

Using the valo-theme everything looks nice:

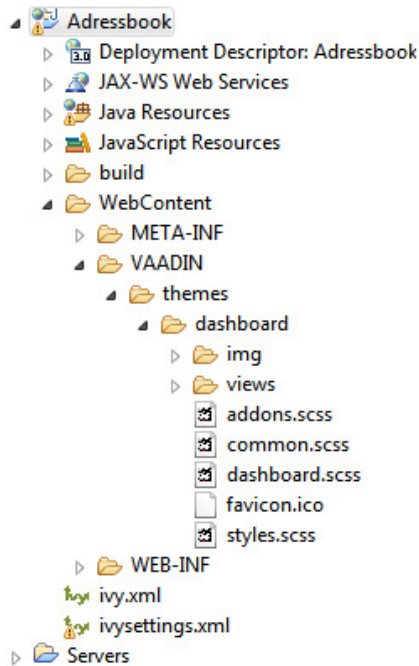
```
AddressbookUI.java | ContactForm.java | styles.scss
19  * the same instance, add @PreserveOnRefresh.
20  */
21  @Title("Addressbook")
22  @Theme("valo")
23  public class AddressbookUI extends UI {
24
25      private static final long serialVersionUID = 1L;
26      /* Hundreds of widgets.
27       * Vaadin's user interface components are just Java objects that encapsulate
28       * and handle cross-browser support and client-server communication. The
29       * default Vaadin components are in the com.vaadin.ui package and there
30       * are over 500 more in vaadin.com/directory.
31       */
32      TextField filter = new TextField();
33      Grid contactList = new Grid();
34      Button newContact = new Button("New contact");
35
36      // ContactForm is an example of a custom component class
37      ContactForm contactForm = new ContactForm();
38
39      // ContactService is a in-memory mock DAO that mimics
40      // a real-world datasource. Typically implemented for
41      // example as EJB or Spring Data based service.
42      ContactService service = ContactService.createDemoService();
43
44
45      /* The "Main method".
46
47      protected void init(VaadinRequest request) {}
48
49
50
51
52
53
54
55
56
57
58      private void configureComponents() {
59          /* Synchronous event handling.
60           *
61           * Receive user interaction events on the server-side. This allows you
62           * to synchronously handle those events. Vaadin automatically sends
63           * only the needed changes to the web page without loading a new page.
64           */
65          newContact.addClickListener(e -> contactForm.edit(new Contact()));
66
67          filter.setInputPrompt("Filter contacts...");
68          filter.addTextChangeListener(e -> refreshContacts(e.getText()));
69
70          contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
71          contactList.setColumnOrder("firstName", "lastName", "email");
72          contactList.removeColumn("id");
73          contactList.removeColumn("birthDate");
74          contactList.removeColumn("phone");
75          // contactList.setStyleName("valo");
76          contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
77          contactList.addSelectionListener(e
78              -> contactForm.edit((Contact) contactList.getSelectedRow()));
79          refreshContacts();
80      }
81
82      /* Robust layout
```

Addressbook

localhost:8080/Adressbook/

Filter contacts...

First Name	Last Name	Email
George	White	george@white.com
Daniel	Thompson	daniel@thompson.com
Timothy	Jones	timothy@jones.com
Peter	Wilson	peter@wilson.com
Dan	Robinson	dan@robinson.com
Dan	Davis	dan@davis.com
Olivia	Davis	olivia@davis.com
Dan	Smith	dan@smith.com
Daniel	Anderson	daniel@anderson.com
Alice	Thomas	alice@thomas.com
Linda	Harris	linda@harris.com
Daniel	Robinson	daniel@robinson.com
Mike	Young	mike@young.com
Umberto	Anderson	umberto@anderson.com
Scott	Thompson	scott@thompson.com
Rene	Martin	rene@martin.com
Lisa	Martin	lisa@martin.com
Peter	Martin	peter@martin.com
Brian	Wilson	brian@wilson.com
Scott	Miller	scott@miller.com
John	White	john@white.com
Peter	Johnson	peter@johnson.com



Using the dashboard-theme

... :

```

19 * the same instance, add @PreserveOnRefresh.
20 */
21 @Title("Addressbook")
22 @Theme("dashboard")
23 public class AddressbookUI extends UI {
24
25     private static final long serialVersionUID = 1L;
26     /* Hundreds of widgets.
27     * Vaadin's user interface components are just Java objects that encapsulate
28     * and handle cross-browser support and client-server communication. The
29     * default Vaadin components are in the com.vaadin.ui package and there
30     * are over 500 more in vaadin.com/directory.
31     */
32     TextField filter = new TextField();
33     Grid contactList = new Grid();
34     Button newContact = new Button("New contact");
35
36     // ContactForm is an example of a custom component class
37     ContactForm contactForm = new ContactForm();
38
39     // ContactService is a in-memory mock DAO that mimics
40     // a real-world datasource. Typically implemented for
41     // example as EJB or Spring Data based service.
42     ContactService service = ContactService.createDemoService();
43
44
45     /* The "Main method".
46     protected void init(VaadinRequest request) {}
47
48
49     private void configureComponents() {
50         /* Synchronous event handling.
51         *
52         * Receive user interaction events on the server-side. This allows you
53         * to synchronously handle those events. Vaadin automatically sends
54         * only the needed changes to the web page without loading a new page.
55         */
56         newContact.addClickListener(e -> contactForm.edit(new Contact()));
57
58         filter.setInputPrompt("Filter contacts...");
59         filter.addTextChangeListener(e -> refreshContacts(e.getText()));
60
61         contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
62         contactList.setColumnOrder("firstName", "lastName", "email");
63         contactList.removeColumn("id");
64         contactList.removeColumn("birthDate");
65         contactList.removeColumn("phone");
66         contactList.setStyleName("valo");
67         contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
68         contactList.addSelectionListener(e
69             -> contactForm.edit((Contact) contactList.getSelectedRow()));
70         refreshContacts();
71     }
72
73
74
75
76
77
78
79
80
81

```

Project Explorer

- Adressbook
 - Deployment Descriptor: Adressbook
 - JAX-WS Web Services
 - Java Resources
 - JavaScript Resources
 - build
 - WebContent
 - META-INF
 - VAADIN
 - themes
 - dashboard
 - img
 - views
 - addons.scss
 - common.scss
 - dashboard.scss
 - favicon.ico
 - styles.scss
 - WEB-INF
 - ivy.xml
 - ivysettings.xml
 - Servers

AddressbookUI.java ContactForm.java styles.scss

```
1 // Import all the mixins for this theme
2 @import "dashboard";
3 @import "../valo/valo.scss";
4
5 .dashboard {
6   // Output the actual theme selectors
7   @include dashboard;
8   @include valo;
9 }
```

Addressbook

localhost:8080/Adressbook/

Suchen

Filter contacts...

New contact

First Name	Last Name	Email
------------	-----------	-------

Project Explorer | Type Hierarchy

- Servers
- transportplanung
 - Deployment Descriptor: transportplanung
 - JAX-WS Web Services
 - Java Resources
 - JavaScript Resources
 - build
 - test
 - WebContent
 - ivy.xml
 - ivysettings.xml


```
113     dashboardPanels.addComponent(buildDienstleister());
114     dashboardPanels.addComponent(buildLokationen());
115     dashboardPanels.addComponent(buildFahrzeuge());
116
117     return dashboardPanels;
118 }
119
120 private Component buildKontakte() {
121     TextArea notes = new TextArea("Kontakte");
122     notes.setValue("Hier kommt die Tabelle der Kontakte rein.");
123     notes.setSizeFull();
124     notes.addStyleName(ValoTheme.TEXTAREA_BORDERLESS);
125     Component panel = createContentWrapper(notes, "");
126     panel.addStyleName("notes");
127     return panel;
128 }
129
130 private Component buildDienstleister() throws IllegalAccessException {
131
132     newContact.addClickListener(e -> contactForm.edit(new Contact()));
133
134     filter.setInputPrompt("Filter contacts...");
135     filter.addTextChangeListener(e -> refreshContacts(e.getText()));
136
137     contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
138     contactList.setColumnOrder("firstName", "lastName", "email");
139     contactList.removeColumn("id");
140     contactList.removeColumn("birthDate");
141     contactList.removeColumn("phone");
142     contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
143     contactList.setFrozenColumnCount(3);
144     contactList.addSelectionListener(e
145         -> contactForm.edit((Contact) contactList.getSelectedRow()));
146     refreshContacts();
147
148     HorizontalLayout actions = new HorizontalLayout(filter, newContact);
149     actions.setWidth("100%");
150     filter.setWidth("100%");
151     actions.setExpandRatio(filter, 1);
152
153     VerticalLayout left = new VerticalLayout(actions, contactList);
154     left.setCaption("Dienstleister");
155     left.setSizeFull();
156     contactList.setSizeFull();
157     left.setExpandRatio(contactList, 1);
158
159     Component panel = createContentWrapper(left, "Dienstleister");
160     //panel.addStyleName("notes");
161     return panel;
162 }
163
164 private Component buildLokationen() {
165     TextArea notes = new TextArea("Lokationen");
166     notes.setValue("Hier kommt die Tabelle der Lokationen rein.");
167     notes.setSizeFull();
168     notes.addStyleName(ValoTheme.TEXTAREA_BORDERLESS);
169     Component panel = createContentWrapper(notes, "");
170     panel.addStyleName("notes");
171     return panel;
172 }
```






Similar Problem with
The Quick-Ticket-Dash-
Board sources
... :

http://localh...sportplanung/ x +

localhost:8080/transportplanung/#stammdaten

Daimler STM-System

 Kate King ▾

-  Arbeitsbereich
-  Kontakte
-  Dienstleister
-  Lokationen
-  Fahrzeuge

Stammdaten

KONTAKTE

Hier kommt die Tabelle der Kontakte rein.

DIENSTLEISTER

Filter contacts...

First Name	Last Name	Email
------------	-----------	-------

LOKATIONEN

Hier kommt die Tabelle der Lokationen rein.

FAHRZEUGE

Hier kommt die Tabelle der Fahrzeuge rein.

Project Explorer | Type Hierarchy

- Servers
- transportplanung
 - Deployment Descriptor: transportplanung
 - JAX-WS Web Services
 - Java Resources
 - JavaScript Resources
 - build
 - test
 - WebContent
 - ivy.xml
 - ivysettings.xml

DienstleisterView.java | TransportplanungUI.java | StammdatenView.java

```
79
80     titleLabel = new Label("Dienstleister");
81     titleLabel.setId(TITLE_ID);
82     titleLabel.setSizeUndefined();
83     titleLabel.addStyleName(ValoTheme.LABEL_H1);
84     titleLabel.addStyleName(ValoTheme.LABEL_NO_MARGIN);
85     header.addComponent(titleLabel);
86
87     return header;
88 }
89
90 private Component buildContent() throws IllegalArgumentException {
91     dashboardPanels = new CssLayout();
92     dashboardPanels.addStyleName("dashboard-panels");
93     Responsive.makeResponsive(dashboardPanels);
94     dashboardPanels.addComponent(buildDienstleister());
95     return dashboardPanels;
96 }
97
98 private Component buildDienstleister() throws IllegalArgumentException {
99
100     newContact.addClickListener(e -> contactForm.edit(new Contact()));
101
102     filter.setInputPrompt("Filter contacts...");
103     filter.addTextChangeListener(e -> refreshContacts(e.getText()));
104
105     contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
106     contactList.setColumnOrder("firstName", "lastName", "email");
107     contactList.removeColumn("id");
108     contactList.removeColumn("birthDate");
109     contactList.removeColumn("phone");
110     contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
111     contactList.setFrozenColumnCount(3);
112     contactList.addSelectionListener(e
113         -> contactForm.edit((Contact) contactList.getSelectedRow()));
114     refreshContacts();
115
116     HorizontalLayout actions = new HorizontalLayout(filter, newContact);
117     actions.setWidth("100%");
118     filter.setWidth("100%");
119     actions.setExpandRatio(filter, 1);
120
121     VerticalLayout left = new VerticalLayout(actions, contactList);
122     left.setSizeFull();
123     contactList.setSizeFull();
124     left.setExpandRatio(contactList, 1);
125
126     return left;
127 }
128
129 @Override
130 public void buttonClick(ClickEvent event) {
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Search

Tomcat v8.0 Server at localhost [Stopped, Synchronized]

Browser window showing a web application interface for "Daimler STM-System". The address bar shows "localhost:8080/transportplanung/#dienstleister". The page title is "Dienstleister".

Left sidebar navigation menu:

- Arbeitsbereich
- Kontakte
- Dienstleister**
- Lokationen
- Fahrzeuge

Header area:

Daimler STM-System

Kate King

Filter contacts... New contact

First Name	Last Name	Email
George	White	george@white.com
Daniel	Thompson	daniel@thompson.com
Timothy	Jones	timothy@jones.com
Peter	Wilson	peter@wilson.com
Dan	Robinson	dan@robinson.com
Dan	Davis	dan@davis.com
Olivia	Davis	olivia@davis.com
Dan	Smith	dan@smith.com
Daniel	Anderson	daniel@anderson.com
Alice	Thomas	alice@thomas.com
Linda	Harris	linda@harris.com
Daniel	Robinson	daniel@robinson.com
Mike	Young	mike@young.com
Umberto	Anderson	umberto@anderson.com
Scott	Thompson	scott@thompson.com
Rene	Martin	rene@martin.com
Lisa	Martin	lisa@martin.com
Peter	Martin	peter@martin.com
Brian	Wilson	brian@wilson.com
Scott	Miller	scott@miller.com
John	White	john@white.com
Peter	Johnson	peter@johnson.com
Scott	Brown	scott@brown.com