Hochschule RheinMain
H. Arega, W. Weitz

Studiengang Medieninformatik
Sommersemester 2024

Webbasierte Anwendungen

Web-Ueb-07 (05.06.2024)

Bei diesem Übungsblatt geht es darum, wie man externe REST-Dienste in die eigene Anwendung integrieren. Wir werden unsere API nach Bedarf in den folgenden Wochen ergänzen.

Als Vorbereitung für unseren Frontend-Entwicklungs-Abschnitt wollen wir einige unserer Anwendungdaten- und Funktionen als Webdienst nach außen zur Verfügung stellen. Dabei verwenden wir nach Möglichkeit Java Records, um Data-Transfer-Objects (DTOs) für die Außenschnittstelle umzusetzen. Auf diese Weise können wir die Datensätze "geschickt" ausgestalten (was bei einem 1:1-Durchreichen der `@Entity`-Objekte nicht immer der Fall wäre) und grenzen die interne Datenhaltung von der externen Schnittstelle ab. Entity-Objekt selbst dürfen **nicht** als Parameter oder Rückgabewerte unserer REST-Handlermethoden verwendet werden (technisch geht das, aber es ist moralisch nicht zu verantworten).

Sie können z.B. die schon früher erwähnte VS-Code-Extension *Thunder Client* hier gut zum REST-Testen verwenden (Extension-ID `rangav.vscode-thunder-client`).

Aufgabe 1 (API für Orte)

Bitte legen Sie in Ihrem Projekt ein neues Package `de.hsrm.mi.web.projekt.api` an und darin in einem Unterpackage `ort` einen REST-Controller `OrtAPIController`, der folgende Endpunkte bedient:

GET /api/ort liefert eine Liste von JSON-Objekten, welche je `Ort`-Objekt ein DTO mit dessen `id`, `name`, `geoBreite` und `geoBreite` enthält (nicht mehr), wie in diesem Beispiel:

```
[
  { "id": 3, "name": "Taucha", "geoBreite": 51.3799905, "geoLaenge": 12.4936336 },
  { "id": 2, "name": "Vollradisroda", "geoBreite": 50.9153714, "geoLaenge": 11.4964286 },
  { "id": 1, "name": "Wiesbaden", "geoBreite": 50.0636, "geoLaenge": 8.2414 }
]
```

Legen Sie dazu bitte (ebenfalls im Package `de.hsrm.mi.web.projekt.api.ort` ein entsprechendes Java-Record `OrtDTO` an, und fügen Sie ihm zwei **statische** Hilfsmethoden hinzu, welche die Attribute eines `Ort`-Objekts auf ein neues `OrtDTO` abzubilden und umgekehrt:

- `OrtDTO fromOrt(Ort o)` erzeugt ein `OrtDTO` aus einem übergebenen `Ort`-Objekt `o`, und entsprechend ein
- `Ort toOrt(OrtDTO dto)`, das aus einem `OrtDTO` ein entsprechend befülltes `Ort`-Objekt erzeugt.

Diese Methoden sind nützlich bei der Implementierung der folgenden Handler-Methoden.

GET /api/ort/{id} liefert ein DTO für den Ort mit der übergebenen ID `id`. Die Struktur des gelieferten DTO ist wie oben. Wenn eine ID übergeben wird, zu der kein Ort in der Datenbank vorhanden ist, soll die Antwort mit dem HTTP-Statuscode 404 (Not Found) zurückkommen (HTTP-Statuscode – nicht "Inhalt").

Hinweis: Spring hat einen vordefinierten Exception-Typ `ResponseStatusException`, dem Sie als Parameter einen der vordefinierten `HttpStatus`-Aufzählung mitgeben können. Herumwerfen mit solch einer Exception erlaubt es, auf einfache Weise einen laufenden Aufruf mit einem gewünschten HTTP-Fehler zu beenden. Im Beispiel genügt also z.B. ein

`throw new ResponseStatusException(HttpStatus.NOT_FOUND)`, um einen HTTP "Not Found"-Fehler zurück zu geben.

Sie können dies z.B. gut mit a) einer gültigen und b) einer ungültigen ID mit dem ThunderClient aus VSCode heraus testen.

Wenn Sie Ihre Spring-Anwendung mit aktivierten DevTools laufen lassen, enthalten Sie im Fehlerfall von Ihrem REEST-Endpunkt ein JSON mit u.a. einem Exception-Stacktrace zurück. In einer fertigen Anwendung will man solche Informationen natürlich nicht nach außen geben. Keine Sorge, im "Production Mode" als fertig paketierte Anwendung oder auch mit abeschalteten DevTools werden solche Informationen nicht eingeschlossen.

Akzeptanzkriterien für diesen Projektstand:

- HTTP GET auf URI-Pfad `/api/ort` liefert eine JSON-Liste mit allen `OrtDTO`-Objekten in der oben beschriebenen Form.
- HTTP GET auf URI-Pfad `/api/ort/`*`id`* liefert eine einem `OrtDTO`-entsprechenden JSON-Struktur mit der ID *id* und den Daten des zugehörigen `Orts` in der oben gezeigten Form.
- Bei Abruf von `/api/ort/`*`id`* mit einer ungültigen `id` erscheint wird eine Antwort mit HTTP-Statuscode 404 (Not Found) zurück gegeben.
- Durch die Nutzung der REST-API wird die Datenbank-Zustand nicht verändert / beschädigt.

Hinweis: Die folgenden Aufgaben des heutigen Blattes dient dem Einstieg in die Vue-Entwicklung. (Nur) diese Aufgaben sind individuell und **außerhalb** des Spring-Projekts anzulegen und zu entwickeln, damit man frei ausprobieren, bei Problemen wegwerfen und neu anlegen kann, ohne das Hauptprojekt zu beeinträchtigen. Sie sind nicht abzugeben. Ab nächster Woche werden wir mit der gewonnenen Erfahrung das Vue-UI **im** bestehenden (Spring-)Hauptprojekt entwickeln.

Aufgabe 2 (Vue-Spielprojekt anlegen)

Vue3/Vite-Spielprojekt in Shell anlegen

Legen Sie bitte mit `npm init vue@latest vue3-intro` ein Vue-**Beispielprojekt** `vue3-intro` **außerhalb (!)** Ihres Spring-Projekts (und dessen Repository) an. Wir nutzen diese Runde zum Kennenlernen der neuen Umgebung und der Tools, und wenn etwas schief geht, wollen wir es ohne Beschädigung des Spring-Projekts auch wieder loswerden.

Bitte wählen Sie bei `npm init` (nur) **folgende Features** aus (mit Tab-Taste auf "yes" gehen, mit Return-Taste geht's weiter, bei Fehler abbrechen und neu anfangen) : "TypeScript" und "ESLint for code quality" (sonst nichts). Vergessen Sie nicht, im angelegten Projektverzeichnis mit `npm install` die nötigen Pakete gemäß `package.json` herunterladen zu lassen.

Bitte schauen Sie ruhig mal in `package.json` hinein. Neben einer Liste der Abhängigkeiten (*dependencies*, benötigte JavaScript-Module) finden Sie im Abschnitt "**scripts**" die Subkommandos, die Sie mit "`npm run ...`" starten können (z.B. `npm run dev` zum Starten des Entwicklungs-Webservers für das Vue-Projekt).

Vue-Extension in VSCode installieren, Anwendung starten

Installieren Sie in Ihrem VSCode nun bitte zusätzlich die Extension mit der ID `vue.volar` ("Vue Official - Language Support for Vue" – *nicht* das TypeScript-Vue-Plugin dazu etc., das ist überholt).

Vue-Projektstruktur

Verschaffen Sie sich bitte mit VSCode einen Überblick über die **Projektstruktur** des eben angelegten `vue3-intro`. Starten Sie dann bitte den lokalen Entwicklungsserver (`npm run dev`) und bebrowsen Sie ihn mit dem zuvor mit dem Vue-Plugin "Vue.js devtools" ausgestatteten Browser, vorzugsweise Chrome (`http://localhost:5173`). Probieren Sie bitte das Vue-Devtools-Browser-Plugin anhand der generierten Beispielanwendung aus (im Browser Entwicklertools öffnen und "Vue"-Tab wählen).

Wieviele Dateien wurden beim Anlegen dieses sehr kleinen Projekts, das praktisch noch nichts tut, angelegt? Mit der Linux-Kommandozeile

```
find . | wc --lines
```

können Sie sich die Anzahl der Ordner und Dateien in und unterhalb des aktuellen Verzeichnisses (".") anzeigen lassen, und so erhalten Sie den belegten Plattenplatz:

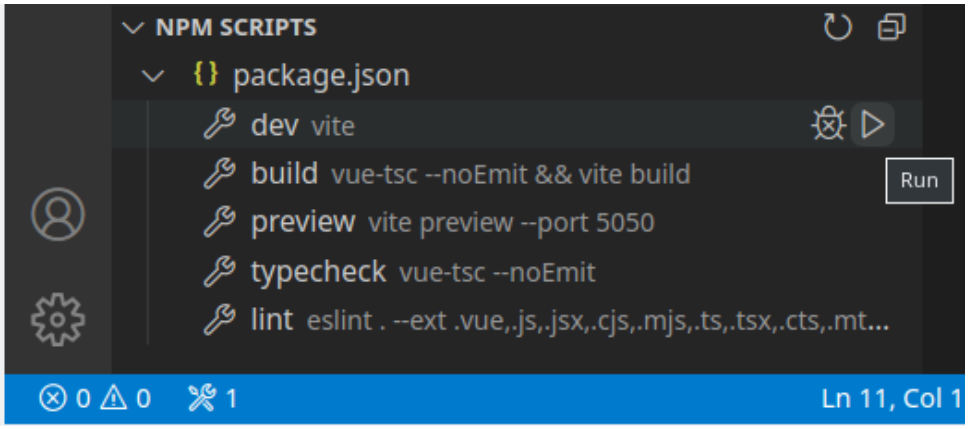
```
du -sh .
```

Führen Sie die Kommandos bitte im Basisverzeichnis Ihres Vue-Projekts aus, setzen Sie sich und staunen Sie. Viele MB, bevor "irgendwas passiert ist", sind ja schon einmal nicht schlecht. Dadurch wird aber vielleicht auch klarer, warum es wichtig ist, mit in den Build-Workflow von Vue integrierten Tools nur die *wirklich* zur Laufzeit des Frontends benötigten Dateien zusammensuchen zu lassen – das alles muss ja beim Laden der Seite zuerst in den Browser übertragen werden, und hunderte MB für "fast nichts" wären je nach Mobiltarif langsam, teuer oder beides.

Projekt bearbeiten

Laden Sie das Projekt nun bitte in VSCode und nehmen Sie kleine Änderungen an der Vue-Komponente `HelloWorld.vue` (im `src/components/`-Ordner) vor, z.B. durch Ausdünnen des `template`-Abschnitts oder Änderung eines Textschnipsels. Der lokale Entwicklungs-Server, den Sie auch direkt aus VSCode im "NPM

SCRIPTS"-Tab unter dem Projekt-Explorer starten können, müsste beim Abspeichern das Projekt neu bauen und der Browser automatisch aktualisiert werden – bitte überprüfen Sie das, indem Sie z.B. einen Ausgabetext im `<template>`-Abschnitt der `HelloWorld.vue` ändern. Behalten Sie bitte dabei den Browser im Auge – die Ausgabe sollte sich kurz nach Abspeichern der Änderung automatisch anpassen.



npm run build

Bauen Sie bitte zwischendurch auch einmal eine **deploybare Ausgabe** der Anwendung (Kommando `npm run build`, geht alternativ ebenfalls auch im VSCode unter "NPM SCRIPTS") und sehen Sie sich im `dist`-Ordner an, was der Modul-Bundler und seine Gehilfen aus Ihrem Projekt produziert haben. Schauen Sie mit `du` (s.o.) auch mal nach, wie umfangreich der Inhalt des `dist`-Ordners ist, und vergleichen Sie mit oben – `dist` das wäre annähernd das, was von Ihren Nutzern im Produktivbetrieb tatsächlich für Ihr Frontend in den Browser geladen wird.

Git

Das Vue-CLI hat beim Anlegen des Projekts zwar kein lokales Git-Repository initialisiert, aber schon einmal eine passende `.gitignore` vorbereitet, da es eine Katastrophe wäre, wenn alle der oben gezählten Dateien eingecheckt würden.

Bitte initialisieren Sie für das Projekt ein lokales Git-Repository und committen Sie den aktuellen Projektstand. Pushen Sie wie gesagt dieses Nebenprojekt **nicht** in das Projekt-Git-Repository (sollte auch nicht gehen, weil Sie die Anweisungen oben aufmerksam gelesen haben und dieses Spielprojekt **nicht** innerhalb Ihres Spring-Projekts angelegt haben). Wenn Sie möchten, können Sie sich auf unserem RhodeCode-Server in Ihrem `stud`-Ordner (wie gesagt **nicht** im Repository für das Semesterprojekt) ein Spiel-Ziel-Repository anlegen und das Spielprojekt dort hineinpushen.

Aufgrund der `.gitignore` sollte der Ordner `node_modules` mit den heruntergeladenen JS-Modulen **nicht** mitversioniert werden. Sie erinnern sich – viele MB Platz, und nach einem frischen Clonen des Repositories würden mit `npm install` (ohne Parameter, siehe oben) alle **Abhängigkeiten** aus der "Packliste" `package.json` automatisch **nachinstalliert**.

Dieser Schritt wäre nach dem initialen Clonen eines Vue-Projekts aus einem Repository also regelmäßig nötig (und falls nicht, hat vermutlich ein Nasenbär die vielen (s.o.) `node_modules`-Dateien doch mit in's Git committet und wird fortan weniger respektiert werden. Für solche Fälle gibt es das `git blame` Kommando.

Aufgabe 3 (Hinweis: Linting)

Im VS Code haben Sie bereits unterhalb des "Explorer"-Dateibaums einen Reiter "NPM Scripts" gefunden. Hier können Sie auch das gründlichere (statische, ohne Programmausführung) Überprüfen eines Frontend-Projekts anstoßen (Eintrag `lint`). `lint` sollte während der Entwicklung regelmäßig verwendet werden (*linting*), damit die Anzahl der entdeckten Probleme in jedem Schritt handhabbar bleibt. Da das generierte Demo-Projekt noch kaum Logik enthält, sollten sich keine JavaScript/TypeScript-Fehler finden, aber für die kommenden Aufgaben sollten Sie diese Funktion im Hinterkopf bewahren.