Vaadin Cheat Sheet

Flow (Java) — Vaadin 24 LTS / 25 — vaadin.com/docs/latest

CREATE & RUN

Scaffold at **start.vaadin.com**, Spring Initializr (+ **Vaadin**), or:

```
mvn -B archetype:generate \
 -DarchetypeGroupId=com.vaadin \
 -DarchetypeArtifactId=vaadin-archetype-application \
 -DarchetypeVersion=LATEST
mvn spring-boot:run          # dev mode + Copilot
mvn -Pproduction package     # optimised build
```

A VIEW

```
@Route(value = "hello", layout = MainLayout.class)
@PageTitle("Hello")
public class HelloView extends VerticalLayout {
  public HelloView() {
    TextField name = new TextField("Name");
    add(name, new Button("Hi", e ->
      Notification.show("Hi " + name.getValue())));
  }
}
```

LAYOUTS

```
var l = new VerticalLayout();   // flex column
l.setSpacing(true); l.setPadding(true);
l.setAlignItems(Alignment.CENTER);
c.setWidthFull(); c.setFlexGrow(1, child);
new HorizontalLayout(a, b);     // flex row
FormLayout f = new FormLayout();
f.setResponsiveSteps(
  new ResponsiveStep("0", 1),
  new ResponsiveStep("30em", 2));
new SplitLayout(left, right);
```

AppLayout: `addToNavbar(new DrawerToggle(), title); addToDrawer(new SideNav());`

INPUT FIELDS & HASVALUE

`TextField TextArea EmailField PasswordField NumberField IntegerField Checkbox CheckboxGroup RadioButtonGroup Select ComboBox MultiSelectComboBox DatePicker TimePicker Upload`

```
field.setValue(v); field.getValue();
field.addValueChangeListener(e -> {
  e.getValue(); e.isFromClient(); });
field.setValueChangeMode(ValueChangeMode.LAZY);
combo.setItems(query -> svc.fetch(...),   // lazy
               query -> svc.count());
field.setHelperText(..); field.setErrorMessage(..);
field.setRequiredIndicatorVisible(true);
```

BINDER

```
Binder<Person> b = new BeanValidationBinder<>(Person.class);
b.forField(ageField)
 .withConverter(new StringToIntegerConverter("num"))
 .withValidator(a -> a >= 18, "adults only")
 .bind(Person::getAge, Person::setAge);
b.bindInstanceFields(this);   // by name/@PropertyId
b.readBean(person);           // buffered: load
if (b.writeBeanIfValid(person)) save(person);
b.setBean(person);            // unbuffered: live
b.addStatusChangeListener(e -> save.setEnabled(b.isValid()));
```

GRID

```
Grid<Person> g = new Grid<>(Person.class);
g.setColumns("firstName", "email");
g.addColumn(Person::getName).setHeader("Name")
 .setKey("name").setSortProperty("name").setAutoWidth(true);
g.addComponentColumn(p -> new Button("Edit"));
g.setItems(list);                  // in-memory
g.setItems(q -> svc.fetch(q.getOffset(), q.getLimit()).stream(),
           q -> svc.count());      // lazy
g.setSelectionMode(SelectionMode.MULTI);
g.asSingleSelect().addValueChangeListener(e -> ...);
g.getDataProvider().refreshAll();
```

Inline edit: **GridPro** (commercial) or `grid.getEditor()` + Binder. `TreeGrid` + `addHierarchyColumn`.

ROUTING

```
@Route("order/:id")            // template param
class OrderView extends Div implements
    HasUrlParameter<Long>, BeforeEnterObserver {
  public void setParameter(BeforeEvent e, Long id){...}
  public void beforeEnter(BeforeEnterEvent e){
    e.getRouteParameters().get("id");
    e.rerouteTo(LoginView.class);   // or forwardTo
  }
}
UI.getCurrent().navigate(OrderView.class);
new RouterLink("Orders", OrderListView.class);
```

Lifecycle: BeforeLeave (`postpone()`) → BeforeEnter (reroute/forward) → AfterNavigation.

`@Layout` auto-parent · `@ParentLayout` nest · `@PageTitle/HasDynamicTitle` · `HasErrorParameter<NotFoundException>`.

OVERLAYS & ACTIONS

```
Notification.show("Saved", 3000, Position.BOTTOM_START);
Dialog d = new Dialog(); d.setHeaderTitle("Edit");
d.add(form); d.getFooter().add(cancel, save); d.open();
new ConfirmDialog("Delete?", "", "Delete",
  e -> delete(), "Cancel", e -> {}).open();
btn.addClickShortcut(Key.KEY_S, KeyModifier.CONTROL);
comp.setTooltipText("...");   // Popover for rich content
new TabSheet(); new Accordion(); new ContextMenu(grid);
```

SERVER PUSH

```
@Push                          // on AppShellConfigurator
UI ui = UI.getCurrent();       // capture on request thread
executor.execute(ui, () -> {
  var result = slowWork();
  ui.access(() -> {            // acquires session lock
    grid.setItems(result);     // auto-push (or ui.push())
  });
});
// no-push fallback:
ui.setPollInterval(2000);
ui.addPollListener(e -> refresh());
```

Unregister listeners in `onDetach` to avoid `UIDetachedException`.

ELEMENT API & JS

```
getElement().setProperty("value", "x");
getElement().getStyle().set("color", "red");
getElement().executeJs("return $0.offsetWidth", el)
            .then(Integer.class, w -> ...);
@ClientCallable void save(String s) { ... }
// this.$server.save(v) on the client

@Tag("my-widget")
@NpmPackage(value = "my-widget", version = "1.0.0")
@JsModule("my-widget/index.js")
class MyWidget extends Component { }
```

THEMING (LUMO)

```
@Theme("my-app")               // src/main/frontend/themes/my-app/
@Theme(themeClass = Lumo.class, variant = Lumo.DARK)
btn.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
grid.addThemeVariants(GridVariant.LUMO_COMPACT);
layout.addClassNames(LumoUtility.Padding.MEDIUM,
  LumoUtility.Gap.SMALL, LumoUtility.Display.FLEX);
```

```
/* styles.css */
html { --lumo-primary-color:#2f6f4f;
       --lumo-border-radius-m:4px; }
vaadin-button::part(label){text-transform:uppercase;}
```

SPRING SCOPES

```
@SpringComponent @UIScope       // per browser tab
@SpringComponent @VaadinSessionScope
@SpringComponent @RouteScope    // + @RouteScopeOwner(V.class)
// constructor injection into @Route views works out of the box
```

`application.properties`: `vaadin.allowed-packages`, `vaadin.launch-browser`, `vaadin.productionMode`, `vaadin.pnpm.enable`.

SECURITY

```
@EnableWebSecurity
class SecurityConfig extends VaadinWebSecurity {
  protected void configure(HttpSecurity http) throws Exception {
    super.configure(http);
    setLoginView(http, LoginView.class);
  }
}
```

Per view (denied by default):

`@AnonymousAllowed`	anyone
`@PermitAll`	any authenticated user
`@RolesAllowed("ADMIN")`	listed roles
`@DenyAll`	no one (default)

`authenticationContext.logout()`. Same annotations secure Hilla endpoints. CSRF is automatic.

HILLA (REACT) IN ONE GLANCE

```
@BrowserCallable @AnonymousAllowed
class PersonEndpoint {
  public List<Person> list() { return repo.findAll(); }
}
// src/main/frontend/views/people.tsx
import { PersonEndpoint } from 'Frontend/generated/endpoints';
const people = await PersonEndpoint.list();  // typed
// <Grid> from '@vaadin/react-components'
// reactive: Flux<T> -> for await (const x of Endpoint.stream())
```

TESTING & BUILD

```
// browserless UI unit test (not commercial)
@ViewPackages(packages = "com.example.views")
class T extends UIUnitTest {
  @Test void t() {
    var v = navigate(CustomerView.class);
    $(Button.class).withText("Save").first().click();
    assertEquals("Saved", $(Notification.class).last().getText());
  }
}
```

E2E: TestBench (commercial) — `$(GridElement.class).id("g")`, `compareScreen(..)`. Also Playwright / Selenium.

Deploy: Spring Boot jar / WAR / Docker / native. Cluster → **sticky sessions** + WebSocket pass-through.

VAADIN 7 → FLOW MAP

`Table`	Grid / TreeGrid
`Property / Item / Container`	Binder + DataProvider
`Navigator / View`	@Route / RouterLayout
`com.vaadin.ui.*`	com.vaadin.flow.component.*
`GWT widget + connector`	Web Component + Element API
`Sass theme`	Lumo CSS custom properties + ::part()
`Fusion`	Hilla

Written and verified against vaadin.com/docs/latest. Generated with AI assistance — verify against the official docs before relying on it in production.