Übung 05
Olypmics mit `Vue / Components`
SWP-W
5abHWII

January 11, 2023

Übungsleiter: Albert Greinöcker

Ziel der Übung:

- Erstellen und Verwenden von eigenen Komponenten in `vue.js`
- Installation und Verwenden von bestehenden Komponenten
- Erstellen von interaktiven Grafiken mit `Plotly`
- Einbinden einer Data Table mit `vue.js`
- Einbinden von `bootstrap` in `vue.js`

1 Allgemeine Aufgabenstellung

Es soll versucht werden mittels `vue.js` - Komponenten ein Dashboard zu erstellen basierend auf historischen Daten zu den olympischen Spielen. Es besteht auch die Möglichkeit ein Dashboard mit anderen Daten zu erstellen.

2 Server

2.1 Datenbank

Es existiert bereits eine `SQLite`-Datenbank[1] (liegt bei den git-Beispielen unter `flask/olympia/olympics.db`). Es sind alle Ergebnisse, die von Teilnehmern an Olympischen Spielen in den letzten 120 Jahren erzielt wurden, aufgelistet.

[1] Quelle: https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results

2.2 Flask

Es existiert bereits ein Server (unter den vue.js-Beispielen unter `flask/olympia` zu finden) so wie auch in der Anwendung mit dem Zettelkatalog. Hier ist auch schon eine Abbildung auf die Olympics-Datenbank mittels `SQLAlchemy` umgesetzt.

Folgende Funktionen sind schon vorhanden:

- `http://127.0.0.1:5000/medals/AUT` liefert die Daten für AUT in folgender Form:

```
[[ "Bronze", 6553],
[ "Gold", 6329 ],
[ "Silver", 6269 ]]
```

- `http://127.0.0.1:5000/medals2/AUT` macht das gleiche aber schon für `Plotly` vorbereitet, so braucht man diese Inhalte nur der `Plotly`-Komponente in der Darstellung zuweisen ohne dort etwas ändern zu müssen:

```
[{"x": ["Bronze", "Gold", "Silver"], "y": [51, 31, 84], "type": "bar"}]
```

- `http://127.0.0.1:5000/events`: Alle Eventbezeichnungen die es gibt (zur Verwendung von weiteren Abfragen). Die Daten kommen in folgender Form:

```
[{"event": "Aeronautics Mixed Aeronautics"},
{"event": "Alpine Skiing Men's Combined"},
{"event": "Alpine Skiing Men's Downhill"},
...
]
```

- `http://127.0.0.1:5000/count_by_sex`: Anzahl der Medaillen, nach Geschlecht. Hier wird gezeigt wie man Abfragen ohne SQLAlchemy machen kann falls das bei komplexeren Aufgaben beginnt zu nerven. Das Ergebnis sieht so aus:

```
[["F", "Bronze", 1611],
["F", "Gold", 1533],
["F", "Silver", 1581],
["M", "Bronze", 4942],
["M", "Gold", 4796],
["M", "Silver", 4688]]
```

2.3 Client

Zusätzlich zu allem, was wir bereits über `vue.js` wissen sollen jetzt bestehende Komponenten (`datatable`, `plotly`, `bootstrap`) installiert und verwendet und auch eigene Komponenten erstellt werden.

2.4 Installation der Client-Packages

Bibliotheken werden jetzt nicht mehr in HTML eingebunden sondern direkt zu Anwendung installiert:

`npm install --save datatables.net-vue3`[2]

[2] `--save` veranlasst eine Speicherung in `package.json` und wird somit bei einem generellen `npm install` mitinstalliert

```
npm install --save datatables.net-bs5
npm install --save bootstrap
npm install --save vue3-plotly
npm install uuid (wird benötigt für datatables)
```

danach am Besten noch `npm install` ausführen um alle Abhängigkeiten zu installieren.

2.5 Verwendung der einzelnen vorhandenen Packages

2.5.1 Datatables

notwendige imports:

```
import DataTable from 'datatables.net-vue3'
import DataTablesLib from 'datatables.net-bs5';
```

hier ein Beispiel zur Datatable mit vue.js:

```
<DataTable class="table table-hover table-striped" width="100%" :data="medals">
    <thead>
        <tr>
            <th v-for="c in columns">{{ c }}</th>
        </tr>
    </thead>
    <tfoot>
    </tfoot>
</DataTable>
```

- `:data` sollte ein verschachteltes Array sein dass die Zeilen beinhaltet (so wie sie der Service-Aufruf `medals` zurückgibt)
- `columns` ist ein Array dass die Spaltennamen beinhaltet

Weitere Infos (Code auch von dort kopieren): `https://datatables.net/blog/2022-06-22-vue#Source`

2.5.2 Plotly

Das ist der Grundaufbau der Plotly-Komponente:

```
<VuePlotly :data="p2" :layout="layout"></VuePlotly>
```

- `:data` Hat einen Aufbau wie beim Service-Aufruf von `medals2`.
- `:layout`: Hier gibt es sehr viele Möglichkeiten des Stylings: `https://plotly.com/javascript/reference/#layout`

Weitere Infos (Code auch von dort kopieren): `https://socket.dev/npm/package/vue3-plotly`

2.6 Bootstrap

Auch wenn die Installation jetzt anders verlaufen ist ändert sich nichts an der bekannten Verwendung von Bootstrap

2.6.1 Genereller Aufbau

Als Vorschlag könnte man das Grid-System verwenden:

```
<div class="container">
    <div class="row">
        <div class="col-sm">
            <!-- hier kommen die einzelnen Charts hin -->
        </div>
    </div>
</div>
```

2.6.2 Weiteres Styling

Nett wäre natürlich wenn weitere Styling-Möglichkeiten von Bootstrap genutzt werden: `https://getbootstrap.com/docs/4.0/getting-started/introduction/`

2.7 Eigene Komponenten

Hier gibt es ein Beispiel in `vue_js`-Projekt auf `github`: `03_sfc/02_components` (wird im Unterricht besprochen)

2.8 Allgemeines noch

Generell nicht vergessen, die Komponenten zu registrieren:

```
export default {
    components:
    {
        DataTable,
        VuePlotly,
        ...
    },
    ...
}
```

3 Aufgabenstellung

Mit den oben besprochenen Komponenten soll eine Art Dashboard für die Olympia-Daten erstellt werden. Folgendes soll zumindest enthalten sein:

a. 2 Auswahlfelder mit den Ergebnissen der Services events und regions (beinhaltet noc - also den Ländercode und den Bezeichner) sollen erzeugt werden. Diese soll für die Auswahl der angezeigten Daten verwendet werden (Wenn z.B. eine bestimmte Region ausgewählt wird, bezieht sich die Statistik auf diese Auswahl)

b. Anzeige der Medaillen (Gold, Silber, Bronze) für die Regionen z.B. als Barplot (Plotly) sowie als Tabelle (Data Table)

c. Darstellung der Medaillen nach Geschlecht als Barplot

d. 3 weitere Statistiken mit zumindest einer anderen grafischen Darstellung als ein Barplot. Andere Charts gibt es jede Menge: `https://plotly.com/javascript/basic-charts/`. Das hat natürlich auch zur Folge dass man den Flask Server mit neuen Services anreichern muss.

e. Zumindest eine Darstellung soll als eigene Komponente zwecks Wiederverwendung implementiert werden.

Viel Spaß!