

VAADIN TRAINING

Introduction to Vaadin 10



Overview

Vaadin is a modern User Interface platform for Java and JavaScript. It is completely back-end agnostic, modular, and comes with a good-looking extendable theme out of the box. With Vaadin, you can build anything from small mobile-first web apps to large-scale enterprise portals, all with the familiar tools you already use.

More full-featured and modular than ever before

With Vaadin Platform 10, the client-side part of the framework has been completely re-written in favour of Web Components. Vaadin is still the server-side framework it always was, but now it's also much more than that. The all-new Flow library makes manipulating the DOM from the server-side a breeze, and dropping GWT means no more lengthy compiles or hard to use pseudo-javascript API.

On the Server

The traditional Vaadin hasn't gone anywhere. The fastest and easiest way to create full-featured web applications is still to write them in 100% server-side Java. Using the Component API, you create components and layouts, and bind them to events and data, all inside a familiar JVM with all the tools that it brings, such as direct access to Spring or JavaEE. There is no need to worry about the browser, javascript, or communication between the client and the server, as the platform will handle this for you transparently.

```
JAVA // Sample server-side code
HorizontalLayout buttonLayout = new HorizontalLayout();

Button saveButton = new Button("Save");
saveButton.addClickListener(event -> save());

Button cancelButton = new Button("Cancel");
cancelButton.addClickListener(event -> cancel());

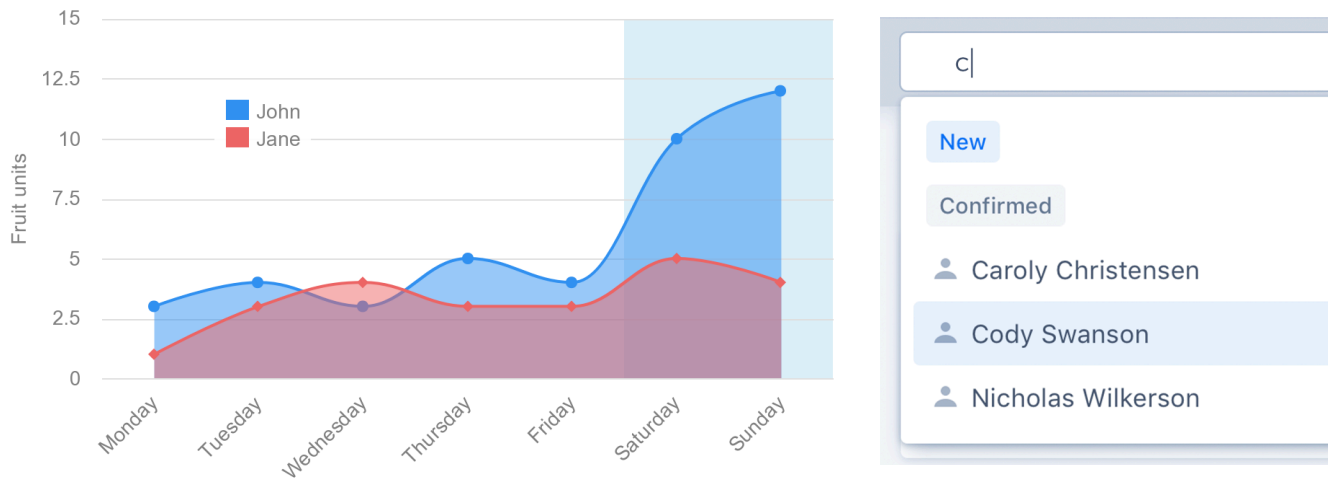
buttonLayout.add(saveButton, cancelButton);
```







Vaadin 10 also adds powerful navigation and HTML template features to simplify development even more. In addition, you can use the Flow API to customize your DOM from the server side, wherever you need to.

On the Client

The Vaadin Platform fully embraces the growing needs of JavaScript developers with the launch of Vaadin Elements. Vaadin Elements are Polymer-based Web Components, fully usable in standalone HTML/JavaScript applications. The collection includes all the basic elements that you might need, as well as more complicated ones such as various Charts, the Grid, and the ComboBox. Each is fully themed and documented, ready for use.

EXAMPLES OF VAADIN ELEMENTS: CHARTS, COMBO-BOX AND GRID



	Email	Name
<input type="checkbox"/>	 kaycee.thompson@price.name	Louisa Brady
<input checked="" type="checkbox"/>	 steuber_nathanael@hotmail.com	Luke Waters
<input type="checkbox"/>	 sipes_connor@gmail.com	Francisco Saunders
<input checked="" type="checkbox"/>	 cullen_lang@elinor.io	Albert Walsh
<input checked="" type="checkbox"/>	 jakubowski_schuyler@yahoo.com	Jeff Bryant
<input type="checkbox"/>	 schmeler.delbert@maci.me	Derrick Stevenson

In practice, Web Components mean that there is no need for plugins on the client side. Everything is standard-compliant HTML and JavaScript. For older browsers, we use the Polymer library by Google, that provides compatibility until browsers are updated.

In between the Server and the Client

The biggest change from previous Vaadin versions is the new API between the server and the client. From the server, it's now extremely easy to access client-side structures and add DOM elements or event listeners. On the client, you can let any event listener propagate down to the server, or handle it in place. You can also share data both ways, if you want.

```

JAVA      // Server-side use of the Flow API:
public MainView() {

    add(new Header());
    add(new ContentArea());

    final Div footerDiv = new Div();
    footerDiv.addClassName("footer");

    final Element footerElem = footerDiv.getElement();
    getElement().appendChild(footerElem);
    footerElem.setAttribute("height", "20px");
    footerElem.addEventListener("click", this::handleFooterClick);
}

// Example of client-initiated events:
// Client side template code:
<div id="div" on-click="updateStatus">[[status]]</div>

// JavaScript event handler, run immediately
updateStatus() {
    this.status = "Connecting to server...";
}

// Java event handler on the server, run asynchronously
@EventHandler
private void updateStatus() {
    getModel().setStatus("Confirmed on the server");
}

```

Bringing it all together

A typical Vaadin 10 application sits in the business world. It has a robust backend based on e.g. Spring or JavaEE, with a well-defined border to the UI code running in the same environment (or another!). The UI comprises of Components, arranged in layouts made with the Java component API, the Flow HTML DOM API, and HTML templates. Powerful data component APIs bind the UI components to your data and backend, providing lazy loading (paging), conversion and validation features. On top of it all lies the fully customizable themes, providing the unique look and feel of your application.

See our success stories here: <https://vaadin.com/success-stories>