

Exercise Sheet 6

Submit until Tuesday, December 1 at **2:00pm**

Exercise 1 (5 points)

Extend the server code provided on the Wiki (equivalent version in Python, Java, and C++) to serve HTTP requests for HTML, JavaScript, CSS, and plain text, with the correct response headers. When a requested file is not found, return an appropriate 404 header.

Note: This was done live in the lecture, however, it is not part of the code provided on the Wiki. The reason is that you only learn something about this by doing it yourself.

Exercise 2 (5 points)

Extend your server code to answer fuzzy prefix queries using your code from the last exercise sheet. Requests should be of the form `http://<host name>:<port>/?q=<query>`. Use the dataset for sheet 6 provided on the Wiki.

Exercise 3 (5 points)

Write a simple web application, where the user can type something into a search field. After each keystroke, the top-10 fuzzy prefix matches should be computed by communicating with the server from Exercise 2, and then interactively displayed on the page. Use an HTML file, a CSS file, and a file containing the JavaScript code, as explained and shown in the lecture by example. The simple HTML, JavaScript, and CSS from the lecture are available on the Wiki.

Note 1: Your Makefile should contain two variables `PORT` and `RECORDS` and a target `start:` that starts your server on the given port, reading records from the given file. See the Makefile on the Wiki for an example. Java users should have a Makefile (with only the `start:` target) in addition to their `build.xml`.

Note 2: The web application should be callable merely with `http://<host name>:<port>`, without any suffix. It is easy to modify the server code such that your search HTML is returned for the corresponding GET request.

Note 3: Take care that the hostname is obtained automatically by your server code, and not hard-coded anywhere else. Similarly, the port should not be hard-coded anywhere in your HTML or JavaScript.

[turn over with enthusiasm]

Exercise 4 (5 points)

Make your web application nicer. As a minimum, you should add at least one non-trivial functional feature + enhance the HTML/CSS to improve the appearance of the page. Examples of such features are: show the top-10 matches in a nice drop-down box (using jQuery UI), add a feature to show more than 10 matches if there are more, with mouse over a displayed match show the picture from the corresponding Freebase page (if available).

Add your code to a new sub-directory *sheet-06* of your folder in the course SVN, and commit it. Make sure that *compile*, *test*, and *checkstyle* run through without errors on Jenkins. Also commit the usual *experiences.txt* with your much appreciated, brief and concise feedback.