

SimpleWebConf 2021

"Cooking Club" Workshop

June 2021 - Vaadin 20

Agenda

Build your own PWA with Lit, Spring Boot, and Vaadin Fusion

- Introduction
- Create a empty project: *start.vaadin.com*
- Presentation of the project
- Let's code

What is Vaadin Fusion?

Vaadin Fusion

Vaadin Fusion

Vaadin Fusion is a full-stack framework for developing reactive client-side web apps with a **Java** backend.

Views are based on **web component**, built with **LitElement** and **TypeScript**.

The framework generates types from the Java server endpoints and data classes, giving you secure, **typesafe** and straightforward form binding and server endpoint access.



What Fusion is then

Opinionated full stack application development framework for Java backend and TypeScript frontend purposed for single page web applications.

Vaadin **web component** library with Lumo as design system

Vaadin **maven** / **gradle** plugin for doing the build

This is important part of the opinionated stack with selected tooling, npm, webpack, etc.
generate client side code based on Endpoints

Lit to render the views and component

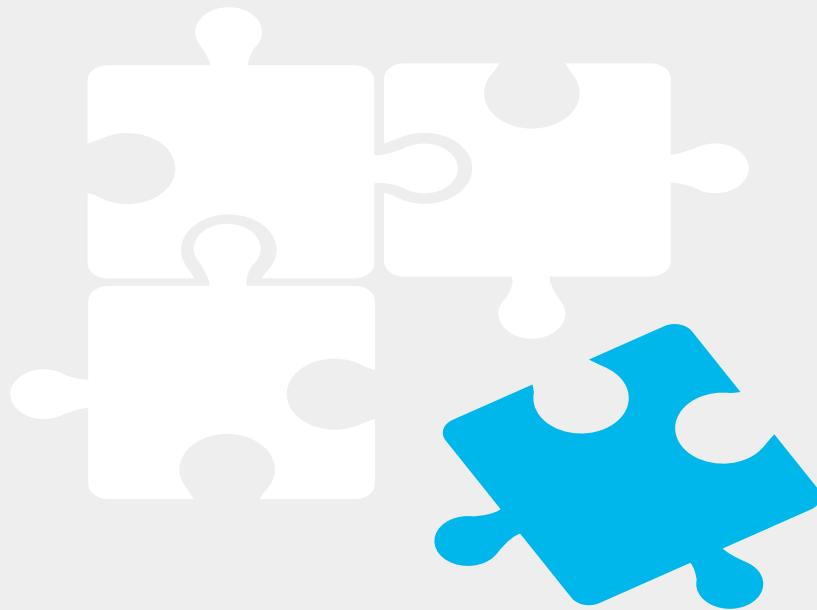
MobX for state management












Core Java backend w/ **Spring Boot**

Vaadin Components

Components

Web Components since 2015



		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
<input type="checkbox"/>		lehner.raina@hotmail.com	Hester Townsend
<input type="checkbox"/>		maeve.powlowski@louie.info	Seth Fitzgerald
<input type="checkbox"/>		braulio.thiel@yahoo.com	Jack Walsh
<input type="checkbox"/>		carmel_ward@mayert.com	Mildred Morgan
<input type="checkbox"/>		buster.maggio@hotmail.com	Francis Hoffman

Grid

Combo box

List item|
X
v

☒ List item



List item

List item

List item

Combobox

Due •

 Date 

17 24 25 26 27 28 29 30

2014

•

2015

•

2016

•

2017

•

2018


•


2019


June 2017




	Mon	Tue	Wed	Thu	Fri	Sat	Sun
18	1	2	3	4	5	6	7
19	8	9	10	11	12	13	14
20	15	16	17	18	19	20	21
21	22	23	24	25	26	27	28
22	29	30	31				

Date Picker

Select files...  Drop files here

✓ test-document.doc 

IMG_5272.JPG 
19.7 MB: 60% (remaining time: 00:12:34)

 website-mockup.pdf  
An error occurred

Upload

Create a view in Fusion = Custom Element

With LitElement

LitElement

A web component library

Created and maintained by Google

Is build to support reactive programming metaphor. This metaphor is also preferred way of writing apps with Fusion as well.

Version 2.5 in Vaadin 20

Version 3.x in Vaadin 21 (Currently in alpha)

LitElement supports development in both TypeScript and plain JavaScript. With Fusion the preference is to use TypeScript.

LitElement

The **@customElement** decorator assigns a unique element name and registers it to the browser's custom element registry.

The **render()** method is responsible for rendering and re-rendering a component template.

@property and **@state** (= **@internalProperty**) attributes are listened and any change will trigger an asynchronous re-render of the component to fill the template with the new data.

It creates **reactive** web components.

```
import { LitElement, html, css, customElement, state } from
'lit-element';

@customElement('about-view')
export class AboutView extends LitElement {
  @state()
  private name:string = "World";

  static get styles() {
    return css`
      :host {
        display: block;
        padding: var(--lumo-space-m) var(--lumo-space-l);
      }
    `;
  }

  render() {
    return html`
      Hello ${this.name}!
    `;
  }
}
```

MobX - Application State management

MobX

MobX manages the frontend state in your app.

Our views are automatically using **(Lit)MobX** so any change to those **MobX** observables is automatically reflected in the view.

In that case, every change of **myState.quote** will be reflected in a view that is using this state.

```
import { makeAutoObservable } from 'mobx';

class MyState {
  quote = `Anything that can be derived from the application
state,
  should be. Automatically. - MobX documentation`;

  constructor() {
    makeAutoObservable(this);
  }

  setQuote(quote: string) {
    this.quote = quote;
  }
}

export const myState = new MyState();
```

Spring Boot

Spring Boot

Vaadin Fusion requires Spring Boot in the backend.

It's an open source Java-based framework that will help us to:

- Query our database
- Secure our application
- Generate the endpoints

Getting started

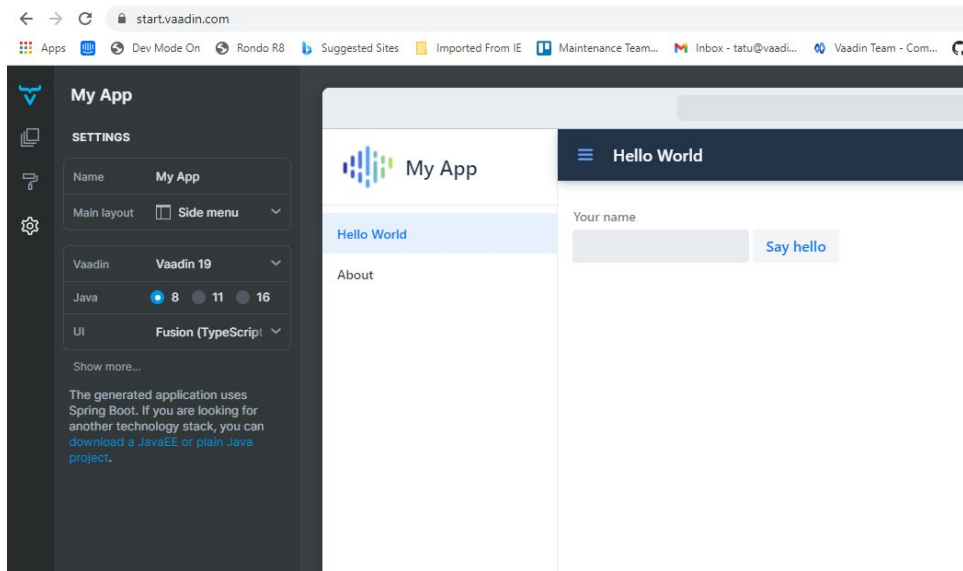
With a new project → How I created the skeleton of the workshop?

Download

Download a starter start.vaadin.com

Select the latest Vaadin version, e.g. 20, Java version and UI = Fusion (TypeScript) project from settings cog.

All the preset views are available, which is handy for exploring and demonstration.



Get the application

If you already have a git client installed, you can clone this repository:

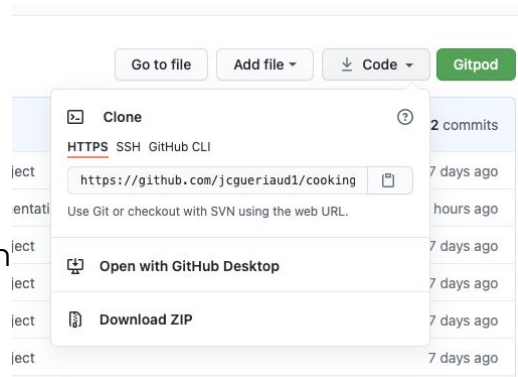
git clone <https://github.com/jcgueriaud1/cooking-club>

Then follow the instructions in the README.md.

You can open the instructions in your browser:

<https://github.com/jcgueriaud1/cooking-club>

If you don't have git installed, you can download the zip of the application or use the one in the email.



Workshop

You can ask questions using the Q&A feature in Google Meet. You can also upvote existing questions.

If you're stuck and you need specific help you can join the Breakout Room: "Problem Solving".

At the beginning of each step, you can get a clean state of the application by checking out the branch from the previous step.

Demo of the final state of the application

Time to code

Step 1

Event list

- Create a new view
- Update the endpoint
- Use CSS
- Create a list of events displayed as "card"

Step 2

Event Store

- Create a new store
- Use the store in the view

*If the step 1 is not completed
Branch:
1__event-view*

Step 3

Subscription

- Subscription form
- Update the backend
- Communication between two components

If the step 2 is not completed
Branch:
2__event-store

Step 4

Security

- Secure the frontend and backend
- Login page

If the step 3 is not completed
Branch:
3__subscribe-event

Step 5

Offline

- Connection status
- Caching

If the step 4 is not completed
Branch:
4__secure-app

https://github.com/jcgueriaud1/cooking-club/blob/main/docs/5__offline.md

Step 6

Deploy to Heroku

- Package the application for production
- Deploy in Heroku

*If the step 5 is not completed
Branch:
5__offline*

Any questions?

Thank you for your attention
And your work

You can find us on Discord:
<https://discord.gg/vaadin>
#fusion Channel

Twitter:
@GueriaudJc
@marcushellberg



Fusion resources

Fusion documentation:

<https://vaadin.com/docs/v20/fusion/overview>

Our components:

<https://vaadin.com/docs/v20/ds/components>

Lit website:

<https://lit.dev/>

MobX:

<https://mobx.js.org/README.html>