Lab assignment 3

TI1506 2017/18; ti1506-students.slack.com

Introduction

In the first part of this assignment, you will you will write node.js code and use Ajax/JSON to enable the client and server to communicate with each other. In the second part of this assignment, you will experiment more with the relational database environment by creating some queries, and testing their execution on the `Habit` database.

Deadline: when this assignment is due depends on the cluster your team is in. Every team is assessed bi-weekly. Make sure you know your cluster and what this means for your lab deadlines. All of this information is available in the course syllabus (on blackboard).

In order to pass the assessment, you have to make a valid attempt at working through the lab assignment.

1. Node.js

So far your application can handle basic interactions, through client-side JavaScript. Lets now implement a first node.js script; if you don't know where to start look at Chapters 5 & 6 of the Web course book and Example 6 of the node.js lecture's code examples[1].

Your script should be able to do the following:

- Keep a list of the habits **in memory** on the server.
- Allow the client (the browser) to **retrieve the habits** from the server; use the **JSON** format for this task.
- Allow the client to **add** a habit to the server.
- Allow the client to **update** a habit on the server.
- Allow the client to **delete** a habit from the server.

2. Ajax

Use Ajax to allow the dynamic updating of the habit page in the browser (i.e. without reloading of the complete page). The lecture's code *Example 6* and book chapters 5 & 6 of the Web course book will help you if you are stuck.

Note: Example 6 also contains a hint of how to enable repeated checking of the server (i.e. every X seconds the habit page polls the server for the habits). This can easily be achieved by wrapping this line into a timer function:

[1] https://github.com/chauff/TI1506-node.js

```
setInterval(function () {
    console.log("Fetching the habit list from the server.");
    $.getJSON("habits", addHabitsToList);
}, 2000);
```

This piece of code will retrieve a list of habits from the server every 2 seconds (it is your job to check whether those habits differ from what is already there).

3. Query `Habit` Database

In this exercise, you will practice with the use of SQL as a DML, to query the data contained in the provided *Habit* database.

Design choice explanation: the database has been designed with some assumptions in mind. A user can have one or more habit lists, a list can contain one or more habits. A habit has a frequency, which indicates a timeframe in which the habit should be performed once. For DAILY habits, a filter can be applied to limit on which days the habit is expected to be performed, once per defined day. For WEEKLY habits, a part of the week can be defined to indicate when in a week the habit is expected to be performed, once every week in that part of the week. A habit can be performed any number of times, even on the same day.

Assignment: Compose and execute the SQL required to provide the correct answer to the following queries.

1. List all the habit lists belonging to a given user. The identifier of the user should be specified as a condition in the WHERE clause of the query.
2. List all the habits belonging to a given habit list. The identifier of the habit list should be specified as a condition in the WHERE clause of the query.
3. As in (2), but now allow for pagination of habits. This means that the query should show a pre-defined amount of results, starting from a given tuple. Use the LIMIT clause implemented in MySQL (http://dev.mysql.com/doc/refman/5.7/en/select.html)
4. Create a query for each of the following habit filters
    a. a frequency filter;
    b. if they are part of a public list;
    c. Completed within a certain date range;
5. For a given user, list all their habits (across habit lists).
6. For a given habit, show all days of the week that are assigned to it.
7. For a given day of the week, show all habit lists that contain habits that are assigned to that day.

8. For a given day of the week, show how many habits that are assigned to that day have been completed once or more, and how many have never been completed.
9. For each week in the current year, calculate how many times a habit was completed.
10. Give the top 5 habits ordered by the difference between the first and last time they were completed. Limit your query to habits for which the first and last occurrence are at least 2 weeks apart
11. Calculate the frequency of co-occurrence of assigned days of the week (i.e. the number of times each possible combination of day-of-week pairs is used in the database)
12. For a given habit list, calculate the average number of habit completions.
13. List the habits that are completed more often than the average number of completions for all habits in the same list.

Deliverable: For each of the information needs above, write your query in a text file, together with the retrieved answer list. Make sure to have the text file with you during the assessment, and be prepared to execute your queries "live" on your laptop.