

Lab 3

Legacy migration

Swing to Vaadin

```
1 Box dataInputBox = Box.createHorizontalBox();
2 dataInputBox.add(new JLabel("Type value:"));
3 final JTextField inputField = new JTextField();
4 dataInputBox.add(inputField);
5 JButton button = new JButton("Submit");
6 ▼ button.addActionListener(new ActionListener() {
7     @Override
8     ▼ public void actionPerformed(ActionEvent e) {
9         facade.save(inputField.getText());
10 ▲     }
11 ▲ });
12 add(dataInputBox);
```

```
1 HorizontalLayout dataInputBox = new HorizontalLayout();
2 dataInputBox.addComponent(new Label("Type value:"));
3 final TextField inputField = new TextField();
4 dataInputBox.addComponent(inputField);
5 Button button = new Button("Submit");
6 ▼ button.addClickListener(new Button.ClickListener() {
7     @Override
8     ▼ public void buttonClick(Button.ClickEvent event) {
9         facade.save(inputField.getValue());
10 ▲     }
11 ▲ });
12 addComponent(dataInputBox);
```

<https://vaadin.com/swing>

JSP to Vaadin

- Reuse service code
- Offer richer UI interaction without writing JavaScript
- Vaadin Designer can help recreate views to save time

What we'll learn in this lab:

- Migrating a legacy JSP view to Vaadin
- Using Vaadin Designer
- Validation
- Keyboard shortcuts

Add new customer

First name

Email







Last name

Status

ImportedLead

Cancel

Add customer

-  Customer List
-  Analyze
-  Map
-  About
-  Legacy
-  New Customer

Add new Customer

First name

Last Name

Email

Status

Contacted

▼

Cancel

Add Customer

Steps:

- Familiarize yourself with `addCustomer.jsp` and `AddCustomer.java`
- Create a new Vaadin Design called `AddCustomerDesign`
- Extend `AddCustomerDesign.java` and implement View
- Bind form to Bean and attach `discard()/commit()` logic to buttons
- Add validation for fields (Hint: use `setRequired` and `BeanValidator`)
- Set keyboard shortcuts for save/cancel buttons