Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Patrick Brosi

Information Retrieval
WS 2016/2017

`http://ad-wiki.informatik.uni-freiburg.de/teaching`

Exercise Sheet 7

Submit until Tuesday, December 13 at **2:00pm**

This exercise sheet (ES7) is the continuation of the last exercise sheet (ES6). If you missed the last sheet for whatever reason, please read it first, then continue reading here.

In particular, make sure that you have a Makefile with target *start*, as explained in Item 2 of ES6, and that your web app is reachable under the URL *http://<host>:<port>* (without anything after the port), as explained in Item 4 of ES6.

Please don't link to or include code from your *sheet-06* folder but copy it to the *sheet-07* folder and continue from there.

Make sure you use the new *cities2.txt* on the Wiki for this sheet.

Exercise 2 (20 points)

Extend your web application from ES6 by the following components and functionality. Items 1, 2 and 7 already appeared on ES6.

1. Write the JavaScript (and include it into your HTML file) such that in your web app, after each keystroke, a list of fuzzy completions of whatever the user typed so far is displayed. The list should contain 10 items (cities), or less if there are less completions. The list should be displayed in a nice way (for example, in a drop-down box). The list of completions must be obtained by communication with your server from ES6. Take care that the host name and port are obtained automatically (from the URL) by your JavaScript, and not hard-coded anywhere.

2. When you select one of the cities from the list, something related to that city should happen in your web app. It's up to you.

3. Make sure that your web app deals properly with special characters. The TIP file on the Wiki provides a number of test cases, which must work. This requires that you properly decode and encode URLs in your communication between the web app and the server. For example, *z%C3%BCrich* should be decoded to *zürich*. In your server code, choose a string representation such that the edit distance between any two characters (for example, *ü* and *u*) is 1. In Java, this is trivial when you use standard strings. In C++, you can use *std::wstring*.

[bitte umbl%C3%A4ttern]

4. Add one feature to your web app that makes good use of cookies.

5. Protect your web app against code injection: there should be no way that the web app can be made to execute arbitrary code by typing something in the input field or feeding a modified input file to the server. Both of these were demonstrated by example in the lecture. In particular, make sure that the queries *corrupted city*, *meteor*, *grubierF* and *santas village* work as expected.

6. Protect your server against security breaches: it should only serve files from the directory where it was started, or subdirectories of this directory.

7. Make sure that your web app looks good and feels nice.

Add your code to a new sub-directory *sheet-07* 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.

Why do we perceive ourselves as “spiritual” when, in fact, we are so machine-like inside? And why do we perceive matter as solid when, in fact, it’s mostly energy and empty space?