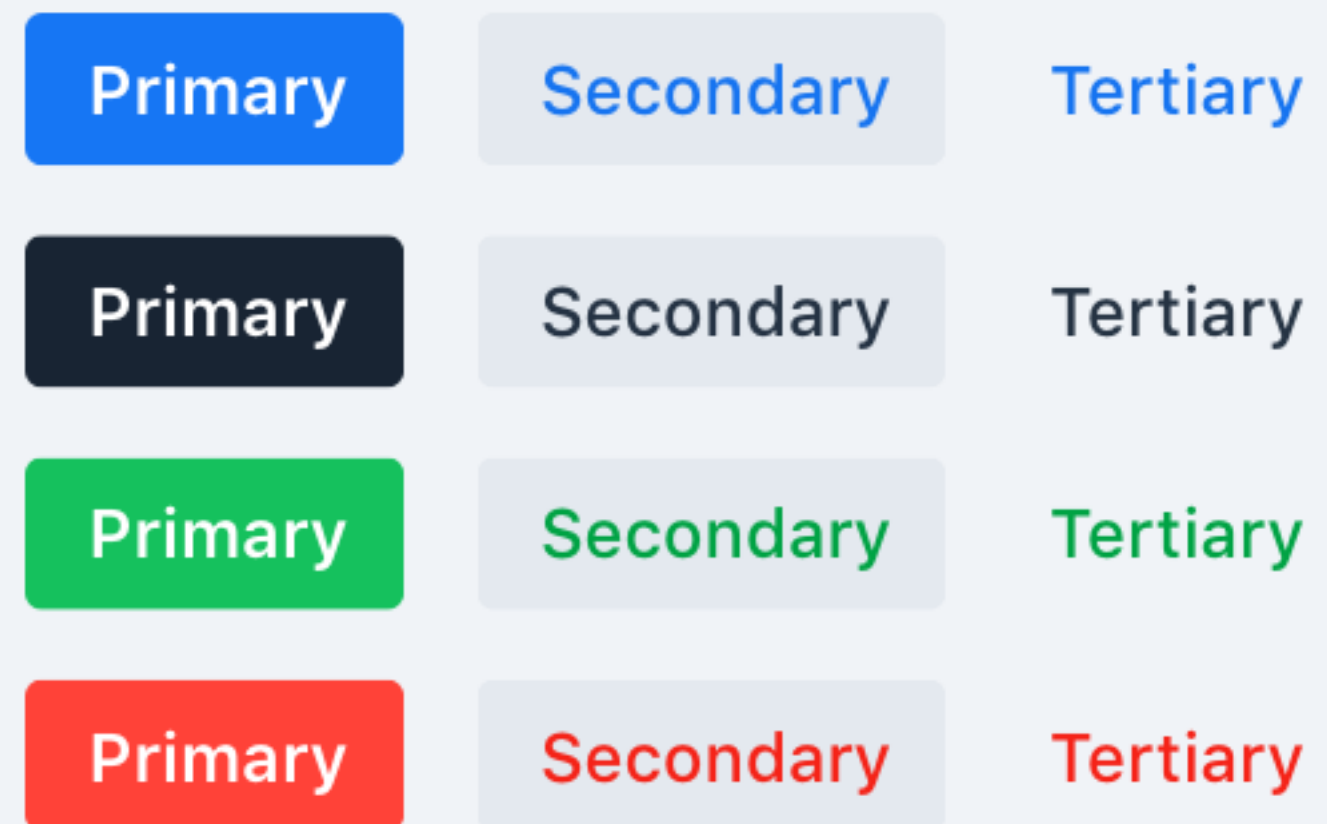


Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Visual & Interaction – Button



VAADIN
BUTTON



Visual & Interaction – Button

```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {

        Button btn1 = new Button("Vaadin button");

        Button btn2 = new Button("Left", new Icon(VaadinIcon.ARROW_LEFT));

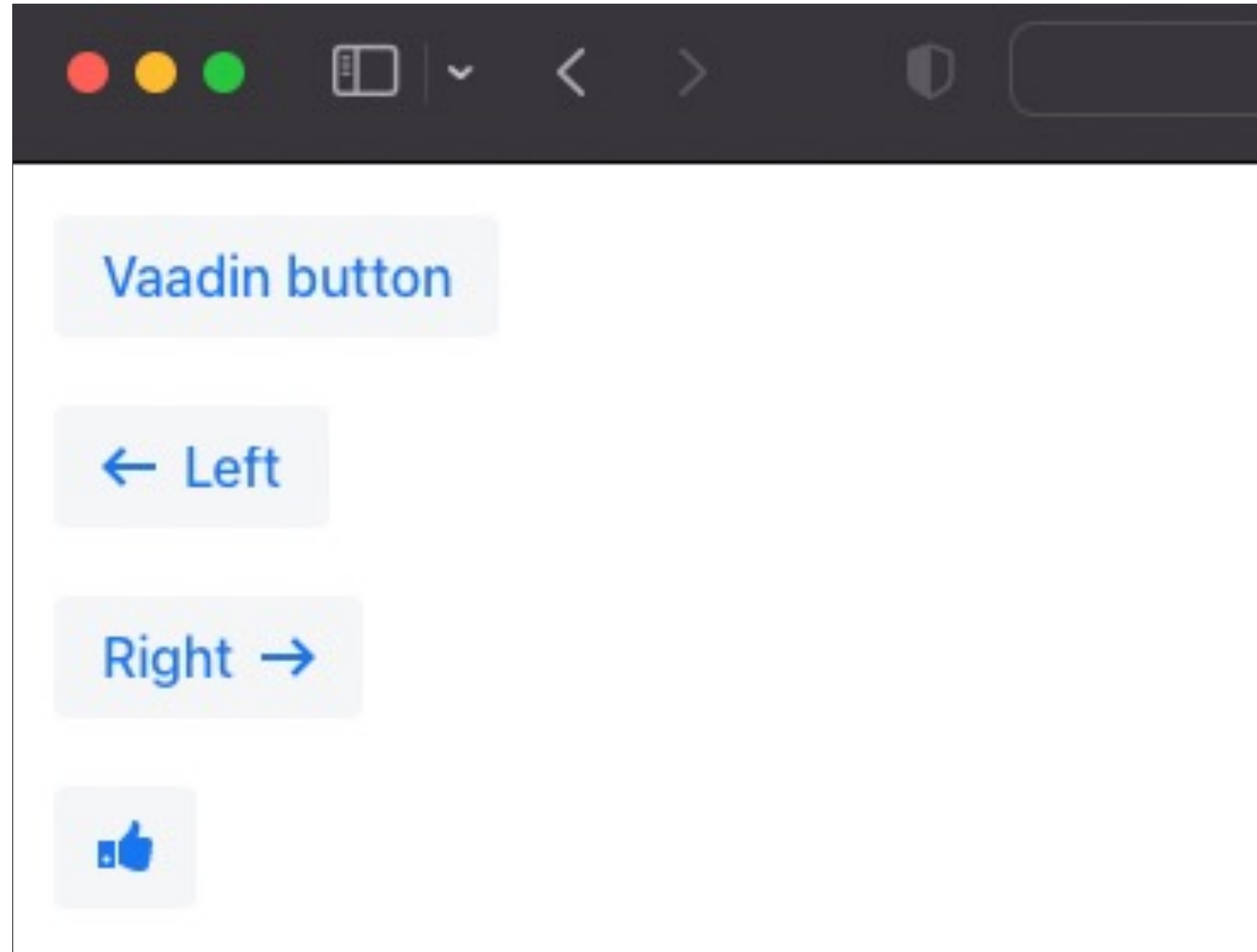
        Button btn3 = new Button("Right", new Icon(VaadinIcon.ARROW_RIGHT));
        btn3.setIconAfterText(true);

        Button btn4 = new Button(new Icon(VaadinIcon.THUMBS_UP));

        add(btn1, btn2, btn3, btn4);
    }
}
```



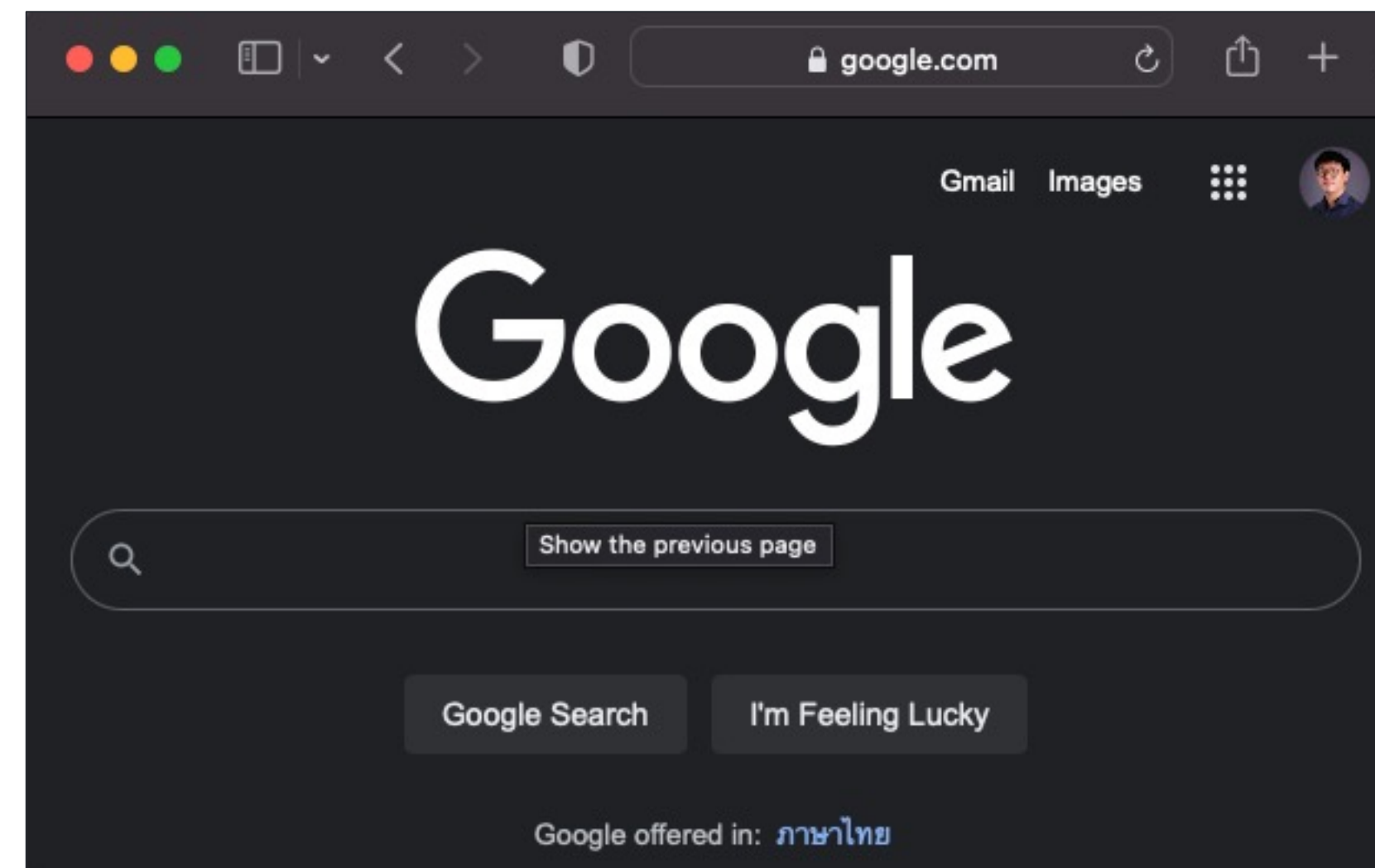
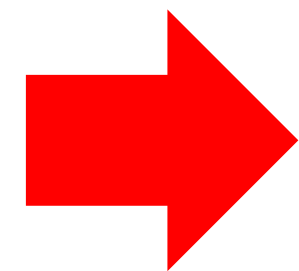
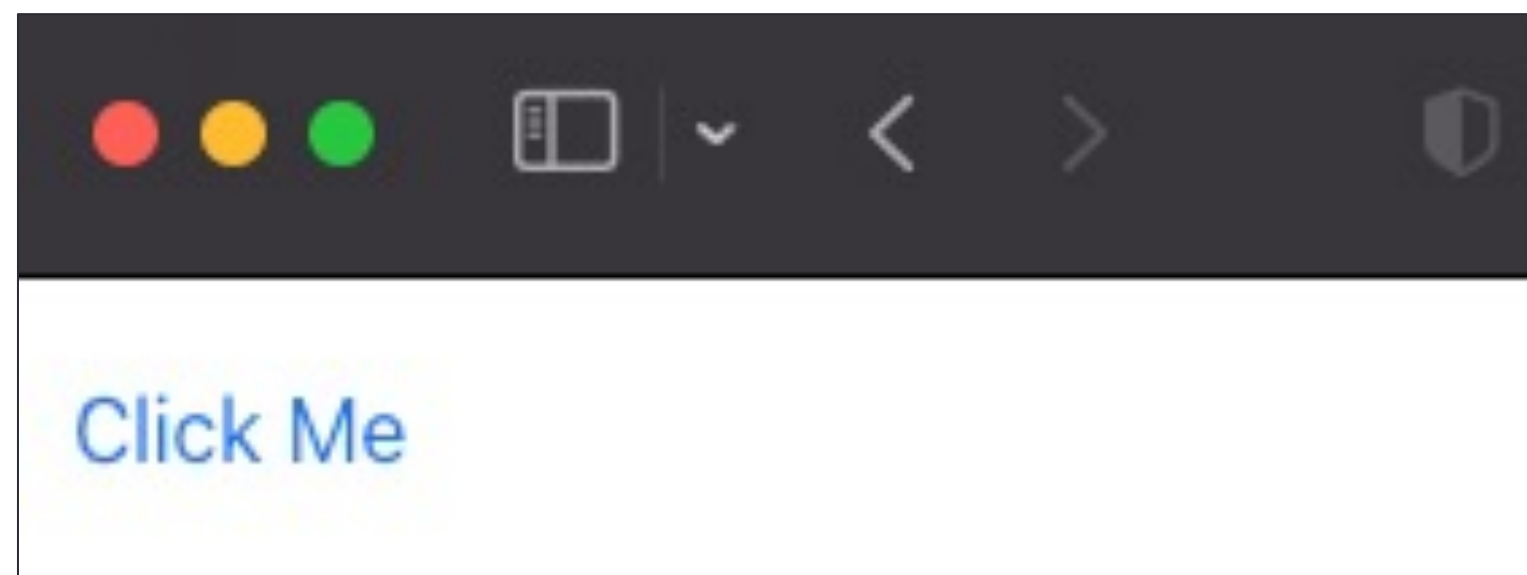
Visual & Interaction – Button





Visual & Interaction – Anchor

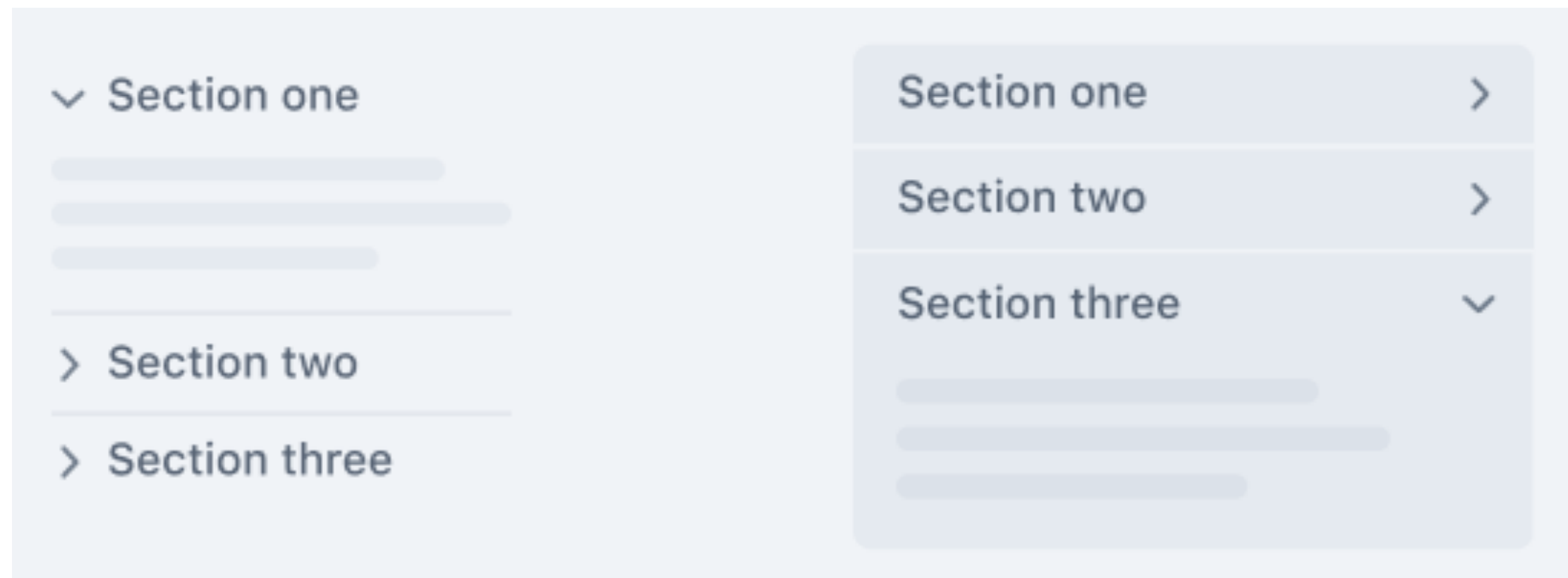
```
@Route(value = "index")
public class View extends VerticalLayout {
    public View() {
        Anchor anchor = new Anchor("https://www.google.com", "Click Me");
        add(anchor);
    }
}
```





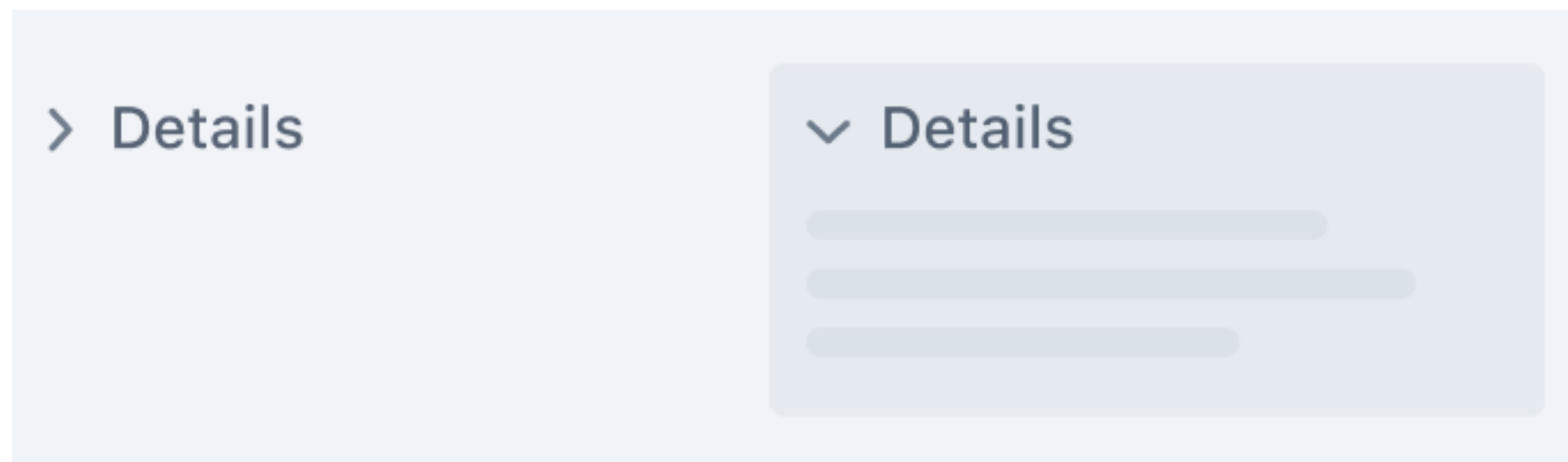
Visual & Interaction – Other

- **Accordion**



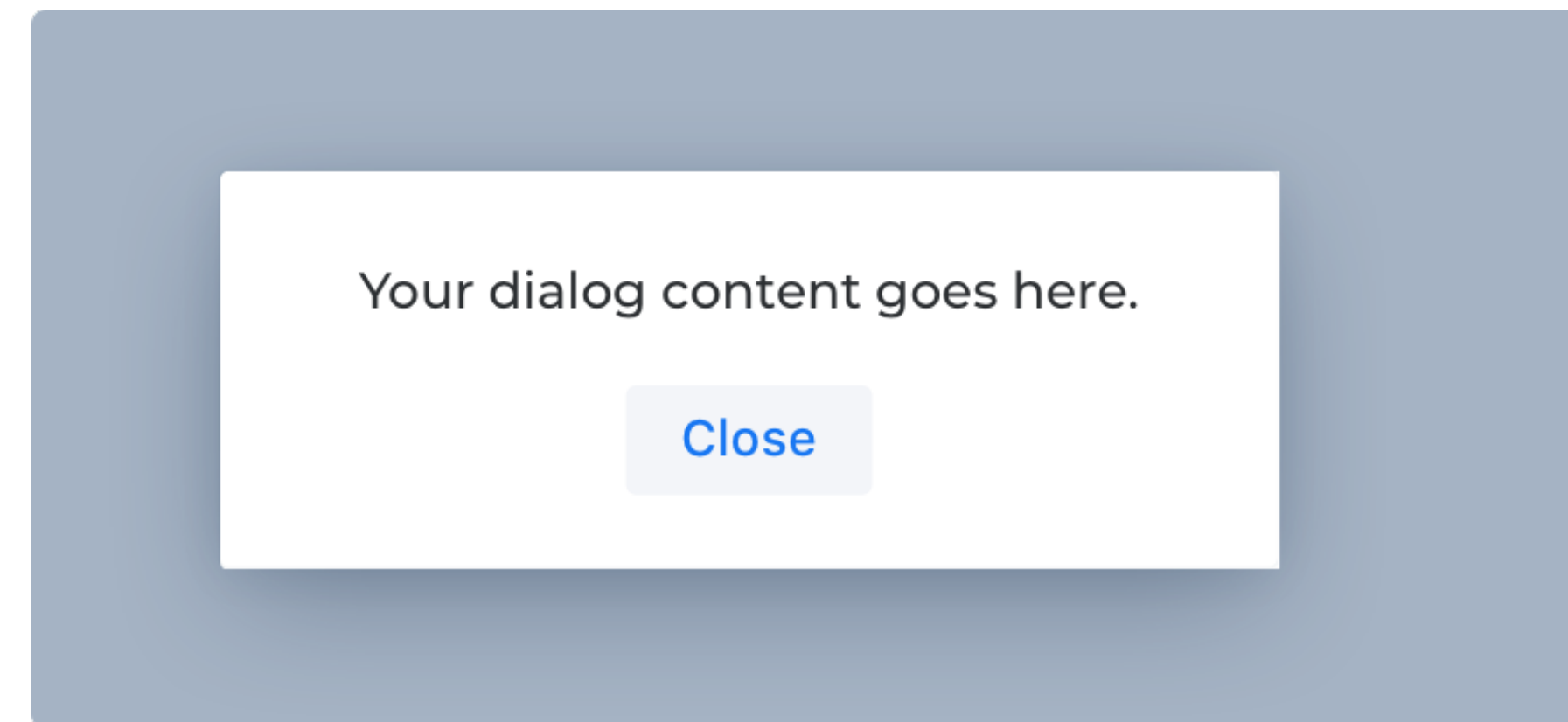
<https://vaadin.com/components/vaadin-accordion/java-examples>

- **Details**



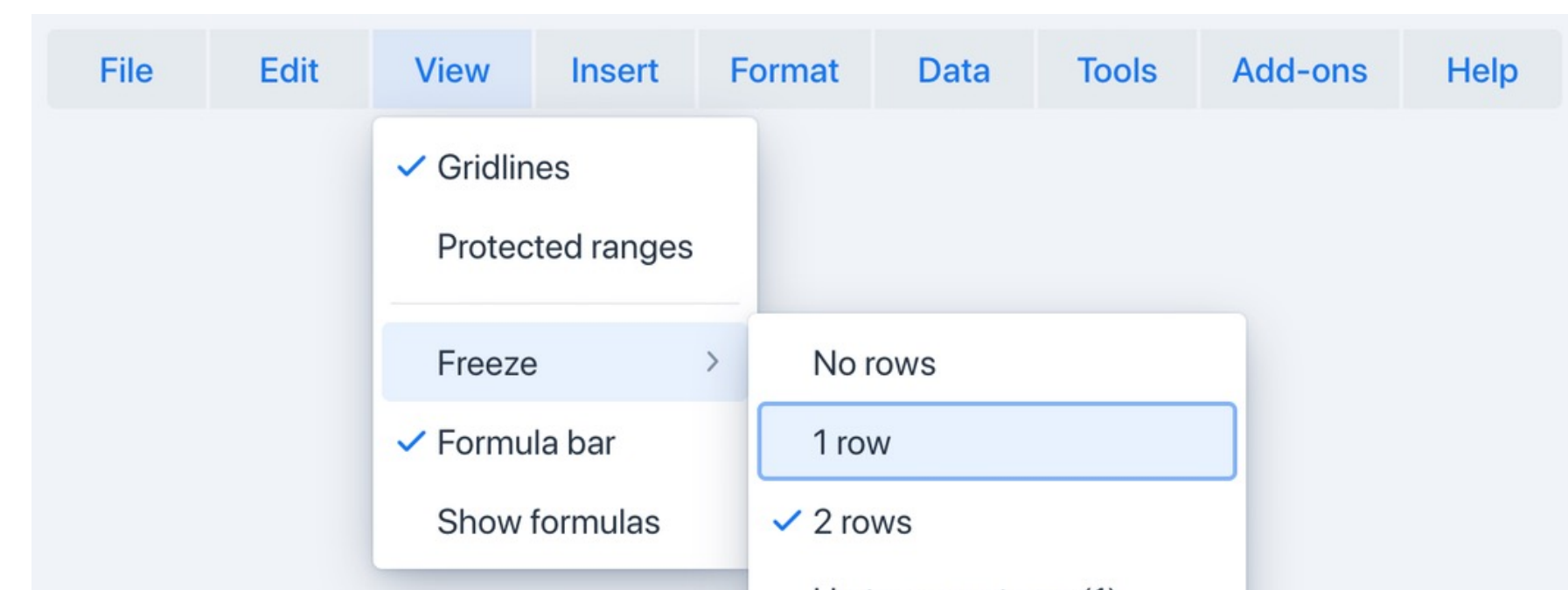
<https://vaadin.com/components/vaadin-details/java-examples>

- **Dialog**



<https://vaadin.com/components/vaadin-dialog>

- **Menu Bar**




<https://vaadin.com/components/vaadin-menu-bar/java-examples>



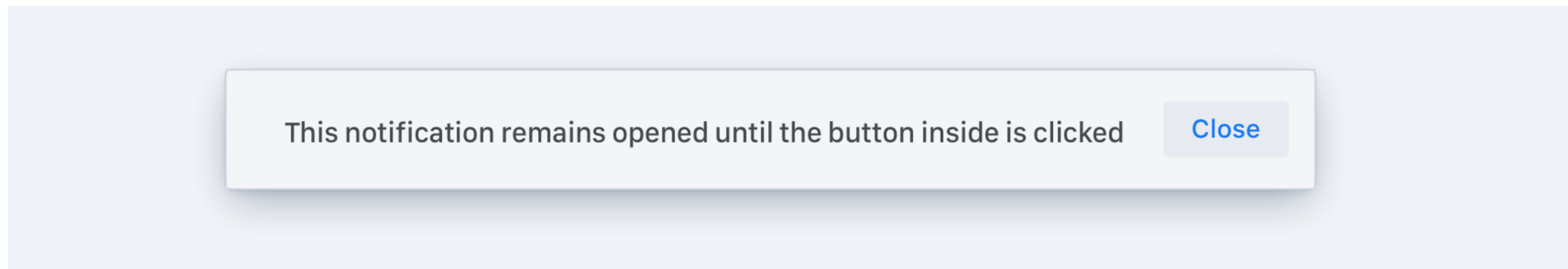
Visual & Interaction – Other

- **Grid**

	First Name ▾	Last Name ▾	Email ▾
<input type="checkbox"/>	Henry	Carter	henry.carter@example.com
<input checked="" type="checkbox"/>	Liam	Perez	liam.perez@example.com
<input type="checkbox"/>	Justin	Garcia	justin.garcia@example.com
<input type="checkbox"/>	Jordan	Howard	jordan.howard@example.com

<https://vaadin.com/components/vaadin-grid/>

- **Notification**



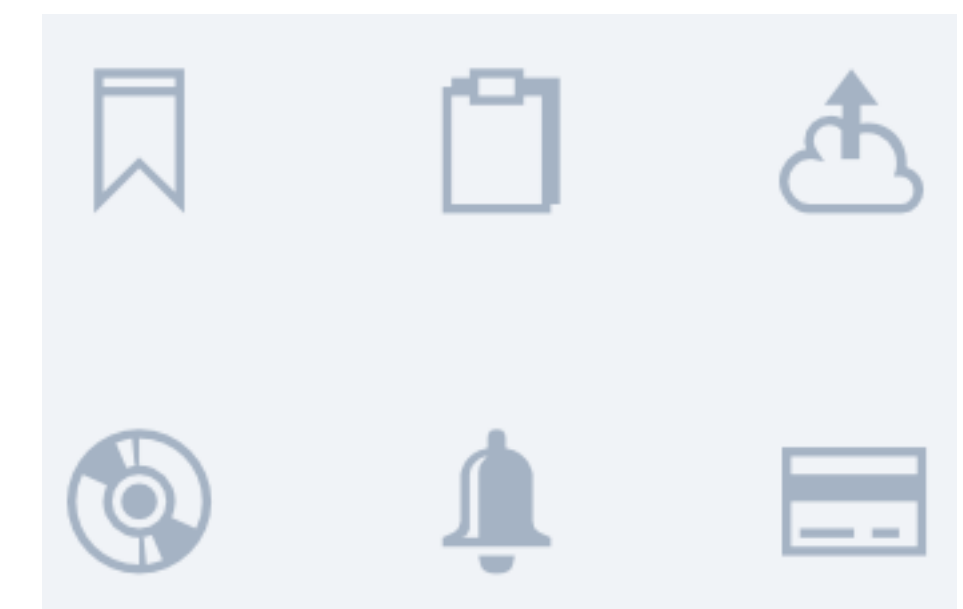
<https://vaadin.com/components/vaadin-notification>

- **Images**



<https://vaadin.com/components/vaadin-image>

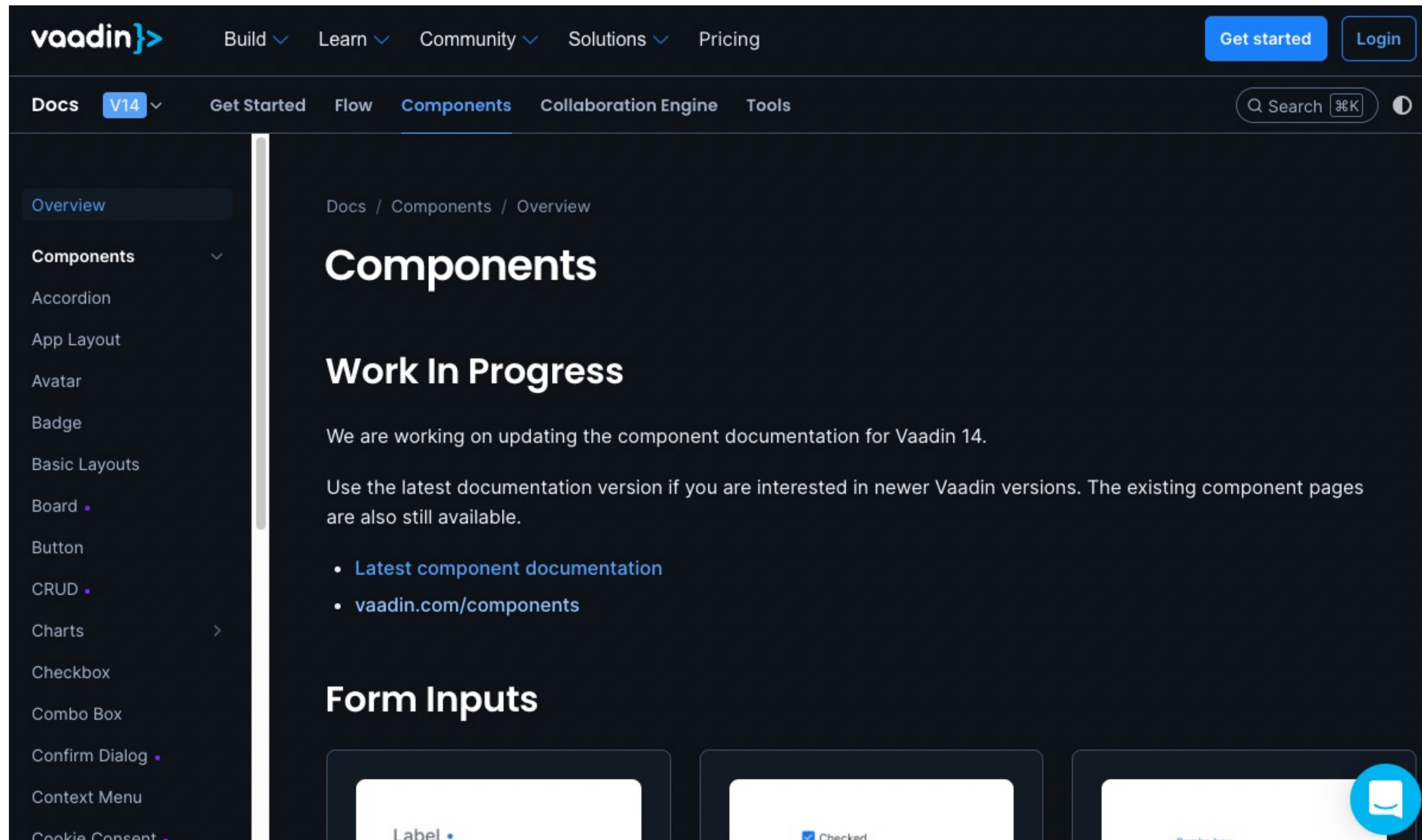
- **Icons**



<https://vaadin.com/components/vaadin-icons>



More Detail



<https://vaadin.com/docs/v14/ds/components/button>



Outline

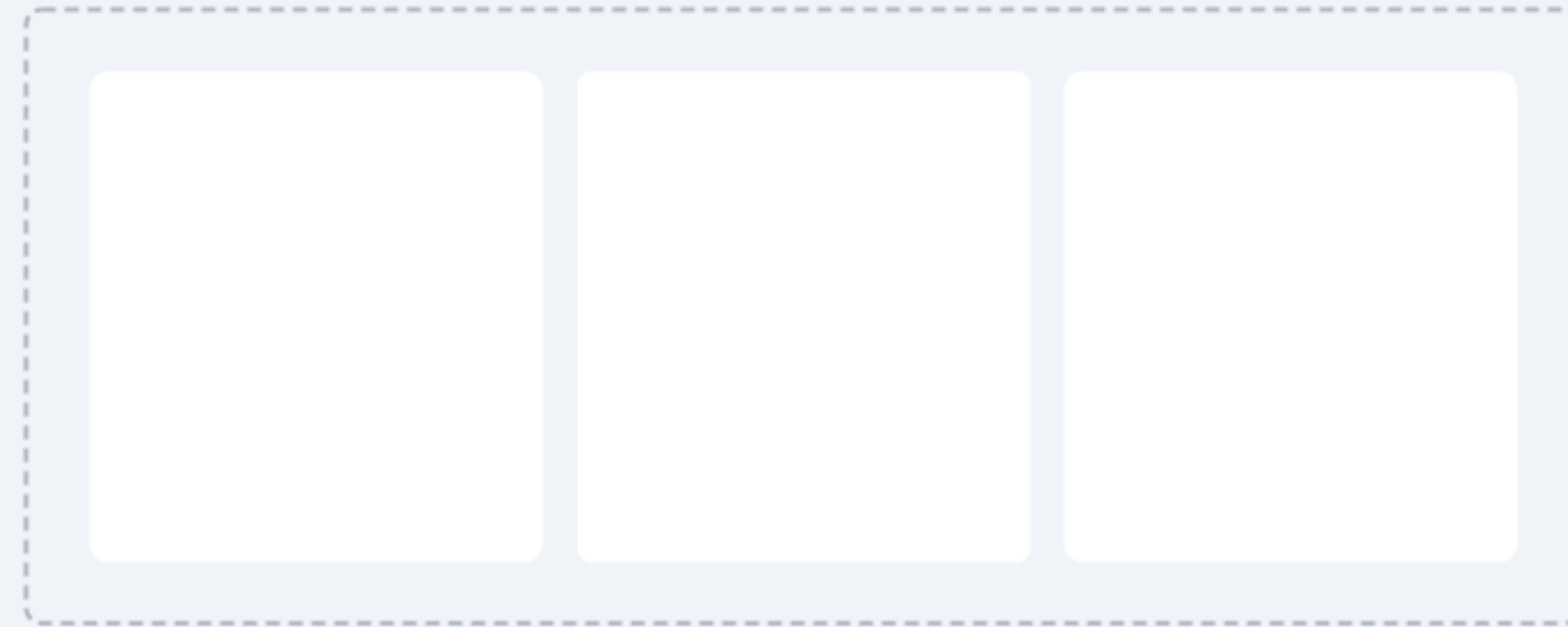
- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Ordered Layout



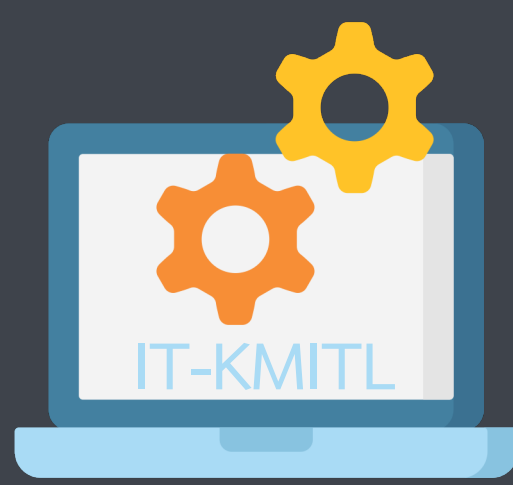
Vertical layout with item gap



Horizontal layout with container padding and item gap

เป็นการจัด Layout ที่จะจัดเรียง Component ตามแนวตั้ง (Vertical) หรือแนวนอน (Horizontal) ซึ่งประกอบไปด้วย (1) Horizontal Layout, (2) Vertical Layout และ (3) Flex Layout

อ้างอิง <https://vaadin.com/components/vaadin-ordered-layout/java-examples>



ตัวอย่างที่ 1 ของ Ordered Layout

```
@Route(value = "index4")

public class View04 extends HorizontalLayout {

    public View04() {

        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");

        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");

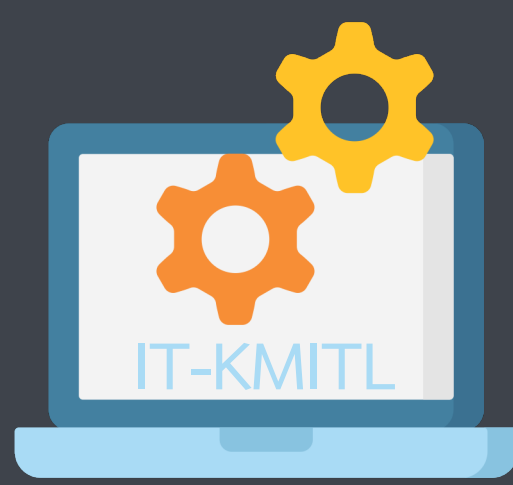
        Button btn = new Button("Send");

        this.add(firstNameField, lastNameField, btn);
    }
}
```




ตัวอย่างที่ 1 ของ Ordered Layout

A screenshot of a web browser window displaying a form. The browser's address bar shows "localhost". The form consists of two input fields, "First name" and "Last name", followed by a "Send" button. The "First name" and "Last name" fields are light gray, while the "Send" button is light blue with blue text.



ตัวอย่างที่ 2 ของ Ordered Layout

```
@Route(value = "index4")

public class View04 extends VerticalLayout {

    public View04() {

        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");

        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");

        Button btn = new Button("Send");

        this.add(firstNameField, lastNameField, btn);
    }
}
```



ตัวอย่างที่ 1 ของ Ordered Layout

A screenshot of a web browser window displaying a form. The browser's address bar shows "localhost". The form consists of two text input fields stacked vertically, labeled "First name" and "Last name". Below these fields is a button labeled "Send".



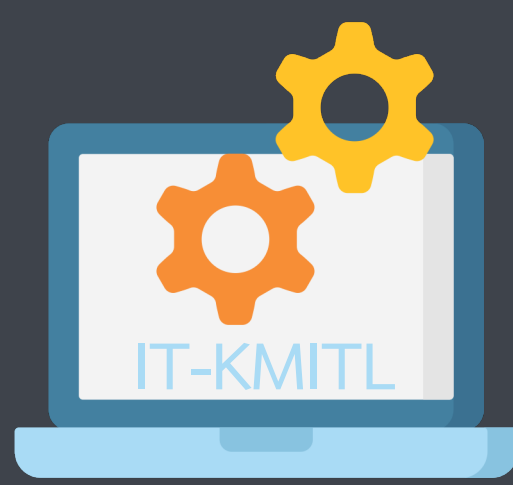
Form Layout

New order

Due •	Time •	Customer •
<input type="text" value="Choose date"/>	<input type="text" value="Time"/>	<input type="text" value="Create or select a customer"/>
Product •	Additional details •	
<input type="text" value="Choose product"/>	<input type="text"/>	
<input type="button" value="Cancel"/>	<input type="button" value="Save"/>	

FORM
LAYOUTING
MADE EASY

เป็นการจัด Layout ที่รองรับการจัดเรียง Component แบบ Responsive สำหรับ Form element



ตัวอย่างที่ 1 ของ Form Layout

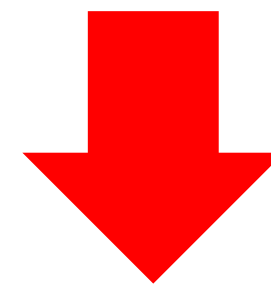
```
@Route(value = "index4")
public class View04 extends FormLayout {
    public View04() {
        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");
        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");
        Button btn = new Button("Send");

        this.add(firstNameField, lastNameField, btn);
        this.setResponsiveSteps(
            // Use one column by default
            new ResponsiveStep("1px", 1),
            // Use two columns, if the layout's width exceeds 450px
            new ResponsiveStep("600px", 2),
            // Use three columns, if the layout's width exceeds 700px
            new ResponsiveStep("700px", 3));
    }
}
```

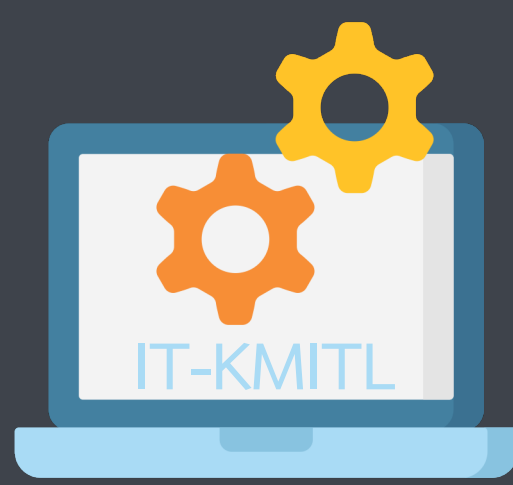



ตัวอย่างที่ 1 ของ Form Layout

A web browser window with a dark header bar showing 'localhost' and navigation icons. The form is displayed horizontally with three light gray input fields. The first field is labeled 'First name', the second is labeled 'Last name', and the third contains the text 'Send' in blue.



A web browser window with a dark header bar showing 'localhost' and navigation icons. The form is displayed vertically with three light gray input fields stacked on top of each other. The top field is labeled 'First name', the middle field is labeled 'Last name', and the bottom field contains the text 'Send' in blue.



ตัวอย่างที่ 2 ของ Form Layout

```
@Route(value = "index")
public class View extends FormLayout {
    public View() {
        this.setResponsiveSteps(new ResponsiveStep("25em", 1),
            new ResponsiveStep("32em", 2), new ResponsiveStep("40em", 3));

        TextField firstName = new TextField();
        TextField lastName = new TextField();
        TextField email = new TextField();
        TextField nickname = new TextField();
        TextField website = new TextField();
        TextField description = new TextField();
        description.setPlaceholder("Enter a short description about yourself");

        firstName.setPlaceholder("First Name");
        lastName.setPlaceholder("Last Name");
        email.setPlaceholder("Email");
        nickname.setPlaceholder("Username");
        website.setPlaceholder("Link to personal website");

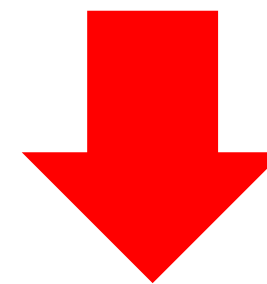
        this.add(firstName, lastName, nickname, email, website);
        // You can set the desired column span for the components individually.
        this.setColspan(website, 2);
        // Or just set it as you add them.
        this.add(description, 3);
    }
}
```



ตัวอย่างที่ 2 ของ Form Layout

localhost

First Name	Last Name	Username
Email	Link to personal website	
Enter a short description about yourself		



localhost

First Name	Last Name
Username	Email
Link to personal website	
Enter a short description about yourself	

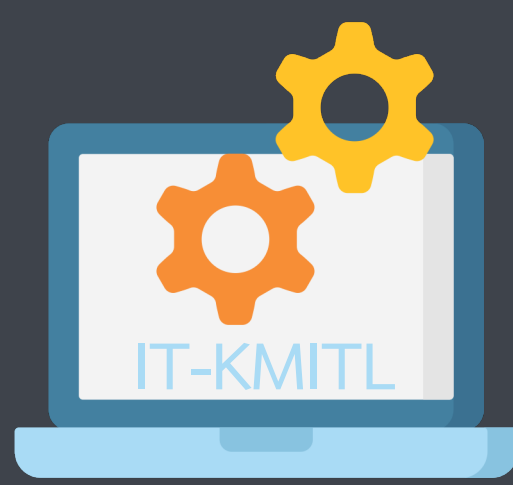


Split Layout

SPLIT

LAYOUT

เป็นการจัด Layout ที่รองรับการแบ่งหน้าเว็บเพจออกเป็นส่วน ๆ แบบปรับแต่งขนาดได้ทันที



ตัวอย่างที่ 1 ของ Split Layout

```
@Route(value = "index4")
public class View04 extends HorizontalLayout {
    public View04() {
        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");
        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");
        Button btn = new Button("Send");

        SplitLayout layout = new SplitLayout(firstNameField, lastNameField);

        this.add(layout, btn);
    }
}
```

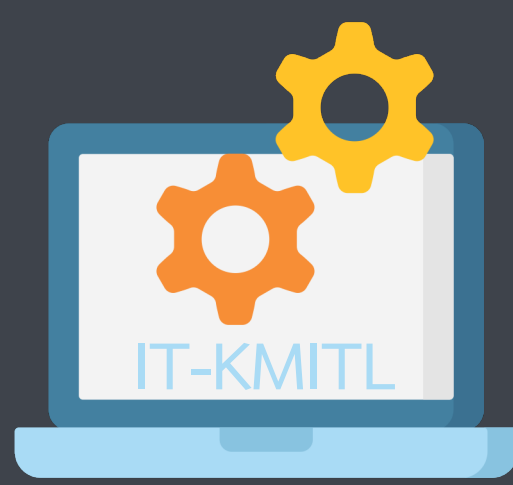


ตัวอย่างที่ 1 ของ Split Layout

A web browser window with a dark header bar containing window controls, a tab labeled 'localhost', and navigation icons. The main content area displays a form with two text input fields, 'First name' and 'Last name', positioned side-by-side. A 'Send' button is located to the right of the 'Last name' field.

A web browser window with a dark header bar containing window controls, a tab labeled 'localhost', and navigation icons. The main content area displays a form with two text input fields, 'First name' and 'Last name', positioned side-by-side. A 'Send' button is located to the right of the 'Last name' field.

A web browser window with a dark header bar containing window controls, a tab labeled 'localhost', and navigation icons. The main content area displays a form with two text input fields, 'First name' and 'Last name', positioned side-by-side. A 'Send' button is located to the right of the 'Last name' field.



ตัวอย่างที่ 2 ของ Split Layout

```
@Route(value = "index4")
public class View04 extends HorizontalLayout {
    public View04() {
        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");
        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");
        Button btn = new Button("Send");

        SplitLayout layout = new SplitLayout(firstNameField, lastNameField);
        layout.setOrientation(SplitLayout.Orientation.VERTICAL);

        this.add(layout, btn);
    }
}
```



ตัวอย่างที่ 2 ของ Split Layout

Show the next page < > localhost ↻

First name Send

Last name

localhost ↻

First name Send

Last name

localhost ↻

Last name Send



ตัวอย่างการผสม Layout

```
@Route(value = "index4")
public class View04 extends FormLayout {
    public View04() {

        TextField firstNameField = new TextField();
        firstNameField.setPlaceholder("First name");
        TextField lastNameField = new TextField();
        lastNameField.setPlaceholder("Last name");
        Button btn = new Button("Send");

        HorizontalLayout hl = new HorizontalLayout();
        VerticalLayout vl = new VerticalLayout();

        vl.add(firstNameField, lastNameField);
        hl.add(vl, btn);
        this.add(hl);
    }
}
```

The screenshot shows a web browser window with a dark header bar containing standard window controls (red, yellow, green buttons) and navigation icons. The main content area is white and contains a form. The form consists of two light blue text input fields stacked vertically. The top field has the placeholder text 'First name' and the bottom field has the placeholder text 'Last name'. To the right of these fields is a light blue button with the text 'Send' in blue.



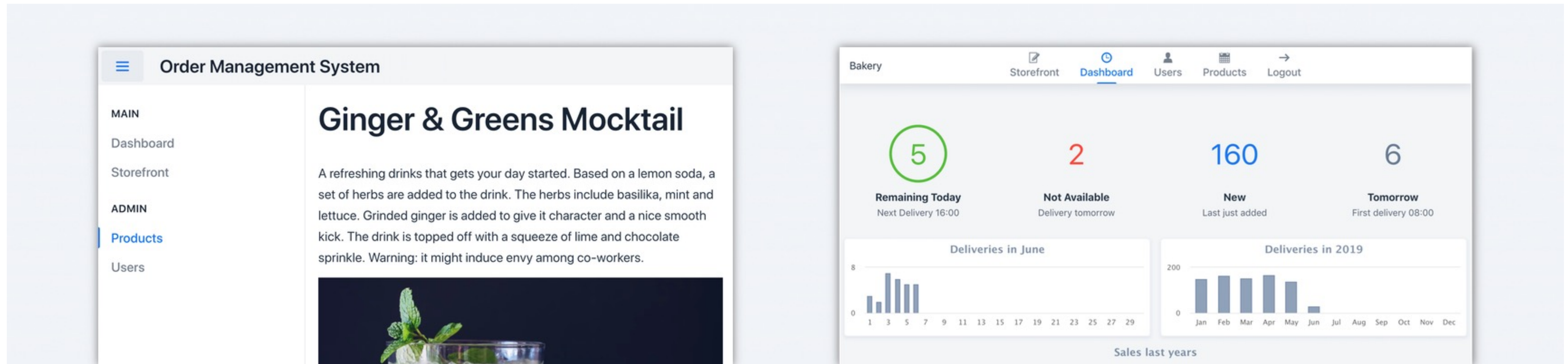
Login Layout

The image shows a mockup of the Vaadin Login Layout component. It features a white background with a light gray border. At the top left, the text "Log in" is displayed in a bold, dark gray font. Below this, there are two input fields: "Username" and "Password". The "Password" field includes a toggle icon (an eye) on its right side. A prominent blue button with the text "Log in" is positioned below the input fields. At the bottom of the layout, there is a link labeled "Forgot password" in a smaller, blue font.

เป็นการจัด Layout และ Component พื้นฐาน
สำหรับหน้าต่าง Login



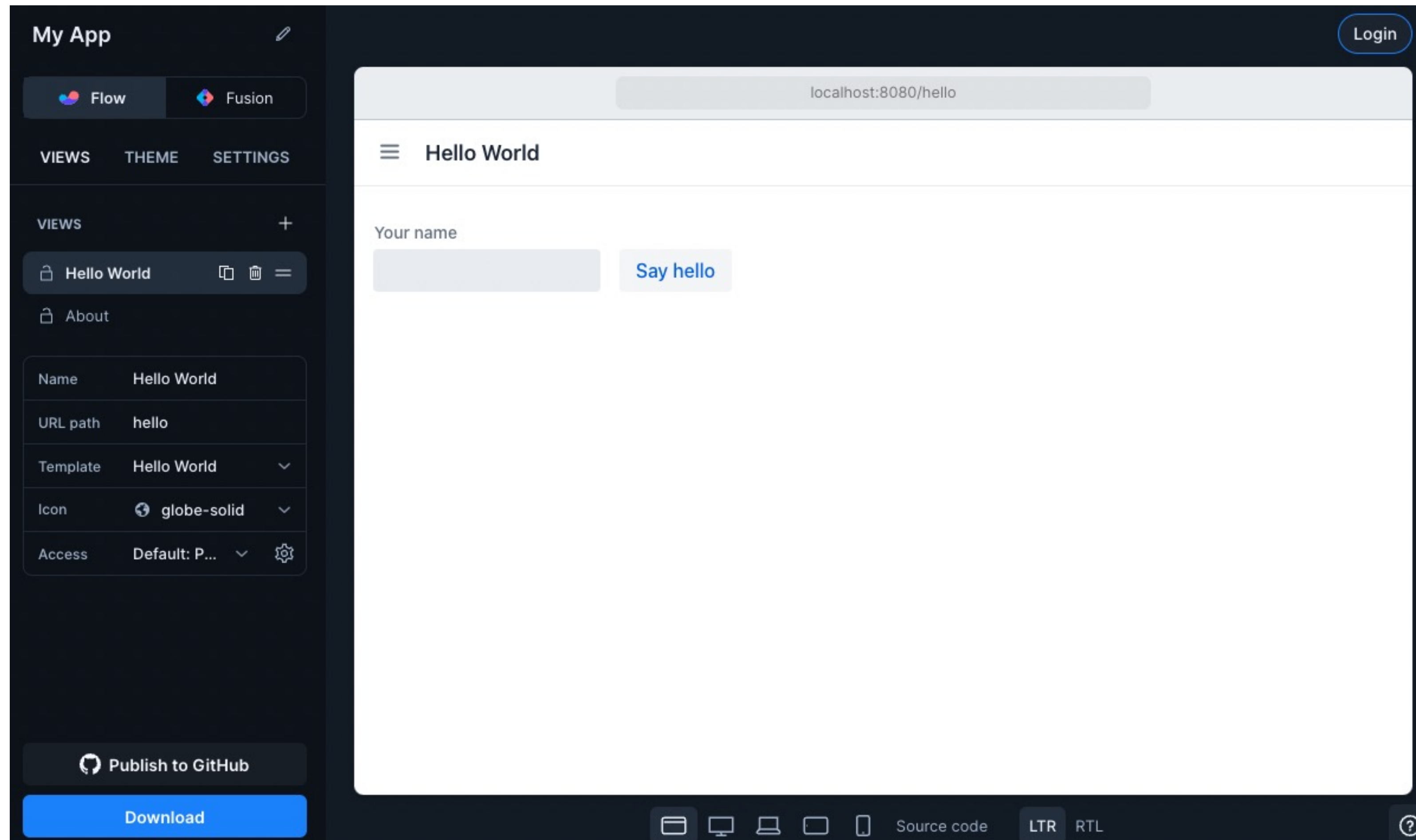
App Layout



เป็นการจัด Layout สำหรับการสร้างเว็บไซต์แบบพื้นฐานและรวดเร็ว ซึ่งรองรับการใส่ Logo, Menu, และรายละเอียดของเนื้อหา



Vaadin Starter



Vaadin Starter เป็นตัวช่วย
สำหรับการออกแบบเว็บไซต์ด้วย
Web Editor จากนั้น ค่อยโหลด
โค้ดไปเปิดด้วย IntelliJ เพื่อ
ปรับแต่งและแก้ไข

<https://start.vaadin.com/app/>

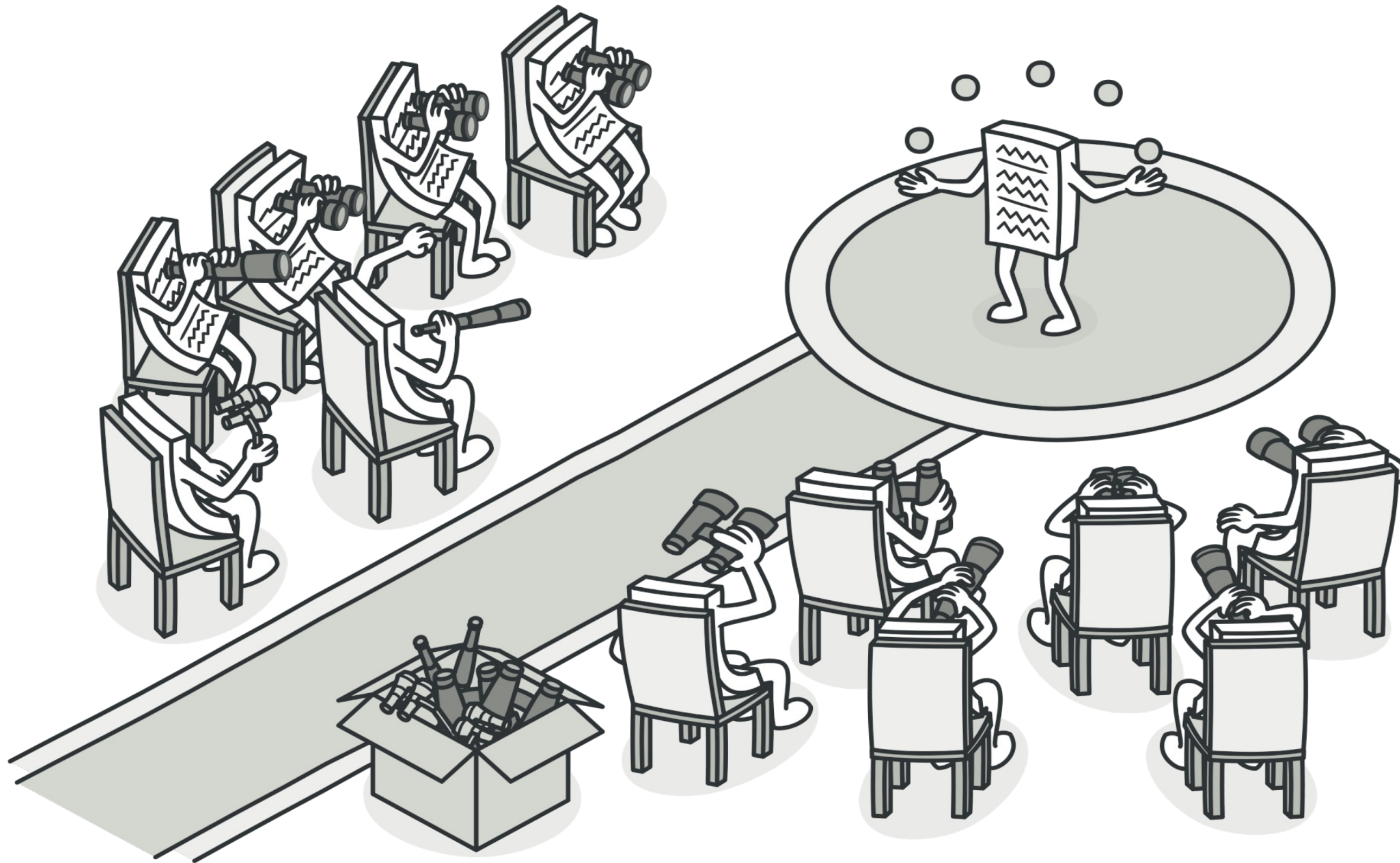


Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



Events & Listeners



Vaadin โปรแกรมแบบ event-driven programming เพื่อจัดการกับการโต้ตอบกับผู้ใช้ เมื่อผู้ใช้ทำอะไรบางอย่างในส่วนติดต่อผู้ใช้งาน เช่น คลิกปุ่มหรือเลือกรายการ โปรแกรมจำเป็นต้องทราบว่าเกิดเหตุการณ์เหล่านี้ ใน Vaadin ด้วย Flow จะทำงานแบบ Observer design pattern เพื่อเชื่อมโยงหรือสื่อสารระหว่าง View รองรับการพัฒนาไป Controller

<https://refactoring.guru/design-patterns/observer>



Events & Listeners

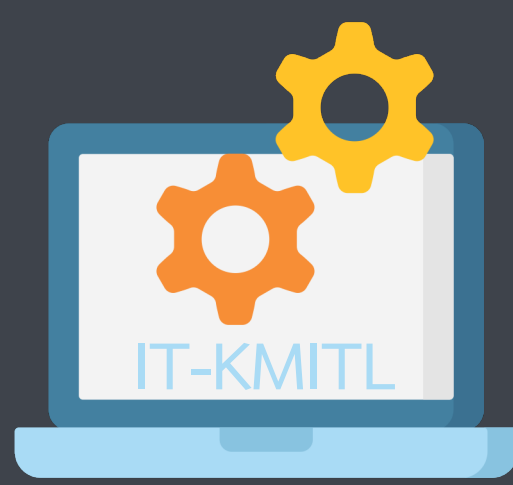


การจัดการเหตุการณ์ด้วย Vaadin มีวัตถุที่เกี่ยวข้อง ได้แก่ ผู้สร้างวัตถุเหตุการณ์, วัตถุเหตุการณ์ และ ผู้รับฟังเหตุการณ์ เมื่อเกิดเหตุการณ์ขึ้นมาวัตถุเหตุการณ์จะส่งการแจ้งเตือนเกี่ยวกับเหตุการณ์ไปยังผู้รับฟัง ในการรับเหตุการณ์ประเภทหนึ่ง ต้องระบุ Event Handler กับ Event Source ด้วยเมธอด `add***Listener()`

```
Button button = new Button("Push it!");

button.addClickListener(event ->
    // do something
);
```

ในจาวาเวอร์ชัน 8 รองรับการใช้งาน functional interfaces ร่วมกับ lambda expression

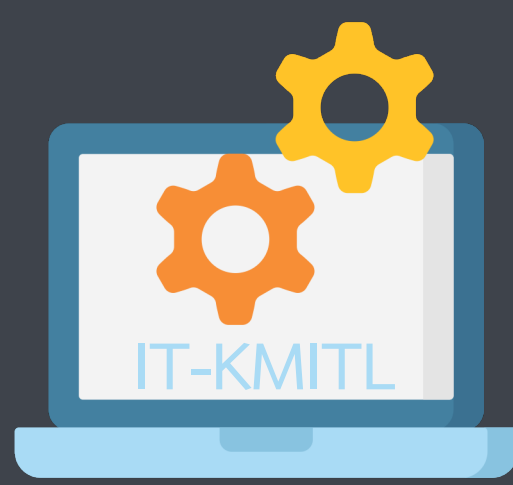


ตัวอย่างที่ 1 Events & Listeners

```
@Route(value = "index")
public class View extends VerticalLayout {
    private TextField n1, n2;
    private TextArea textArea;
    private Anchor link;
    private Button btnPlus;
    private RadioButtonGroup<String> rd;
    private ListBox<String> listBox;

    public View() {
        n1 = new TextField("Number 1");
        n2 = new TextField("number 2");
        textArea = new TextArea("Answer");
        btnPlus = new Button("+");

        textArea.getStyle().set("maxHeight", "150px");
        textArea.setPlaceholder("Write here ...");
```



ตัวอย่างที่ 1 Events & Listeners

```
rd = new RadioButtonGroup<String>();
rd.setLabel("Question 2: Which is an animal?");
rd.setItems("Ant", "Boat", "Car", "Doll");
rd.addThemeVariants(RadioGroupVariant.LUMO_VERTICAL);

listBox = new ListBox<String>();
listBox.setItems("Ant", "Bee", "Cat", "Dog");

add(n1, n2, textArea, btnPlus, rd, listBox);
// For RadioButtonGroup
rd.addValueChangeListener(event ->{
    new Notification(rd.getValue() + " (AAA)", 1000).open();
});
// For ListBox
listBox.addValueChangeListener(event ->{
    new Notification(listBox.getValue() + " (BBB)", 1000).open();
});
```

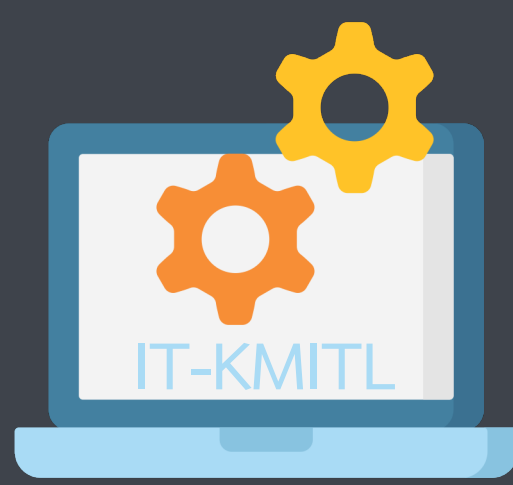


ตัวอย่างที่ 1 Events & Listeners

```
// When users press enter on the keyboard
n1.addValueChangeListener(event ->{
    new Notification(n1.getValue() + " (CCC)", 5000).open();
});

// When users press the keyboard
n1.addKeyPressListener(event ->{
    new Notification(n1.getValue() + " (DDD)", 5000).open();
});

// When users click button
btnPlus.addClickListener(event ->{
    double num1 = Double.parseDouble(n1.getValue());
    double num2 = Double.parseDouble(n2.getValue());
    textArea.setValue(num1+num2+"");
});
}
}
```

ตัวอย่างที่ 1 Events & Listeners

localhost

Number 1

number 2

Answer

+

Question 2: Which is an animal?

☐ Ant

☒ Boat

☐ Car

☐ Doll

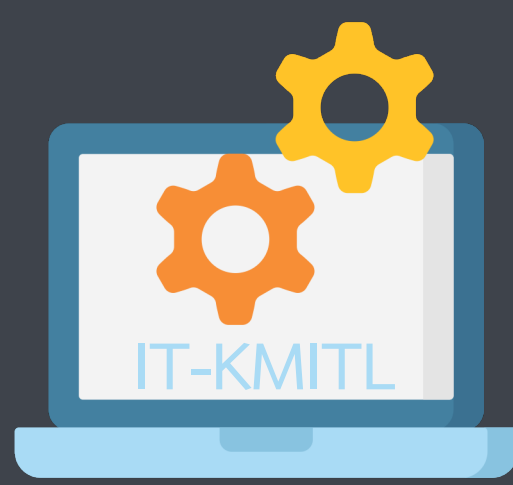
Ant

Bee

Cat

Ant (AAA)

<https://youtu.be/l79BlEeLLxs>



ตัวอย่างที่ 2 Events & Listeners

```
@Route(value = "index")
public class View extends VerticalLayout {
    private StreamResource resource;
    private MemoryBuffer memoryBuffer;
    private Upload upload;
    private InputStream inputStream;           // byte Stream
    private Image image;

    public View() {

        // Code อยู่หน้าถัดไป

    }
}
```

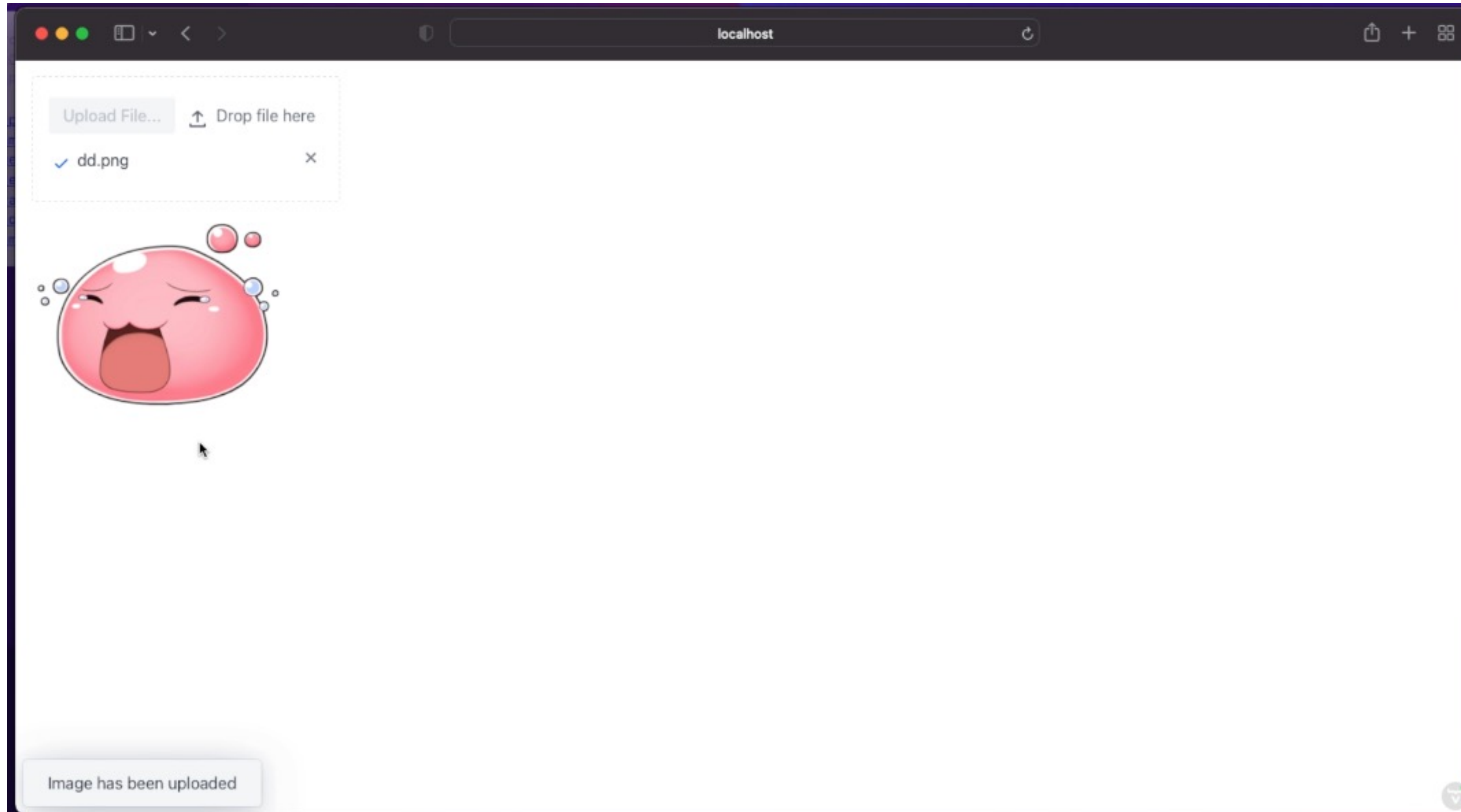


ตัวอย่างที่ 2 Events & Listeners

```
memoryBuffer = new MemoryBuffer();  
upload = new Upload(memoryBuffer); // ข้อมูลที่อัปโหลดมาจะเก็บไว้ใน memoryBuffer  
upload.addFinishedListener(event -> { // เมื่ออัปโหลดเสร็จสิ้น  
    try {  
        inputStream = memoryBuffer.getInputStream(); // นำเข้าข้อมูลเป็น byte  
        byte[] imageBytes = inputStream.readAllBytes(); // แปลงข้อมูลเป็น byte  
        resource = new StreamResource("dummyImageName.jpg",  
            () -> new ByteArrayInputStream(imageBytes));  
        image = new Image(resource, "dummy image");  
        // สามารถ save วัตถุ image เป็นไฟล์ภาพได้  
  
        image.setHeight("360"); image.setWidth("480");  
        add(image);  
  
        new Notification("Image has been uploaded", 10000).open();  
    } catch (IOException ex) { System.out.println("Have error"); }  
});  
add(upload);
```



ตัวอย่างที่ 2 Events & Listeners



<https://youtu.be/KWgH4s-60hQ>



Outline

- การสร้างส่วนติดต่อประสานผู้ใช้งานด้วย Vaadin
 - Form Inputs
 - Visualization & Interaction
 - Layouts
 - Events And Listeners
- การเรียกใช้ Rest API ผ่าน GUI ด้วย Vaadin



การเรียกใช้ Rest API ผ่าน GUI



Spring Boot

WebClient

RestTemplate

การเรียกใช้งาน Rest API ใน Spring Framework สามารถทำได้โดยอาศัย Spring WebClient หรือ RestTemplate ซึ่งมีรายละเอียดและความแตกต่างกันดังนี้



การเรียกใช้ Rest API ผ่าน GUI

- **RestTemplate**

เป็นคลาสหนึ่งของ Spring Framework ใช้สำหรับการสื่อสารแบบ synchronous HTTP requests ซึ่งคล้ายคลึงกับ JdbcTemplate, JmsTemplate และอื่น ๆ โดยที่ RestTemplate อาศัย Java Servlet API ที่อิงตามหลักการ thread-per-request กล่าวคือเซิร์ฟเวอร์จะบล็อกจนกว่า Client จะได้รับการตอบกลับ ซึ่งทำให้เกิดการถือครองหน่วยความจำและ CPU ไว้แบบสถานะรอ

- **WebClient**

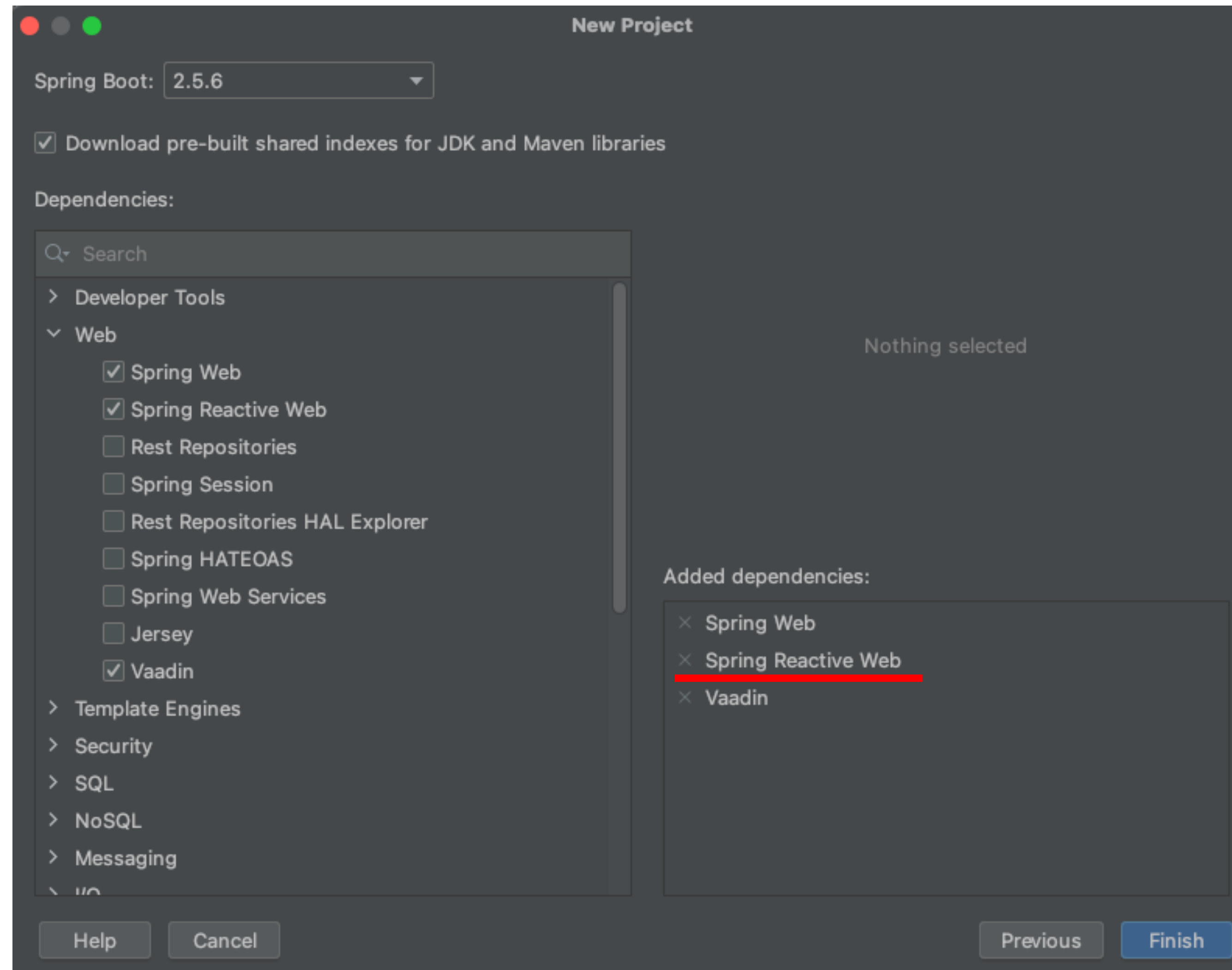
เป็นส่วนหนึ่งของ Web Reactive Framework ที่ช่วยทำให้สร้างเว็บแอปพลิเคชันง่ายขึ้นและยังรองรับการทำงานแบบ Fully non-blocking reactive streams และเป็นการทำงานแบบ Asynchronous ซึ่งผลลัพธ์จะอยู่ในรูปแบบ

- Mono สำหรับการคืนค่าข้อมูลออกมา 0 หรือ 1 ตัว
- Flux สำหรับการคืนค่าข้อมูลออกมา 0 หรือหลายตัว

หมายเหตุ ต้องอาศัย Add Dependencies ของ Spring Reactive Web ด้วย เนื่องจาก WebClient อยู่ในนี้



ตัวอย่างการสื่อสารระหว่าง Service กับ UI





ตัวอย่างการสื่อสารระหว่าง Service กับ UI

```
@Route(value = "index")
public class View extends VerticalLayout {
    private TextField n1, n2, n3;
    private Button btnPlus, btnMinus, btn;

    public View() {
        n1 = new TextField("Number 1");
        n2 = new TextField("Number 2");
        n3 = new TextField("Answer");
        btnPlus = new Button("+");
        btnMinus = new Button("-");
        btn = new Button("Show names");
        add(n1, n2, n3, btnPlus, btnMinus, btn);

        // Insert Code to Event Handler with Call REST API SERVICE
    }
}
```



ตัวอย่าง

```
@RequestMapping(value = "/{n1}+{n2}", method = RequestMethod.GET)
public double calPlus(@PathVariable("n1") double n1,
                      @PathVariable("n2") double n2) {
    return n1+n2;
}
```

```
// การส่งค่าแบบ GET โดยอาศัย String ผ่าน URL
```

```
btnPlus.setOnClickListener(event ->{
```

```
    double num1 = Double.parseDouble(n1.getValue());
```

```
    double num2 = Double.parseDouble(n2.getValue());
```

```
    String out = WebClient.create()
```

```
        .get()
```

```
        .uri("http://localhost:8080/"+num1+"-"+num2)
```

```
        .retrieve()
```

```
        .bodyToMono(String.class)
```

```
        .block();
```

```
        n3.setValue(out);
```

```
});
```

```
// สร้างช่องทางการสื่อสาร
```

```
// กำหนดรูปแบบการสื่อสาร
```

```
// กำหนดที่อยู่ของ Service+Data
```

```
// ให้รอรับข้อมูลกลับมา
```

```
// กำหนด Spec ของ Response
```

```
// Blocking thread
```




ตัวอย่าง

```
public class MyData {  
    private ArrayList<String> name = new ArrayList<>();  
    public ArrayList<String> getName() { return name; }  
    public void setName(ArrayList<String> name) { this.name = name; }  
}
```

```
// คื่นค่ากลับมาแบบ Object
```

```
btn.setOnClickListener(event ->{
```

```
    MyData out = WebClient.create()
```

```
        .get()
```

```
        .uri("http://localhost:8080/names")
```

```
        .retrieve()
```

```
        .bodyToMono(MyData.class)
```

```
        .block();
```

```
    for(String name: out.getName()){
```

```
        new Notification(name, 5000).open();
```

```
    }
```

```
});
```

```
@RequestMapping(value = "/names", method = RequestMethod.GET)
```

```
public MyData getName() {
```

```
    MyData d = new MyData();
```

```
    d.getName().add("Alex");
```

```
    d.getName().add("Peter");
```

```
    d.getName().add("John");
```

```
    return d;
```

```
}
```



ตัวอย่าง

```
@RequestMapping(value = "/test1", method = RequestMethod.POST)
public String testPost1(@RequestBody MultiValueMap<String, String> n) {
    Map<String, String> d = n.toSingleValueMap();
    double out = Double.parseDouble(d.get("n1")) - Double.parseDouble(d.get("n2"));
    return out+"";
}
```

หมายเหตุ @RequestBody ใช้สำหรับการรับค่าผ่าน Body ของ Message

```
// การส่งค่าแบบ POST โดยอาศัย MultiValueMap
btnMinus.setOnClickListener(event ->{
    // ส่งหลายค่าผ่าน FORM ด้วย MultiValueMap ในรูปแบบ String และ String
    MultiValueMap<String, String> formData = new LinkedMultiValueMap<>();
    formData.add("n1", n1.getValue()); // key คือ n1, value คือ n1.getValue()
    formData.add("n2", n2.getValue());

    String out = WebClient.create()
        .post()
        .uri("http://localhost:8080/test1")
        // เป็นการกำหนดชนิดข้อมูลของ Request Message ในส่วนของ Body
        .contentType(MediaType.APPLICATION_FORM_URLENCODED)
        .body(BodyInserters.fromFormData(formData))
        .retrieve()
        .bodyToMono(String.class)
        .block();
    n3.setValue(out);
});
```




ตัวอย่าง

// การส่งค่าแบบ POST โดยอาศัย Object

```
btnMinus.setOnClickListener(event ->{
```

```
    double num1 = Double.parseDouble(n1.getValue());
```

```
    double num2 = Double.parseDouble(n2.getValue());
```

```
    String out = WebClient.create()
```

```
        .post()
```

```
        .uri("http://localhost:8080/test2")
```

```
        .body(Mono.just(new MyNumber(num1, num2)), MyNumber.class)
```

```
        .retrieve()
```

```
        .bodyToMono(String.class)
```

```
        .block();
```

```
    n3.setValue(out);
```

```
});
```

```
public class MyNumber {
    private double n1,n2;
    public MyNumber(double n1, double n2) {
        this.n1 = n1; this.n2 = n2;
    }
    public double getN1() { return n1; }
    public double getN2() { return n2; }
    public void setN1(double n1) { this.n1 = n1; }
    public void setN2(double n2) { this.n2 = n2; }
}
```

```
@RequestMapping(value = "/test2", method = RequestMethod.POST)
public String testPost2(@RequestBody MyNumber n) {
    double out = n.getN1() - n.getN2();
    return out+"";
}
```



ตัวอย่างที่ 2

localhost

Number 1
4

Number 2
5

Answer
9.0

+

-

Show names