IWGS2_SS20_Zettel6

July 7, 2020

1 Informatische Werkzeuge in den Geistes- und Sozialwissenschaften II

1.1 Hausaufgabe 6 (Einfügen, Löschen, Editieren in SQL und Bottle)

Erschienen: 06.06.2020
Abgabe bis: 21.06.2020

Bitte laden Sie Ihre Notebooks bis 23:59 Uhr am Abgabetag in Ihrer Übungsgruppe bei StudOn hoch.

Wenn Ihnen einige der hier verwendeten Konzepte unbekannt sind oder Sie nicht wissen, wie Sie fortfahren sollen, können Sie die Vorlesungsunterlagen zu Rate ziehen oder jederzeit Fragen im Forum oder auf Slack stellen, im Tutorium nachfragen, oder Ihrem Tutor eine Mail schreiben.

1.2 Präambel

Beatrice hat Ihnen eine lange Email geschrieben. Irgendwie wirkt sie gerade besonders nachdenklich. Sie bedankt sich für Ihre bisherige Hilfe beim Aufbau der Datenbank.

Letzte Woche musste sie einen Philosophieaufsatz in ihrem Zweitstudium abgeben, in dem sie über Impermanenz und Veränderung auf dem persönlichen, kulturellen und ideologischen Level referieren sollte. Nachdem sie so viel drüber nachgedacht hat und den wirklichen Wert von Veränderung für menschliches Zusammensein verinnerlicht hat, fühlt sie sich gezwungen, auch an anderer Stelle, nämlich in der erwähnten Datenbank, Änderungen und Verfall zu erlauben. Alles andere wäre wider die höhere Ordnung des Universums.

Sie möchte deshalb als nächstes die Datenbank um ein paar letzte fehlende grundlegende Operationen erweitern. Unter anderem möchte sie das Editieren und Löschen von Daten erlauben, um Veränderung und Verfall bestmöglich abzubilden.

1.3 Aufgabe 6.1 (Primary Key, 10 Punkte)

Um Einträge in der Datenbank editieren und löschen zu können, ist es praktisch, jedem Dateneintrag eine eindeutige ID zuzuweisen. Dies macht man in relationalen Datenbanken über einen Primary Key. So einen Key haben wir in Aufgabe 4 schon verwendet.

Aufgabe: Erweitern Sie *image_database.py* so, dass die Tabelle *Images* einen Primary Key `ID` als erste Spalte enthält. Die ID soll ein Integer sein.

Schauen Sie sich außerdem nochmal an, wie die Einträge in dem Skript in die Tabelle eingefügt werden. Wichtig ist, dass hier explizit nochmal die Spaltennamen angegeben werden `(Title, Subtitle, ...`. Sie müssen diese Zeile **nicht** verändern. SQL wird neuen Einträgen automatisch eine neue ID zuweisen, die sich mit jedem neuen Eintrag um 1 erhöht.

Führen Sie das Skript *image_database.py* danach aus, damit wir die ID in den späteren Teilaufgaben verwenden können.

1.4 Aufgabe 6.2 (Neue Einträge, 40 Punkte)

Ein paar Kleinigkeiten haben sich im Vergleich zu letzter Woche am Bottle-Server verändert. Am einfachsten sehen Sie das, indem Sie den Server ausführen und ausprobieren. Beachten Sie, dass sie wieder den Port ändern müssen.

Wir haben am Anfang von *image_server.py* folgende Zeile eingefügt:

```
db.row_factory = sqlite3.Row
```

Diese bewirkt, dass die Ergebnisse von SQL-Anfragen nicht wie Tupel verwendet werden müssen, sondern wie Dictionaries. Ein Beispiel:

```
cursor = db.cursor()
cursor.execute("SELECT Title ...")

data = cursor.fetchone() # Wir fragen den Cursor hier nur nach dem ersten Element, nicht nach

# Vorher haben wir Ergebnisse so verwendet:
title = data[0]
# Das geht immernoch. Allerdings geht jetzt auch das:
title = data["Title"]
```

Auf der Webseite sehen Sie 3 neue Buttons (*Add new entry*, *Edit*, *Delete*). Die Knöpfe funktionieren bisher nicht. In dieser Teilaufgabe widmen wir uns dem *Add new entry*-Button.

Das Einfügen von neuen Elementen findet in 2 Schritten statt. In einem ersten Schritt bieten wir BenutzerInnen unserer Seite ein Formular, in dem die neuen Daten eingetragen werden. Diese werden dann an den zweiten Schritt übertragen, der die Daten dann in die Datenbank einfügt. Deshalb sind in *image_server.py* zwei neue Routen, `add` und `finish_add`, die diese beiden Schritte durchführen sollen.

Zur Erinnerung: Ein HTML-Formular hat folgende Form:

```
<form action="/target_url" method="post">
    Input text here: <input type="text" name="my_name" value="my_value"><br>
    ...
    <input type="submit" value="Submit">
</form>
```

Das `method`-Attribut von `form` ist optional. Standardmäßig hat es den Wert `get`, was bedeutet, dass wir einen GET-Request starten, sobald auf den Submit-Button geklickt wird. So haben wir letzte Woche unsere Daten übertragen und so funktioniert unser Suchformular. Hier verwenden wir jetzt `post`, für einen POST-Request. Auch hier werden Daten übertragen, aber nicht in der URL, sondern separat. Auf diese Weise kann man größere Datenmengen übertragen.

Das `<input>`-Feld hat den Typ `text`, weil wir Text eingeben wollen. Über das `name`-Attribut können wir dann in unserem Python-Code die Werte abfragen. `value` ist optional und beschreibt einen Wert, der schon vorausgefüllt in dem Textfeld stehen soll, bevor BenutzerInnen etwas eingegeben haben. `value` werden wir in dieser Teilaufgabe nicht verwenden, aber später beim Editieren dann schon.

Aufgabe: Erstellen Sie eine Template-Datei *add.tpl*, die ein HTML-Formular anzeigt. Das Formular soll Textfelder für alle Spalten in unserer Datenbank haben, mit Ausnahme von *ID* und *FileName*. Wie oben beschrieben, wird die *ID* automatisch vergeben. Für *FileName* werden wir in späteren Übungsaufgaben ein System bauen, mit dem man Dateien auf unsere Webseite hochladen kann. Für den Moment tragen wir für neue Datenbankeinträge `None` im *FileName* ein.

Ihr Formular soll auf die Route `/finish_add` weiterleiten. Für diese Route ist schon etwas Python-Code vorgegeben. Schauen Sie sich den Code genau an. Setzen sie die `name`-Attribute in Ihrem Formular so, dass sie mit dem Python-Code übereinstimmen.

Vervollständigen Sie `finish_add` so, dass die Daten in die Tabelle aufgenommen werden. Testen Sie Ihren Server gründlich. Funktioniert das Einfügen von Daten? Taucht das neue Element in der Liste auf der Hauptseite auf?

Testen Sie auch folgendes: Starten Sie Ihren Server neu. Ist das eingefügte Element immernoch in der Liste? Wenn nein, erinnern Sie sich, was nötig ist, um Daten permanent in die Datenbank zu schreiben und passen Sie Ihre `finish_add`-Funktion entsprechend an.

1.5 Aufgabe 6.3 (Editieren, 30 Punkte)

Das grundsätzliche Vorgehen beim Editieren ist genauso, wie beim Einfügen. Auch hier haben wir zwei Schritte: erst ein Formular und dann das eigentliche Editieren.

Aufgabe: Erstellen Sie eine Template-Datei *edit.tpl*. Diese sieht im Wesentlichen aus wie *add.tpl*, nur dass wir dieses Mal das `value`-Attribut verwenden, um den bisherigen Wert eines jeden Elements anzuzeigen. Vervollständigen Sie dazu auch die Route `/edit` in *image_server.py*. Diese Route bekommt als Argument die ID eines Datenbankeintrages (also den Primary Key). Verwenden Sie den Key, um den entsprechenden Eintrag in der Datenbank abzufragen. Übergeben Sie dann die Daten an Ihre neue Template-Datei *edit.tpl*, damit diese die Werte in den `value`-Attributen anzeigen kann.

Ihr Formular soll dieses Mal auf die Route `/finish_edit/<ID>` weiterleiten, wobei `<ID>` die ID des Bildes ist. Wenn Sie zum Beispiel das Bild mit der ID 27 editieren möchten, soll Ihr Formular auf die Route `/finish_edit/27` weiterleiten.

Fragen Sie danach in `/finish_edit` wieder (wie oben schon) die Daten aus dem POST-Request ab. Führen Sie dann den geeigneten SQL-Befehl aus, um den entsprechenden Datenbankeintrag zu editieren. Denken Sie auch hier wieder daran, Ihre Änderungen an der Datenbank permanent zu machen.

1.6 Aufgabe 6.4 (Löschen, 20 Punkte)

Löschen ist einfacher als die anderen beiden Routen, schon alleine weil wir nur eine Route brauchen.

Aufgabe: Fügen Sie in *image_server.py* eine neue Route `/delete/<id>` ein, die (wie `/edit`) die ID übergeben bekommt. Führen Sie dann den geeigneten SQL-Befehl aus, um das Element mit der ID zu löschen.

[ ]: