Übung Web-Anwendung

Patrick Bucher, seantis GmbH

29.08.2020

Viele Softwarelösungen werden heutzutage als Web-Anwendung umgesetzt. Gegenüber herkömmlichen Programmen, die zunächst auf einem Computer installiert werden müssen, haben Web-Anwendungen viele Vorteile. Man kann von jedem Computer auf sie zugreifen, sofern ein Browser installiert und eine Internetverbindung zur Verfügung steht. Auch funktioniert Sie auf allen Betriebssystemen ohne Anpassungen.

In diesem Dokument findest Du Übungen zu einer beispielhaften Web-Anwendung. Es handelt sich dabei um eine kleine Notenverwaltung, die mit der Programmiersprache Python und dem Framework Flask entwickelt worden ist.

Einrichten der Umgebung

Der Instruktor stellt Dir eine Umgebung bereit, womit Du einerseits die Web-Anwendung ausführen, und sie andererseits weiterentwickeln kannst. Sollte etwas nicht funktionieren, frage einfach beim Instruktor nach.

Mache Dich mit der Notenverwaltung etwas bekannt. Du kannst Informationen einsehen, hinzufügen und löschen. Dabei kann nichts kaputt gehen. (Andernfalls kann der Instruktor die Datenbank wiederherstellen.)

Aufgaben

Mit den folgenden Aufgaben kannst Du Dich mit dem Aufbau einer modernen Web-Anwendung beschäftigen. Dabei lernst Du verschiedenste Technologien kennen, die man als Web-Entwickler heutzutage kennen sollte. (Für manche Technologien gibt es auch Alternativen.)

Halte Deine Lösungen in einer separaten Textdatei fest. (Speichere diese häufig ab, um keine Informationen zu verlieren.) Obwohl man eine Textdatei nicht wie ein Word-Dokument formatieren kann, sollst Du diese möglichst gut strukturieren und lesbar formatieren. Bewahre diese Datei bis zum Schluss auf!

Es geht nicht darum, alle Aufgaben möglichst korrekt und vollständig zu lösen, sondern einen möglichst breit gefächerten Eindruck über die verschiedenen Technologien zu erhalten. Wenn Du bei einer Aufgabe lange anstehst, frage den Instruktor um Hilfe. (Alle beschriebenen Technologien zu beherrschen ist nicht das Ziel des heutigen Nachmittags, sondern dasjenige einer vierjährigen Lehre.)

Zurechtfinden im Verzeichnis

Das Arbeitsverzeichnis `flask-demo-app` enthält verschiedene Unterverzeichnisse und Dateien. (Hinweis: die Verzeichnisse `env` und `__pycache__` kannst Du für alle Übungen ignorieren, aber nicht löschen!)

Aufgabe 1.1: Stelle eine Liste mit allen Dateiendungen zusammen, die Du im Arbeitsverzeichnis finden kannst.

Aufgabe 1.2: Versuche die verschiedenen Dateien zu öffnen. Schaue Dir den Inhalt der Dateien an, und versuche zu verstehen, was dieser Inhalt wohl bedeuten könnte. Suche Dir pro Dateiendung eine Datei aus, und versuche deren Inhalt stichwortartig zu beschreiben.

Cascading Style Sheets (CSS): Definition der Darstellung

Das Aussehen einer Webseite oder Web-Anwendung wird heutzutage über *Cascading Style Sheets* (kurz: CSS) gesteuert. In der Datei `static/style.css` findest Du die Definition für die Darstellung der Notenverwaltung.

Eine Stildefinition besteht jeweils aus zwei Teilen: Einem Selektor, der bestimmt, für welche Elemente bestimmte Regeln gelten; und den eigentlichen Regeln (in geschweiften Klammern).

Aufgabe 2.1: Schaue dir den Notenüberblick an. Erfasse nun ein paar zusätzliche Fächer. Was fällt Dir auf, wenn Du den Notenüberblick erneut öffnest? Kannst Du die Darstellung der Seite verbessern, indem du Änderungen an `style.css` durchführst? (Forciere das Neuladen der Änderungen mit `[Ctrl]+[Shift]+[R]`.)

Aufgabe 2.2: Die Anwendung verwendet ein Farbschema, das auf blauen Farbtönen basiert. Verwende das Material Design Color Tool, um ein neues Farbschema zu kreieren. Du musst die alten Farben in `style.css` durch die neuen ersetzen. Welche Angaben musst Du ändern?

Hypertext Markup Language (HTML): Beschreibung der Seitenstruktur

Die Struktur einer Webseite wird in HTML angegeben. Da eine Web-Anwendung nicht bloss *statische* Seiten (mit immer gleichbleibendem Inhalt), sondern *dynamische* Seiten verwendet, ist die HTML-Struktur in sogenannten *Templates* (Vorlagen) definiert. Du findest also keine `.html`-Dateien im vorliegenden Projekt.

HTML basiert auf sogenannten *Tags*. Dies sind Elemente, die in spitzen Klammern eingeschlossen sind. Die meisten Tag müssen geschlossen werden. Hier ein Beispiel für einen Titel (*Headline 1*):

```
<h1>Meine Webseite</h1>
```

Tags können verschachtelt werden. Dabei muss jeweils derjenige Tag zuerst geschlossen werden, der als letztes geöffnet worden ist. Hier ist ein Absatz, bei dem ein Wort speziell hervorgehoben wird:

```
<p>Hier ist ein <strong>wichtiges</strong> Wort.</p>
```

Jinja2 Templates: Vorlagen für dynamisches HTML

Im Verzeichnis `templates/` findest Du verschiedene Dateien mit der Endung `.jinja2`. Jede Datei beschreibt eine andere Seite bzw. einen anderen Seitenbereich der Web-Anwendung.

Ein grosser Teil des Codes ist in HTML geschrieben. Im Gegensatz zu den HTML-Tags werden bei Jinja2-Templates geschweifte Klammern eingesetzt. Dabei gibt es zwei wichtige Arten von Elementen:

Erstens: Kontrollstrukturen, die in `{% ... %}` eingefasst sind:

```
{% if grade > 4.0 %}
    <p>die Prüfung wurde bestanden</p>
{% endif %}
```

Zweitens: Werte, die in `{{ ... }}` eingefasst sind:

```
<p>Die Prüfung wurde mit der Note {{ grade }} bestanden.</p>
```

Aufgabe 3.1: Besuche die verschiedenen Seiten der Notenverwaltung. Öffne anschliessend die verschiedenen Template-Dateien im Unterverzeichnis `templates/`. Kannst Du diese Templates den verschiedenen Seiten zuordnen? Gibt es zu jedem Template eine Seite?

Aufgabe 3.2: Die Schüler werden in einer Liste dargestellt. Das Element `<ul>` definiert eine ungeordnete Liste (*unordered list*). Diese enthält `<li>`-Elemente (*list item*). Ändere das entsprechende Template, sodass die Schüler nicht mehr als Listenelemente, sondern als Absätze (`<p>`: *paragraph*) dargestellt werden. Was musst Du dabei besonders beachten?

Aufgabe 3.3: Die Notenübersicht eines Schülers (z.B. für Bettina) zeigt unten ein Formular, womit neue Noten eingetragen werden können. Die Note wird als normales Textfeld angezeigt. Was kannst Du in dieses Feld alles eintragen? Siehst Du hier ein mögliches Problem?

Aufgabe 3.4: Damit nur noch Zahlen als Noten eingetragen werden können, soll das entsprechende Textfeld (`<input id="grade" ...>`) durch ein numerisches Eingabefeld ersetzt werden. Schaue Dir die Dokumentation zu diesem Feld an und nimm die entsprechende Anpassung im Template vor. Definiere sinnvolle Werte für die Attribute `min`, `max`, `step` und `value`. Welche Probleme können dadurch gelöst werden?

JavaScript (JS): Interaktives Verhalten im Browser

CSS und HTML sind keine wirklichen Programmiersprachen, sondern Beschreibungssprachen. Mit Jinja2 hingegen kann man die Ausgabe von Webseiten ausprogrammieren. Zusätzliche Interaktion im Browser kann mit einer weiteren Sprache gesteuert werden: JavaScript.

In der Notenverwaltung kommt JavaScript nur an zwei Stellen vor: Beim Löschen eines Faches oder eines Schülers wird der Benutzer zunächst gefragt, ob er den Eintrag *wirklich* löschen will. So wird verhindert, dass der Benutzer aus Versehen Daten löscht.

Aufgabe 4.1: Öffne die Datei `static/script.js`. Versuche den Code zu verstehen und beschreibe stichwortartig, was hier ungefähr passieren könnte.

Aufgabe 4.2: Die Link-Elemente werden mithilfe der eingebauten Funktion `doc.querySelectorAll('a.delete')` ermittelt. Kommt Dir die Angabe `a.delete` bekannt vor? Wo gibt es ähnliche Bezeichnungen?

Aufgabe 4.3: Lösche die Zeile `e.preventDefault();` und lade die Notenverwaltung neu mit `[Ctrl]+[Shift]+[R]`, damit der JavaScript-Code sicher nachgeladen wird. Versuche nun ein paar Schüler zu löschen. Die entsprechende Bestätigung kannst du einmal verneinen und einmal bestätigen. Was fällt Dir dabei auf? Was bedeutet "`preventDefault`" auf Deutsch übersetzt? Was für eine Bedeutung hat es? Was könnte das Ausrufezeichen (`!`) in der Zeile darüber bedeuten?

Structured Query Language (SQL): Zugriff auf Datenbanken

Die Noten werden in einer sogenannten *relationalen Datenbank* abgespeichert. Eine relationale Datenbank besteht aus Tabellen, in denen man verschiedene Daten der gleichen

Struktur ablegen kann. Oftmals stehen die einzelnen Tabellen in Beziehung (englisch: "in relation") zueinander.

Relationale Datenbanken werden mit einer Sprache namens SQL definiert und abgefragt. Es gibt verschiedene Befehle um die Datenstruktur zu definieren (DDL: *Data Definition Language*), Daten abzufragen (DQL: *Data Query Language*), Daten zu manipulieren (DML: *Data Manipulation Language*) und den Zugriff auf Daten zu kontrollieren (DCL: *Data Control Language*).

Aufgabe 5.1: Die Datei `tables.sql` enthält die Definition der Datenbanktabellen. Jede Tabelle hat einen Namen und mehrere Spalten. Eine Tabelle hat zusätzlich Beziehnungen zu anderen Tabellen. Versuche diese Tabellen auf einem Blatt Papier aufzuzeichnen. Beispieldaten (zwei, drei pro Tabelle genügen) kannst du erfinden und als Zeilen eintragen.

Aufgabe 5.2: Die Dateien `data.py` und `insert_test_data.py` enthalten zwar in erster Linie Python-Code (siehe nächstes Kapitel), dieser enthält aber wiederum SQL-Befehle. Suche Dir einige dieser Befehlszeilen heraus und überlege Dir, was mit diesen erreicht wird.

Aufgabe 5.3: Zu welcher Gruppe (DDL, DML, DQL, DCL) gehören die einzelnen Befehle? Begründe deine Aussage stichwortartig.

Python: Serverseitige Programmlogik

Der grösste Teil der Programmlogik einer Web-Anwendung wird nicht im Browser, sondern auf dem Server ausgeführt. So kann sichergestellt werden, dass die Logik für alle Benutzer gleich ist. Zudem hat der serverseitige Code direkten Zugriff auf die Datenbank, die auch auf dem Server abgelegt ist.

Python unterstützt verschiedene Programmierstile: strukturiert, objektorientiert, funktional. Hier soll es zunächst um die strukturierte Programmierung gehen. Diese besteht u.a. aus den folgenden Elementen:

- Variablen: z.B. `a`, `length`, `result` oder `grade` mit verschiedenen Typen
- Anweisungen: z.B. `a = 3`, welche der Variablen `a` den Wert `3` zuweist
- Sequenzen: eine Reihe von Anweisungen, die hintereinander ausgeführt werden
- Entscheidungen: Anweisungen, die nur unter bestimmten Bedingungen ausgeführt werden
- Wiederholungen: Anweisungen, die mehrmals hintereinander ausgeführt werden

Das folgende Programm kombiniert all diese Elemente:

```
eingabe = input("Wie viele Wiederholungen sind gewünscht? ")
wiederholungen = int(eingabe)
if wiederholungen < 0:
    print("Sie wünschen keine Wiederholungen")
else:
    while wiederholungen > 0:
        print("Es gibt noch", wiederholungen, "Wiederholungen")
        wiederholungen = wiederholungen - 1
print("Jetzt ist Schluss!")
```

- Das Programm verwendet die Variablen `eingabe` und `wiederholungen`.

- Es gibt verschiedene Anweisungen zur Eingabe (`input`), zur Ausgabe (`print`) und Zuweisungen (=).
- Die ersten beiden Anweisungen werden als Sequenz abgearbeitet.
- Die Prüfung, ob `wiederholungen` kleiner als 0 ist, ist eine Entscheidung: Je nach Ergebnis wird anderer Code ausgeführt.
- Mithilfe einer `while`-Schleife wird eine Wiederholung durchgeführt. Die Ausgabe der verbleibenden Wiederholungen und das Verkleinern des Wertes um eins wird bei jedem Durchlauf durchgeführt.

Bei Python ist es wichtig, dass der Code, der zu einer Entscheidung oder Wiederholung gehört, immer gleich eingerückt wird.

Aufgabe 6.1: Das Projekt enthält vier Python-Dateien (Dateiendung `.py`). Öffne diese, überfliege den Code, und überlege Dir, was wohl in den verschiedenen Dateien passiert. Notiere deine Ideen stichwortartig.

Aufgabe 6.2: In der Datei `calculations.py` gibt es eine Funktion namens `calculate_deficiency_points`, welche die Mangelpunkte eines Schülers berechnet. Diese gibt immer den Wert `0.0` zurück. Die Funktion wurde noch nicht ausprogrammiert. Lies die Anweisungen in der Funktion durch, und versuche, sie korrekt auszuprogrammieren. (Die Zeilen, die mit `#` beginnen, sind Kommentare, und werden nicht ausgeführt.) Um die Funktion zu testen kannst du im Notenüberblick einen Schüler mit ungenügenden Noten öffnen. Die Mangelpunkte sollten dann oberhalb des Titels *Note hinzufügen* angezeigt werden.

Aufgabe 6.3: Öffne die Datei `app.py`. Diese enthält verschiedene Funktionen, die jeweils mit der Zeile `@app.route` beginnen. Öffne die verschiedenen Seiten der Notenverwaltung im Browser. Schaue Dir die Adresszeile genau an. Kannst Du anhand der Adresse die verschiedenen Seiten den verschiedenen Funktionen in `app.py` zuordnen?

Makefile: Erzeugen von Dateien anhand von Abhängigkeiten

Das Projekt enthält eine Datei namens `Makefile`. Diese Datei wird zum Erzeugen der Datenbank (`grades.db`) und des vorliegenden Dokuments verwendet.

Aufgabe 7.1: Zum Erstellen der Datenbank wird ein Python-Programm verwendet, das Du bereits kennengelernt hast. Um welche Python-Datei handelt es sich dabei?

Aufgabe 7.2: Die Tabellen werden mithilfe von `tables.sql` in reinem SQL erzeugt. Die Testdaten werden hingegen mit einem Python-Skript `insert_test_data.py` erzeugt. Warum wird nicht an beiden Stellen einfach SQL verwendet?

Shell-Skripte: Nützliche Helferlein für verschiedene Aufgaben

Das Projekt enthält zwei Dateien mit der Endung `.sh`: `run.sh` zum Ausführen der Web-Anwendung, und `install.sh` zum Installieren derselben. Dies sind kleine Skripte, mit denen verschiedene Abläufe automatisiert werden.

Aufgabe 8.1: Öffne die beiden Dateien. Findest Du darin Referenzen auf andere Dateien, die Du bereits kennst? Wenn ja, öffne diese, und notiere Dir die Dateinamen. Was enthalten diese Dateien?

Weitere Übungen zu Shell-Skripten folgen im nächsten Block.

Ausblick

Du hast eine sehr kleine Web-Anwendung kennengelernt. Selbst ein erfahrener Programmierer benötigt ungefähr einen halben Arbeitstag, um so eine Anwendung zu erstellen. Damit eine solche Anwendung produktiv eingesetzt werden kann, benötigt es aber noch einiges an Zusatzaufwand.

Aufgabe 9.1: Was gefällt Dir an der Notenverwaltung? Was stört Dich an ihr? Würdest Du das Programm Deinen Lehrern empfehlen oder Ihnen davon abraten?

Aufgabe 9.2: Welche Probleme wären noch zu lösen, um die Notenverwaltung im Internet als Produkt («*Software as a Service*») anbieten zu können? Haben diese Probleme mit dem Programmieren oder mit anderen Tätigkeiten zu tun?

Fazit

Selbst eine triviale Web-Anwendung wie die Notenverwaltung basiert auf einer Vielzahl von Technologien: CSS, HTML, Jinja2, JavaScript, SQL, Python. Das `Makefile` und die Shell-Skripte erleichtern zudem den Alltag des Entwicklers.

In den Übungen hast Du nicht nur diese Technologien kennengelernt, sondern auch eine Idee davon erhalten, wie diese Technologien zusammenarbeiten. Ein Web-Entwickler muss also nicht nur einzelne Technologien beherrschen, sondern auch wissen, in welchem Zusammenhang diese stehen.

Eine Anwendung muss nicht nur entwickelt, sondern auch gewartet, betrieben und vermarktet werden. Ein Softwareentwickler muss auch in diesen Bereichen mithelfen.