Software Engineering I (IB)

Blatt 2

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 09.10.2018 12:52

Vorbemerkung

Webseiten sind interaktiv. JavaScript ist eine Programmiersprache mit der sog. client-sided code, also Code auf dem Client, ausgeführt werden kann. Die Laufzeitumgebung stellt dabei i.d.R. die JavaScript-Engine des Browsers, z.B. die Chrome V8 oder Mozillas SpiderMonkey.

Da JavaScript einige Nachteile hat, gibt es einige Projekte, die darauf aufsetzen und versuchen die Probleme in den Griff zu bekommen. Ein weit verbreiteter Abkömmling ist TypeScript. TypeScript ist eine typisierte Obermenge von JavaScript. Mit dem TypeScript Compiler `tsc` wird TypeScript in JavaScript übersetzt.

Zur Softwareenwicklung im Bereich der Webentwicklung ist es sinnvoll ein Framework zu nutzen in dem schon viele komplexere Abläufe und Zusammenhänge integriert sind. Wir werden das weit verbreitete Angular-Framework nutzen.

Vorbereitungen auf dem privaten Rechner

Das Angular-Framework bringt ein CLI (Command Line Interface) mit. Dieses muss zunächst mit Hilfe von npm installiert werden. Nachdem wir im Laufe des Semesters auch noch mit Node.js entwickeln werden, installieren Sie zunächst einfach die LTS-Version von Node.js, die auch npm enthält.

Auf den Laborrechnern ist Node.js und npm bereits installiert.

Aufgabe 1 — Angular Quickstart

Die ersten Schritte mit dem Angular-Framework finden Sie unter Getting Started.

Installieren Sie das Angular-CLI mit dem Kommando

```
npm install -g @angular/cli
```

Der Schalter `-g` steht dabei für Global, also nicht im Projekt, sondern gobal auf dem Rechner. Das Kommando für das Angular-CLI heisst dann `ng`.

Mit dem Kommando

```
ng new my-app
```

können Sie im aktuellen Verzeichnis das Unterverzeichnis `my-app` erstellen welches ein Angular-Projekt enthält. Das dauert beim ersten Mal eine ganze Weile.

Sie können die Applikation anschließend mit `ng serve` starten. Dazu müssen Sie zunächst in das Verzeichnis `my-app` wechseln. Fügen Sie an das `ng serve` noch `--open` an, also

```
ng serve --open
```

wird die Applikation gleich in Ihrem Standardbrowser geöffnet. Wenn Sie anschließend in den Sourcen editieren, wird die Applikation beim Speichern gleich übersetzt und im Browser neu geladen, solange der `ng serve`-Befehl nicht abgebrochen wurde. Abbrechen können Sie ihn im Terminal übrigens mit `Ctrl-c`.

Das Angular-Projekt können Sie im WebStorm dann einfach öffnen und bearbeiten. Lesen Sie sich dazu den Step 4 durch und probieren Sie herum und lernen Sie die Struktur eines Angular-Projektes kennen.

Schöner Code

Bereits in den ersten beiden Semestern Softwareenwicklung habe ich versucht Ihnen zu vermitteln, wie wichtig die Lesbarkeit des Codes ist. Daher habe ich einen Stil vorgegeben und diesen per CheckStyle überprüfen lassen.

Stellen Sie für den Code, den Sie committen, selbst immer sicher, dass er mindestens mit WebStorm formatiert ist.

Aufgabe 2 — Tour of Heroes

Im Rahmen dieser Aufgabe sollen Sie das Angular-Tutorial Schritt für Schritt durcharbeiten. Ziel ist dabei nicht einfach nur den jeweiligen Code in ein eigenes Projekt zu kopieren, sondern möglichst alles zu **verstehen**.

Nutzen Sie für den Code ein leeres Repository welches Sie unter dem Link `https://classroom.github.com/a/HyeIt0ha` bekommen. Gehen Sie anschließend, wie im Tutorial beschrieben vor, um die Angular-Applikation zu erzeugen. Nachdem Sie die Aufgabe nicht abgeben müssen, können Sie **ausnahmsweise** auch direkt im `master` committen. Es bietet sich aber an den GitHub Flow oder den Git Flow weiter einzuüben.

Auf jeden Fall sollten Sie aber als allererstes das erzeugte Projekt, ohne dass Sie daran etwas editiert haben, als initiales Projekt committen. Anschließend bietet es sich an immer kleine Stücke zu committen und auch zu pushen, z.B. im Tutorial nach jedem *Final Code Review*.

Analog zur HTML/CSS-Aufgabe auf dem letzten Blatt, soll der jeweilige Stand als GitHub-Page verfügbar gemacht werden. Nachdem dazu das Angular-Projekt gebaut werden muss, nutzen wir dafür eine Travis CI. Travis CI ist eine *Continuous Integration*-Plattform, so wie der Jenkins, den wir in den ersten beiden Semestern genutzt haben.

Um den Travis CI zu nutzen (nachdem ich die gesamte GitHub-Organisation bereits angebunden habe), ist eine Datei mit dem Namen `.travis.yml` top-level in Ihrem Repository notwendig. Diese in YAML geschriebene Datei steuert die Abläufe auf dem Travis CI. Damit können Sie z.B. Tests ausführen oder ein Projekt erzeugen und sogar— das werden wir nutzen —die fertige Angular-App in ihr GitHub-Repository pushen.

Erzeugen Sie zunächst eine Datei `.travis.yml` mit folgendem Inhalt:

```yaml
language: node_js # Node.js muss auf dem Travis installiert werden
node_js:
- stable # ...und zwar die Stable-Version

# Der Travis-Job soll für alle Branches, außer dem gh-pages-Branch
# ausgeführt werden
branches:
  except:
  - gh-pages

script:
- npm run lint # erstmal checken ob alles schön ist
- npm run build # Angular App bauen

# und nach dem Bauen ins GitHub-Repo pushen
deploy:
  provider: pages # speziell für GitHub Pages
```

```
skip_cleanup: true # der Branch soll **nicht** vorher gelöscht werden
github_token: $GITHUB_TOKEN # das Token damit der Travis das darf,
                            # soll nicht in den Logs stehen, daher Variable
on:
  branch: master # auch wenn alle Branches gebaut werden, deployed werden
                 # soll nur vom master
local_dir: dist/angular-tour-of-heroes # in den Branch soll dann nur die
                                       # erzeugte Angular App
```

Damit das dann vernünftig läuft, müssen wir noch eine Zeile in der Datei `package.json` anpassen. Und zwar ersetzen Sie die Zeile

```
"build": "ng build",
```

durch

```
"build": "ng build --prod --base-href ./",
```

Um nun als Travis wirklich in Ihr Repository pushen zu dürfen, erzeugen Sie ein GitHub-Token unter https://github.com/settings/tokens. Wählen Sie dabei unter `Select scopes` den Oberpunkt `repo` aus. Gehen Sie anschließend auf die Travis CI Plattform um den GitHub-Token dort zu hinterlegen. Ersetzen Sie in der GitHub-URL Ihres Repositorys einfach `github` durch `travis-ci`, z.B.

`https://travis-ci.com/ob-swengiib-ws18/blatt-2-obcode`

Loggen Sie sich mit Ihrem GitHub-Account ein. Rechts unter `More Options` finden Sie den Punkt `Settings`. Tragen Sie im Abschnitt `Environment Variables` eine neue Variable ein mit dem Namen `GITHUB_TOKEN` und dem Wert Ihres GitHub-Tokens. `Display value in build log` muss dabei ausgeschaltet bleiben. Wenn Sie jetzt wieder auf der Seite des Repositorys (raus aus Settings) auf `Restart build` klicken, sollte der Travis am Ende des Builds die erzeugte Site in den `gh-pages`-Branch ihres Repositorys pushen. Damit ist die Site unter der entsprechenden URL erreichbar, z.B.

`https://ob-swengiib-ws18.github.io/blatt-2-obcode/`

Fügen Sie Ihre URL auf der GitHub-Page Ihres Repositorys oben zur Beschreibung hinzu. Klicken Sie dazu auf `Edit` rechts von dem Text `blatt-2-... created by GitHub Classroom` und tragen Sie die URL unter Website ein.