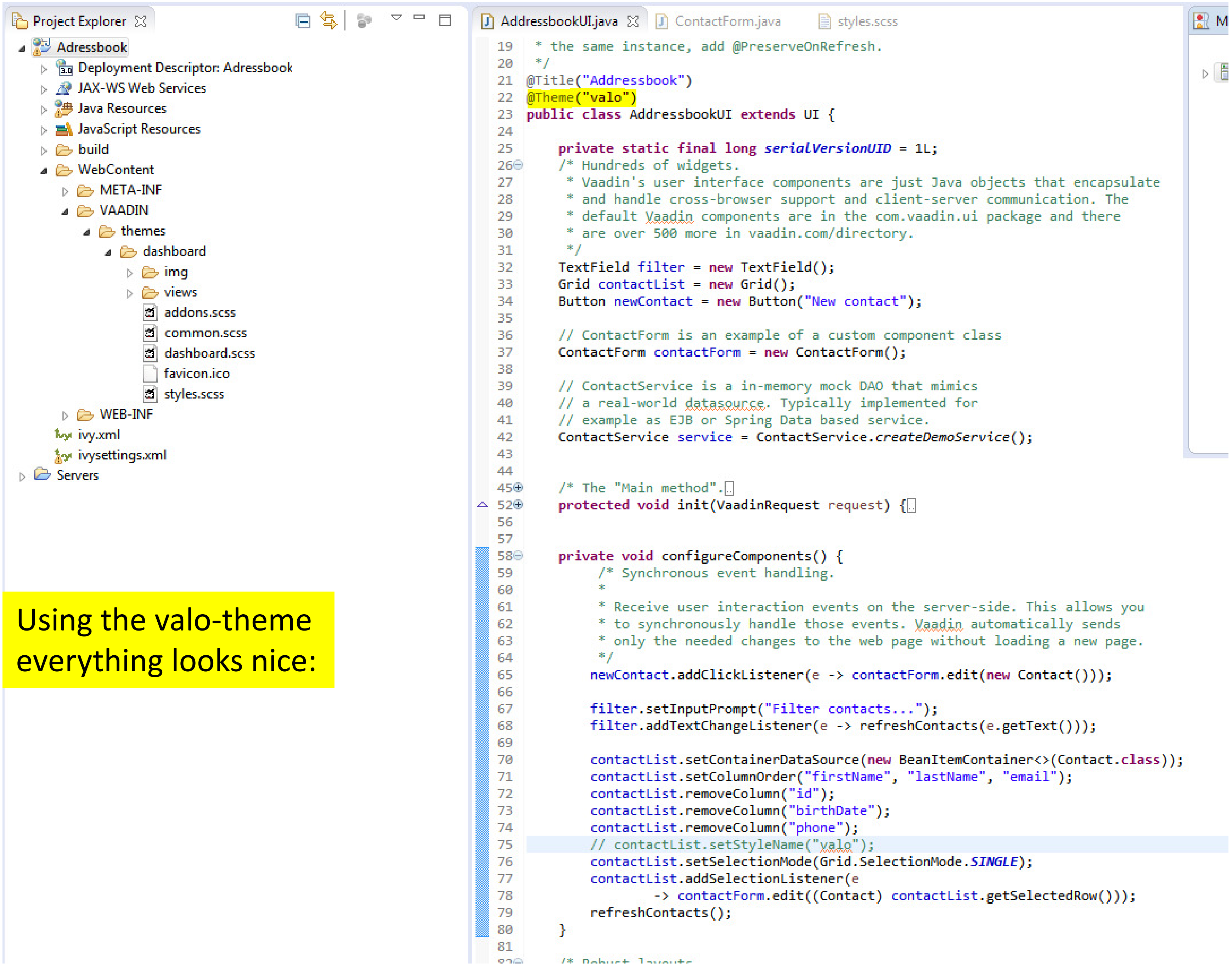
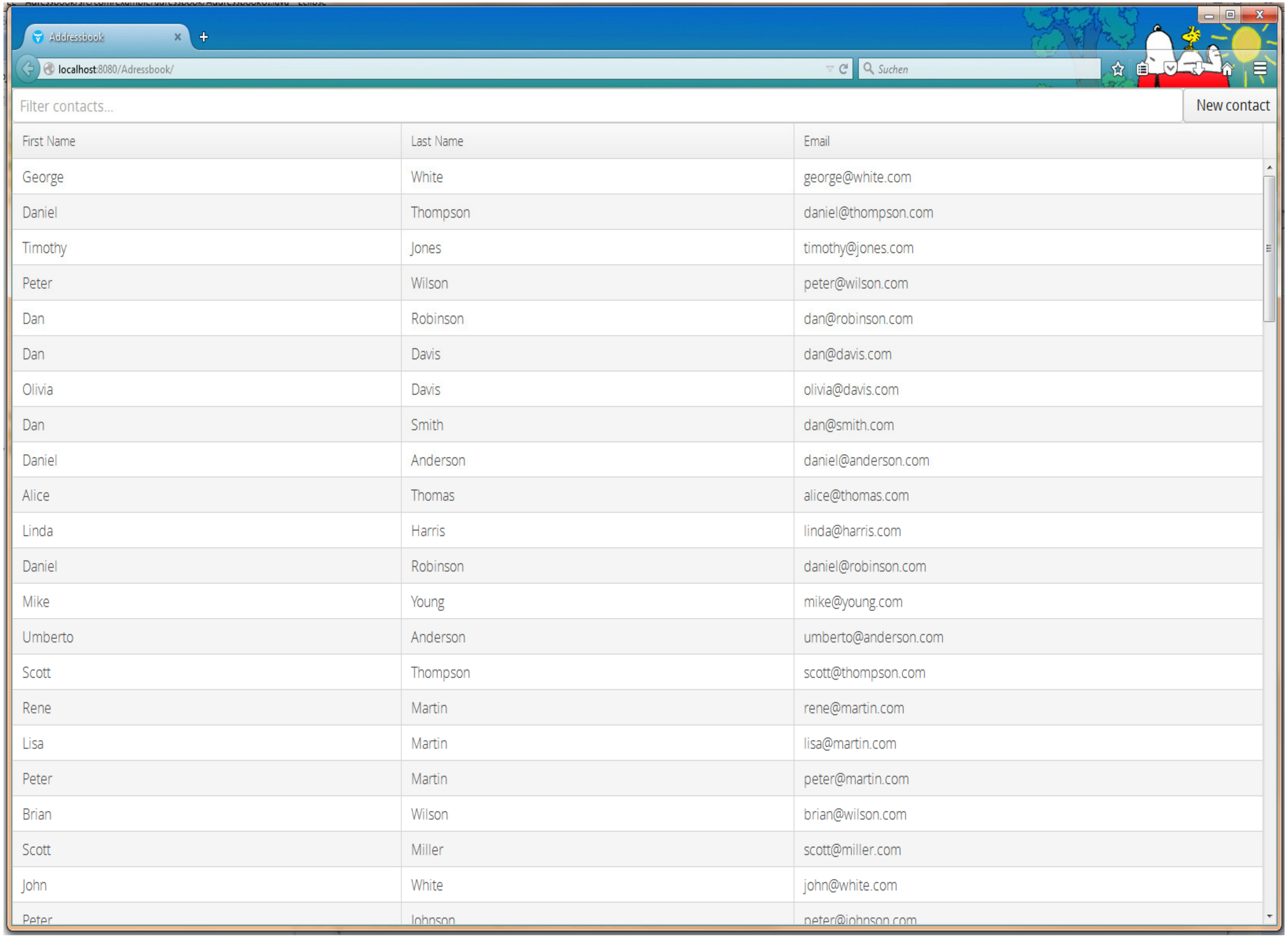
Using the valo-theme everything looks nice:

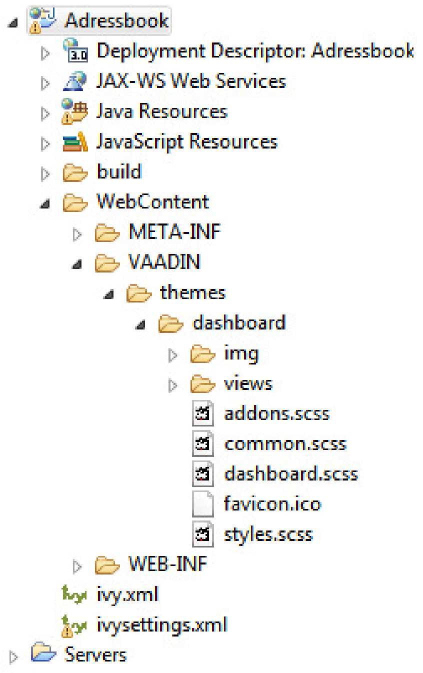


```
Adressbook
  Deployment Descriptor: Adressbook
  JAX-WS Web Services
  Java Resources
  JavaScript Resources
  build
  WebContent
    META-INF
    VAADIN
      themes
        dashboard
          img
          views
          addons.scss
          common.scss
          dashboard.scss
          favicon.ico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

```java
 * the same instance, add @PreserveOnRefresh.
 */
@Title("Addressbook")
@Theme("valo")
public class AddressbookUI extends UI {

    private static final long serialVersionUID = 1L;
    /* Hundreds of widgets.
     * Vaadin's user interface components are just Java objects that encapsulate
     * and handle cross-browser support and client-server communication. The
     * default Vaadin components are in the com.vaadin.ui package and there
     * are over 500 more in vaadin.com/directory.
     */
    TextField filter = new TextField();
    Grid contactList = new Grid();
    Button newContact = new Button("New contact");

    // ContactForm is an example of a custom component class
    ContactForm contactForm = new ContactForm();

    // ContactService is a in-memory mock DAO that mimics
    // a real-world datasource. Typically implemented for
    // example as EJB or Spring Data based service.
    ContactService service = ContactService.createDemoService();


    /* The "Main method".
    protected void init(VaadinRequest request) {


    private void configureComponents() {
         /* Synchronous event handling.
          *
          * Receive user interaction events on the server-side. This allows you
          * to synchronously handle those events. Vaadin automatically sends
          * only the needed changes to the web page without loading a new page.
          */
        newContact.addClickListener(e -> contactForm.edit(new Contact()));

        filter.setInputPrompt("Filter contacts...");
        filter.addTextChangeListener(e -> refreshContacts(e.getText()));

        contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
        contactList.setColumnOrder("firstName", "lastName", "email");
        contactList.removeColumn("id");
        contactList.removeColumn("birthDate");
        contactList.removeColumn("phone");
        // contactList.setStyleName("valo");
        contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
        contactList.addSelectionListener(e
                -> contactForm.edit((Contact) contactList.getSelectedRow()));
        refreshContacts();
    }

    /* Robust layouts
```

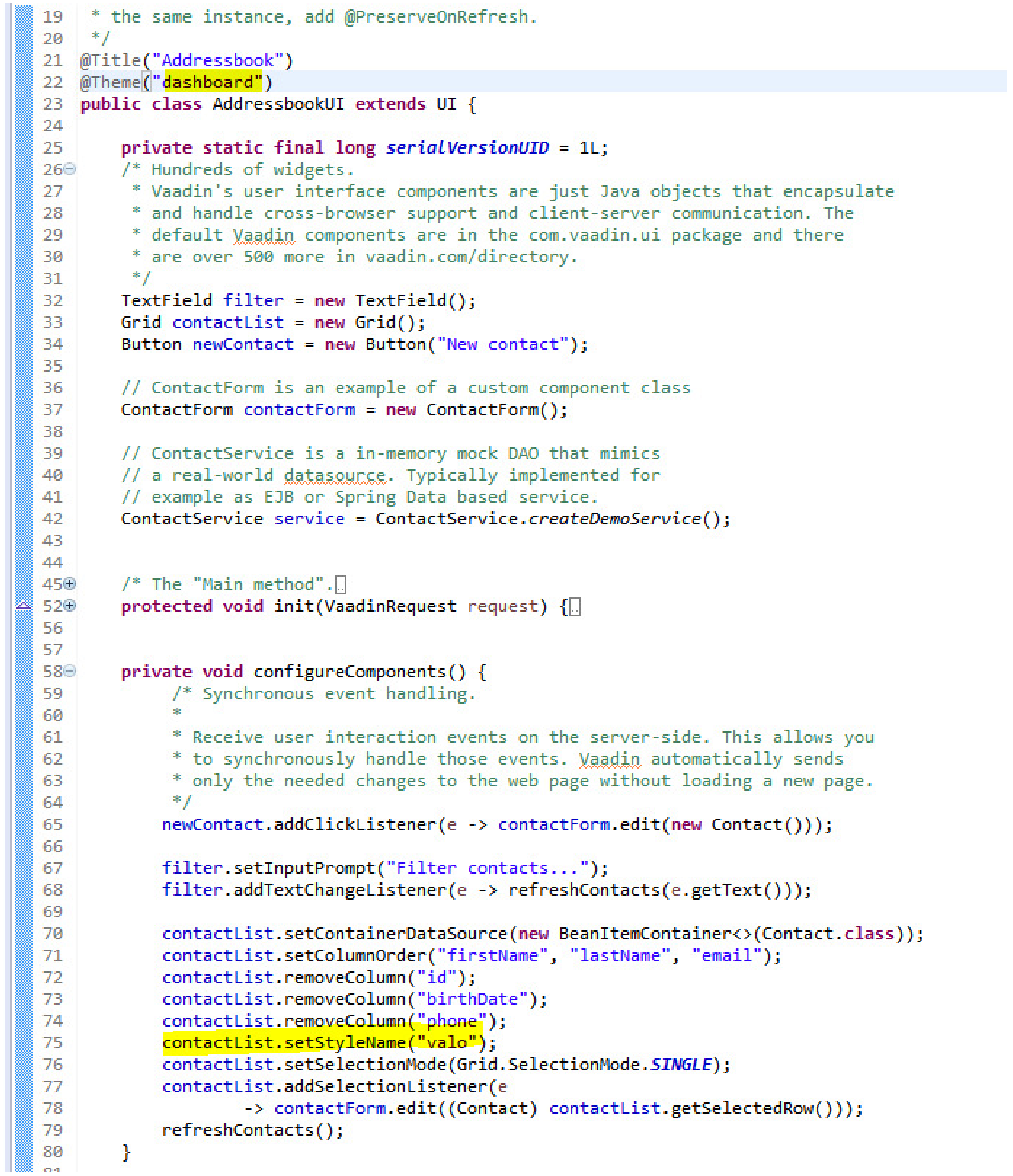


First Name	Last Name	Email
George	White	george@white.com
Daniel	Thompson	daniel@thompson.com
Timothy	Jones	timothy@jones.com
Peter	Wilson	peter@wilson.com
Dan	Robinson	dan@robinson.com
Dan	Davis	dan@davis.com
Olivia	Davis	olivia@davis.com
Dan	Smith	dan@smith.com
Daniel	Anderson	daniel@anderson.com
Alice	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

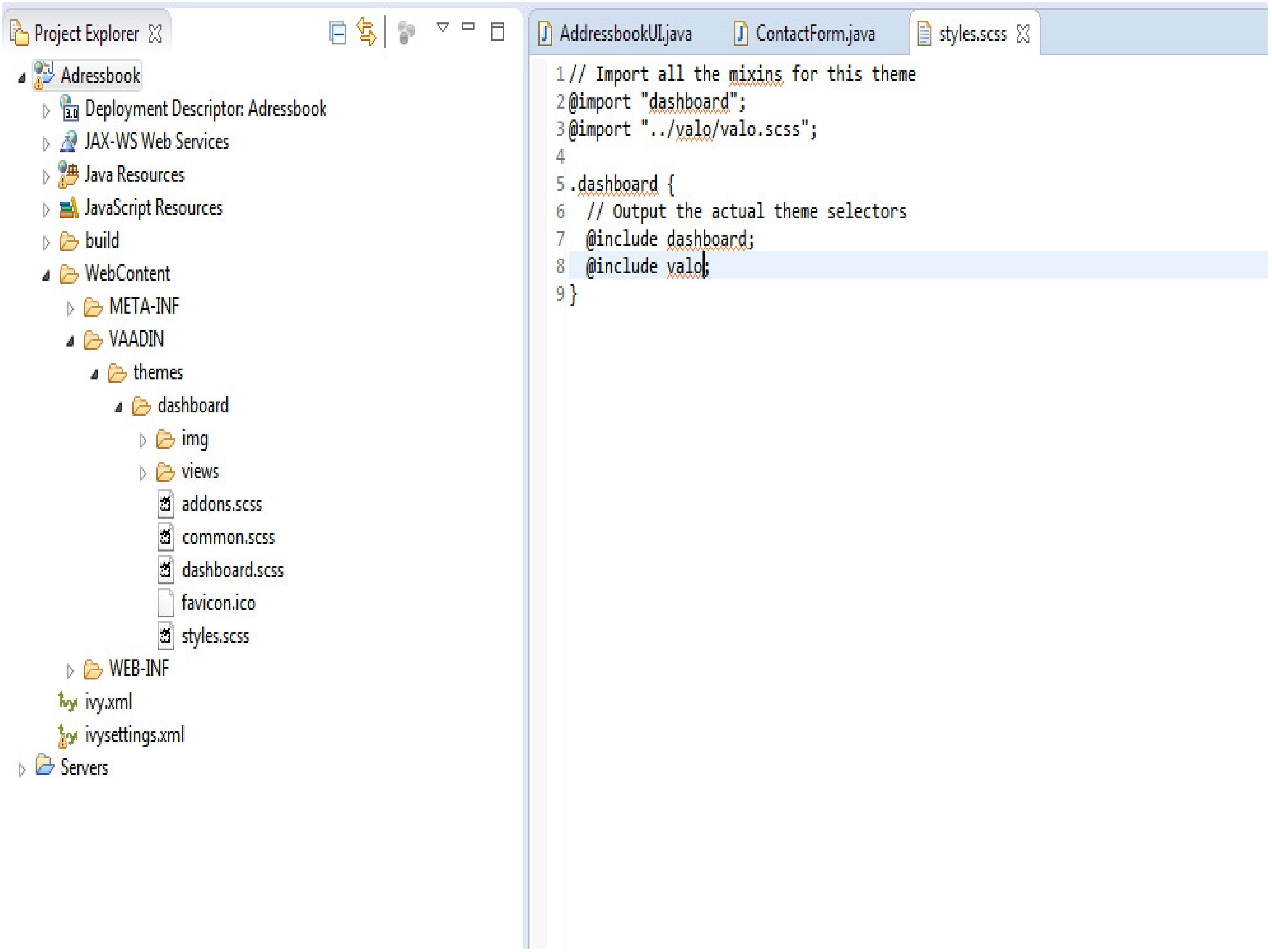


```
Adressbook
  Deployment Descriptor: Adressbook
  JAX-WS Web Services
  Java Resources
  JavaScript Resources
  build
  WebContent
    META-INF
    VAADIN
      themes
        dashboard
          img
          views
          addons.scss
          common.scss
          dashboard.scss
          favicon.ico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

Using the dashboard-theme … :

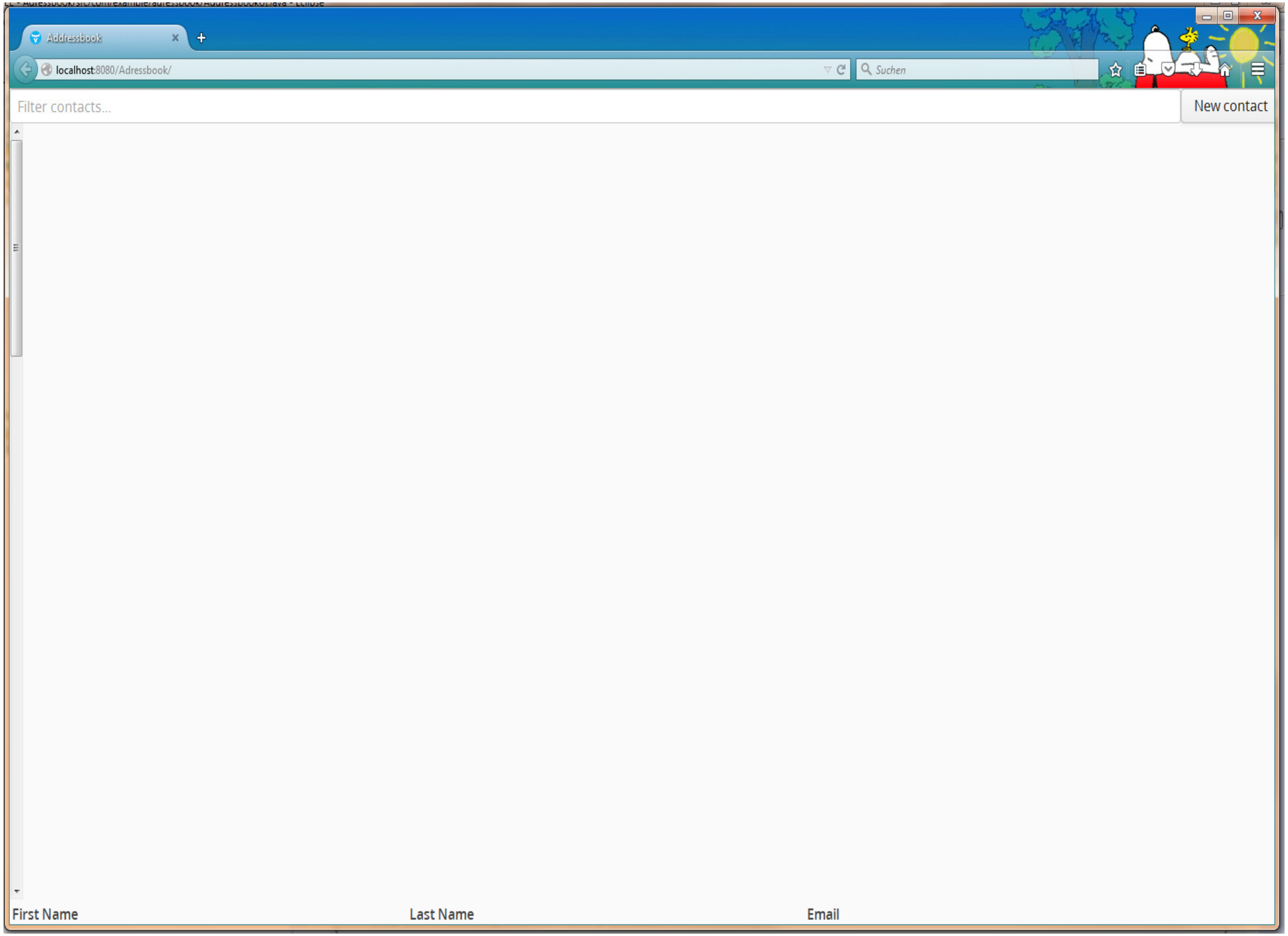


```java
 * the same instance, add @PreserveOnRefresh.
 */
@Title("Addressbook")
@Theme("dashboard")
public class AddressbookUI extends UI {

    private static final long serialVersionUID = 1L;
    /* Hundreds of widgets.
     * Vaadin's user interface components are just Java objects that encapsulate
     * and handle cross-browser support and client-server communication. The
     * default Vaadin components are in the com.vaadin.ui package and there
     * are over 500 more in vaadin.com/directory.
     */
    TextField filter = new TextField();
    Grid contactList = new Grid();
    Button newContact = new Button("New contact");

    // ContactForm is an example of a custom component class
    ContactForm contactForm = new ContactForm();

    // ContactService is a in-memory mock DAO that mimics
    // a real-world datasource. Typically implemented for
    // example as EJB or Spring Data based service.
    ContactService service = ContactService.createDemoService();


    /* The "Main method".
    protected void init(VaadinRequest request) {


    private void configureComponents() {
         /* Synchronous event handling.
          *
          * Receive user interaction events on the server-side. This allows you
          * to synchronously handle those events. Vaadin automatically sends
          * only the needed changes to the web page without loading a new page.
          */
        newContact.addClickListener(e -> contactForm.edit(new Contact()));

        filter.setInputPrompt("Filter contacts...");
        filter.addTextChangeListener(e -> refreshContacts(e.getText()));

        contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
        contactList.setColumnOrder("firstName", "lastName", "email");
        contactList.removeColumn("id");
        contactList.removeColumn("birthDate");
        contactList.removeColumn("phone");
        contactList.setStyleName("valo");
        contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
        contactList.addSelectionListener(e
                -> contactForm.edit((Contact) contactList.getSelectedRow()));
        refreshContacts();
    }
```



```
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
```

```
// Import all the mixins for this theme
@import "dashboard";
@import "../valo/valo.scss";

.dashboard {
  // Output the actual theme selectors
  @include dashboard;
  @include valo;
}
```



Addressbook

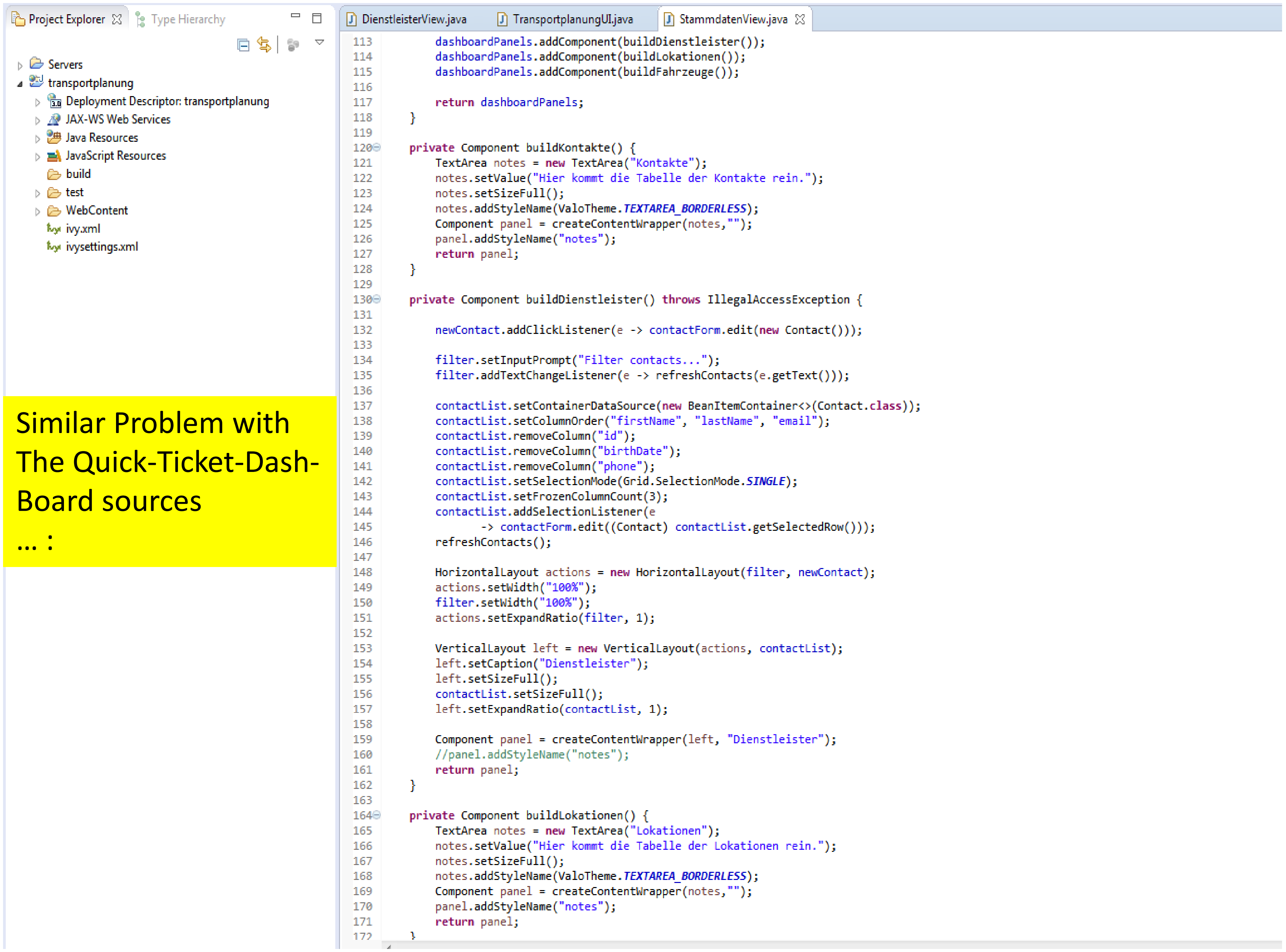
localhost:8080/Adressbook/

Suchen

Filter contacts...

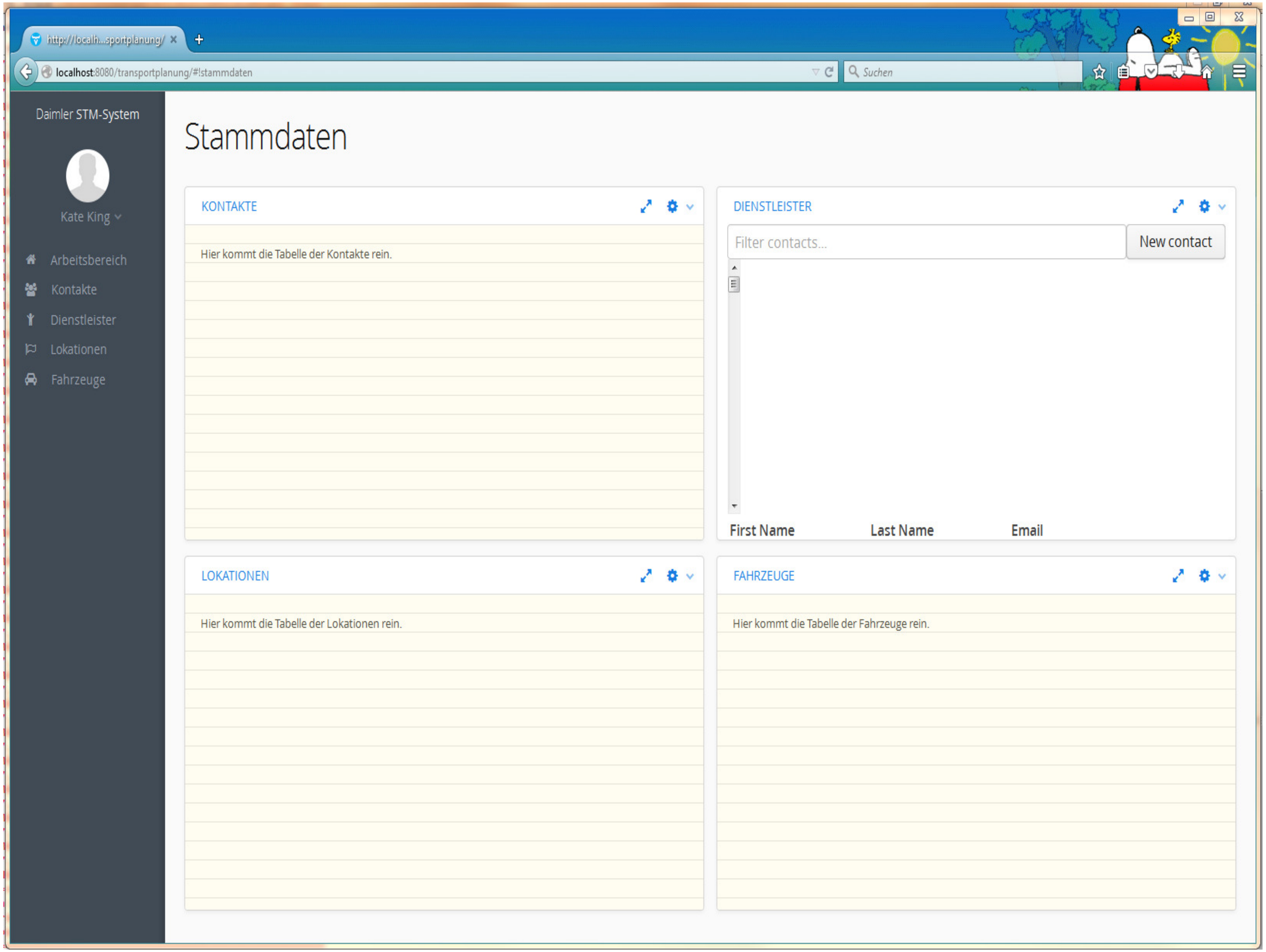
New contact

First Name Last Name Email



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

```java
        dashboardPanels.addComponent(buildDienstleister());
        dashboardPanels.addComponent(buildLokationen());
        dashboardPanels.addComponent(buildFahrzeuge());

        return dashboardPanels;
    }

    private Component buildKontakte() {
        TextArea notes = new TextArea("Kontakte");
        notes.setValue("Hier kommt die Tabelle der Kontakte rein.");
        notes.setSizeFull();
        notes.addStyleName(ValoTheme.TEXTAREA_BORDERLESS);
        Component panel = createContentWrapper(notes,"");
        panel.addStyleName("notes");
        return panel;
    }

    private Component buildDienstleister() throws IllegalAccessException {

        newContact.addClickListener(e -> contactForm.edit(new Contact()));

        filter.setInputPrompt("Filter contacts...");
        filter.addTextChangeListener(e -> refreshContacts(e.getText()));

        contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
        contactList.setColumnOrder("firstName", "lastName", "email");
        contactList.removeColumn("id");
        contactList.removeColumn("birthDate");
        contactList.removeColumn("phone");
        contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
        contactList.setFrozenColumnCount(3);
        contactList.addSelectionListener(e
                -> contactForm.edit((Contact) contactList.getSelectedRow()));
        refreshContacts();

        HorizontalLayout actions = new HorizontalLayout(filter, newContact);
        actions.setWidth("100%");
        filter.setWidth("100%");
        actions.setExpandRatio(filter, 1);

        VerticalLayout left = new VerticalLayout(actions, contactList);
        left.setCaption("Dienstleister");
        left.setSizeFull();
        contactList.setSizeFull();
        left.setExpandRatio(contactList, 1);

        Component panel = createContentWrapper(left, "Dienstleister");
        //panel.addStyleName("notes");
        return panel;
    }

    private Component buildLokationen() {
        TextArea notes = new TextArea("Lokationen");
        notes.setValue("Hier kommt die Tabelle der Lokationen rein.");
        notes.setSizeFull();
        notes.addStyleName(ValoTheme.TEXTAREA_BORDERLESS);
        Component panel = createContentWrapper(notes,"");
        panel.addStyleName("notes");
        return panel;
    }
```



Stammdaten

KONTAKTE

Hier kommt die Tabelle der Kontakte rein.

DIENSTLEISTER

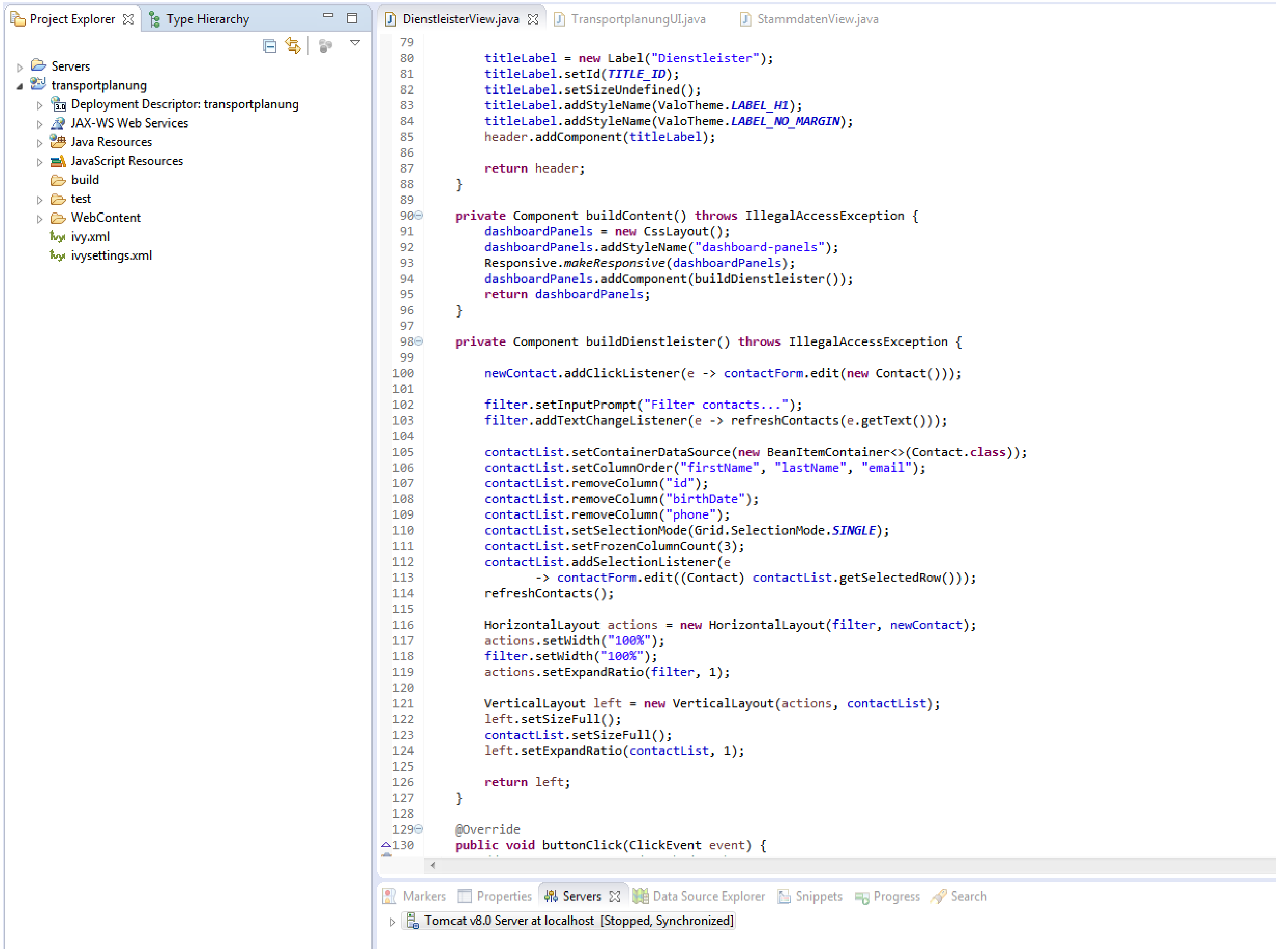
First Name	Last Name	Email

LOKATIONEN

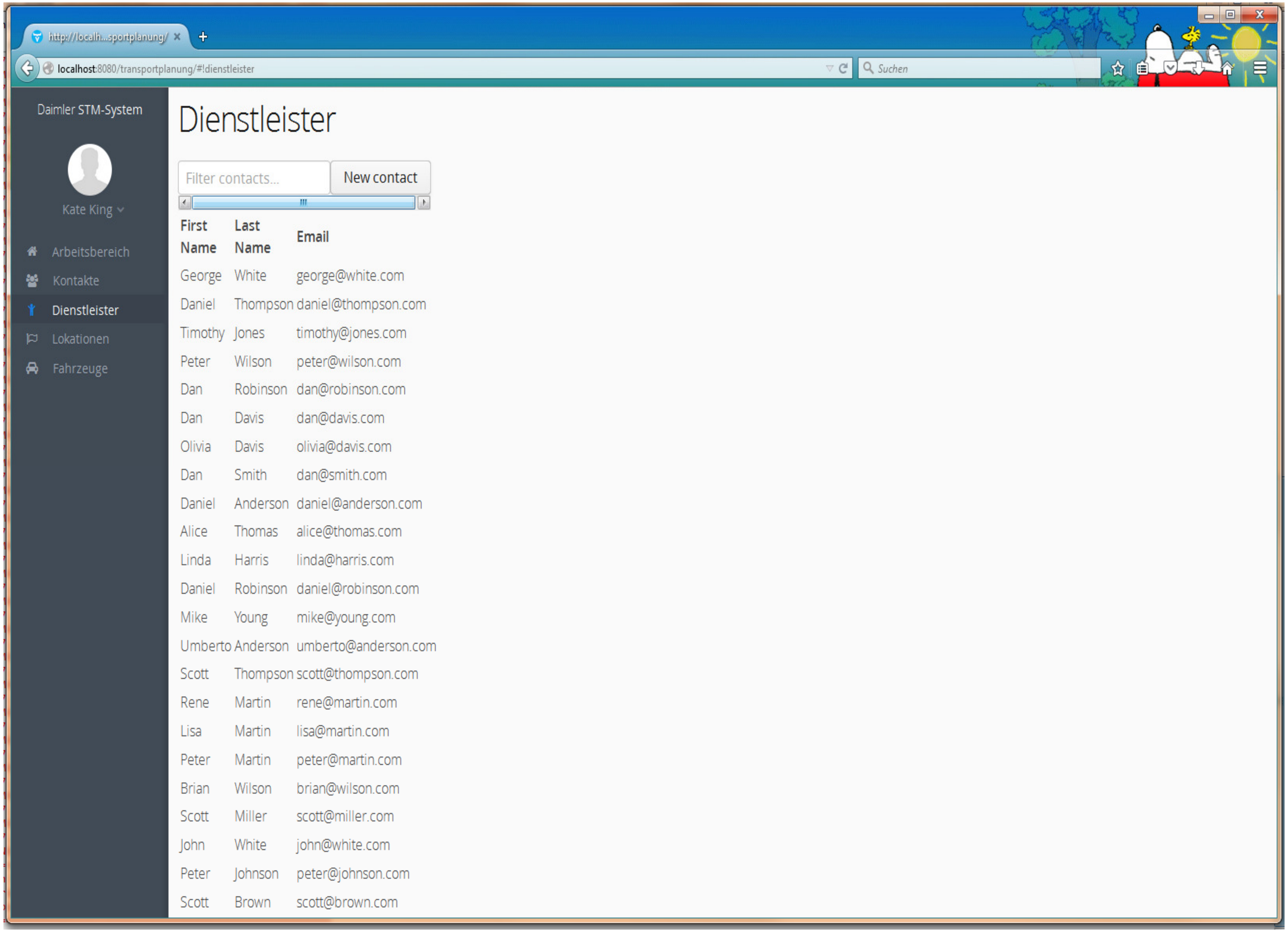
Hier kommt die Tabelle der Lokationen rein.

FAHRZEUGE

Hier kommt die Tabelle der Fahrzeuge rein.



```java

        titleLabel = new Label("Dienstleister");
        titleLabel.setId(TITLE_ID);
        titleLabel.setSizeUndefined();
        titleLabel.addStyleName(ValoTheme.LABEL_H1);
        titleLabel.addStyleName(ValoTheme.LABEL_NO_MARGIN);
        header.addComponent(titleLabel);

        return header;
    }

    private Component buildContent() throws IllegalAccessException {
        dashboardPanels = new CssLayout();
        dashboardPanels.addStyleName("dashboard-panels");
        Responsive.makeResponsive(dashboardPanels);
        dashboardPanels.addComponent(buildDienstleister());
        return dashboardPanels;
    }

    private Component buildDienstleister() throws IllegalAccessException {

        newContact.addClickListener(e -> contactForm.edit(new Contact()));

        filter.setInputPrompt("Filter contacts...");
        filter.addTextChangeListener(e -> refreshContacts(e.getText()));

        contactList.setContainerDataSource(new BeanItemContainer<>(Contact.class));
        contactList.setColumnOrder("firstName", "lastName", "email");
        contactList.removeColumn("id");
        contactList.removeColumn("birthDate");
        contactList.removeColumn("phone");
        contactList.setSelectionMode(Grid.SelectionMode.SINGLE);
        contactList.setFrozenColumnCount(3);
        contactList.addSelectionListener(e
                -> contactForm.edit((Contact) contactList.getSelectedRow()));
        refreshContacts();

        HorizontalLayout actions = new HorizontalLayout(filter, newContact);
        actions.setWidth("100%");
        filter.setWidth("100%");
        actions.setExpandRatio(filter, 1);

        VerticalLayout left = new VerticalLayout(actions, contactList);
        left.setSizeFull();
        contactList.setSizeFull();
        left.setExpandRatio(contactList, 1);

        return left;
    }

    @Override
    public void buttonClick(ClickEvent event) {
```



Dienstleister

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