Assignment 8 (HF)

Due: Mon 22.12.2014; 16:00h (2 Weeks)

Please note: Although the due date is two weeks from now, there will be another assignment next week!

Task 1: Retrieve Suggestions from a Database

This task builds on the solution of assignment 7. An example solution[1] will be provided on the website starting Monday 15.12.2014. However, you are encouraged to solve assignment on your own.

Last week, you built a website, where people could vote for certain suggestions. The votes were non-persistent, so restarting the web app lead to data loss. In this task, you should address this issue and make user votes persistent.

Use MongoDB to store the suggestions and the according up or down votes. The modifications that you have to do are quite small, but test them carefully.

a) In data/data.js there is an array of JSON objects containing the suggestions. Take this array and insert it into a database. Please use the following naming convention for the database: <lastname>-assignment08. Since databases are created implicitly this makes inspecting your submissions easier for us. It then does not matter how you name your collection(s).
b) Modify app.js such that the monk middleware injects a db object to all requests (see tutorial slides).
c) Modify the file that you use to serve and render the suggestions such that the suggestions are now read from a MongoDB database. Beware of the asynchronous nature of queries and time the response carefully, or else the results are never sent to the client.

Task 2: Allow the user to submit a suggestion

Now the user should be able to submit suggestions. A simple text field with a submit button should do for now. Consider this:

a) Modify your template file to show a simple input field above the list of suggestions.
b) On the client side, modify your script to handle the form submit event. The handler should call the API helper to submit the suggestion. Once the API call is done, the UI updates with the newly added suggestion.
c) Also on the client side, extend your API helper and send the suggestion via POST. A rudimentary JSON object in the form {suggestionText: 'Lorem Ipsum'} is all this should take.
d) The backend should allow POST requests on `'/service/suggestions'`. Perform a sanity check (e.g. is the text length > 0) and put the suggestion with all necessary keys into your collection in the database. Send the response to the client only after this operation has finished.

[1] Note: No solution provided by us claims to be the „best solution". In fact, we try to keep things as simple as possible and deliberately refuse to utilize more advanced concepts.

e) It is advised to mark the suggestion with the user's IP and that they have voted with 'up'.

There are a number of ways to handle the response now. Pick one below or think of something else:

a) If the suggestion was recorded, display a notification informing the user to refresh the page (an alert or, more unobtrusive, a message in a dedicated notification area that slowly fades out).
b) Make the browser refresh the page to view the update.
c) Construct the response in a way that you can handle it on the client side gracefully. Create a new DOM node and put the suggestion in the according spot. Make sure that the user can't vote for it twice.
d) Use socket.io to update the view. If you went the rocky but more elegant path with socket.io, you can probably easily extend the functionality to show new suggestions on the clients as they are made.

Task 3: AngularJS vs jQuery

As of now, you should be quite familiar with jQuery. However, AngularJS is a new framework following a rather different approach. Try and find out what it is:

a) Read the Wikipedia Article on AngularJS (https://en.wikipedia.org/wiki/AngularJS). Describe in your own words what advantages Angular's core concepts bear compared to jQuery.
b) Having heard about dynamic web applications during the lecture, what do you think is the most important motivation for using AngularJS?

Task 4: MEAN vs LAMP

You now have made the acquaintance of all four letters in MEAN.

a) Describe in your own words what the most significant differences are between MEAN and LAMP.
b) What could hamper growing success of the MEAN stack?
c) Given the following scenarios, would you choose MEAN or LAMP? Explain why you would make that decision
    a. A client asks you to set up a company blog on the website. There are around 10 different authors, and the blog will be read by 2000 visitors per month.
    b. You want to create a site that queries Twitter Tweets containing a certain hashtag and display those tweets immediately on the website.
    c. You have an idea for a new portal that compares flight prices. More than 2 Million visitors are expected each month.
    d. You offer a service similar to Google Keep, where people can take notes and create simple to-do-lists.
    e. You want to create the next big social network that challenges Facebook.

Further notes:

- Please make sure to comment your code sufficiently to facilitate correction of your submission.
- Incomplete submissions are welcome. However, please make sure to include a "README" text file to tell us how far you have come.