Assignment 2

Android, HTTP and HTML

Deadline: Monday, 9th November 2015, 9:00 am

Task 2.1: Mensa Viewer (5 Points + *1 Points)

The student services provide a website that allows you to view the Mensa menus (`http://www.studentenwerk-aachen.de/de/gastronomie/speiseplaene.html`). In this task we want to develop a standalone Android App that allows us to view the menus of the different Mensa. Looking closer, you will find that there is a distinct website for each Mensa that always provides the current menu. To accomplish this task, you need to implement an `HTTP GET` request, parse the response and display the results using Android GUI components.

You are not allowed to use:

1. Any of the `org.apache.http.*` classes.
2. Do not use any Android component that allows displaying `HTML` or automatically formatting `HTML` (e.g., `android.text.html`, `android.webkit.WebView`, ...).
3. Any 3rd-party libraries.

Follow this guideline:

1. Create a new Android application
2. Create a class that allows you to fetch an `HTML` website using a `HTTP GET` request. (See * at the bottom)
   2.1. React properly when you can't reach the server or it takes more than 15 seconds to fetch the resource.
3. Create a class that is capable of parsing the Mensa `HTML` pages to an internal representation.
   3.1. You may use any technique that feels right for you, e.g., `XML` parsing, `XPath`, String search, Regular Expressions, ... be creative, but you may not use 3rd party libraries.
4. Use the internal representation to create Android GUI components to display the Mensa menus visually appealing.
   4.1. Check out the different GUI components and layouts and choose appropriate ones.
5. Make sure you application reacts properly to Android life-cycle changes.
   5.1. E.g., rotating the view, closing the App, ...

Your App should be capable to display at least the menus of the Mensa Ahornstraße and Mensa Vita. As always, we expect you to be able to explain your code in detail. Furthermore, your code should be well structured and documented.

Teaching assistants:
Jan Rüth: `rueth@comsys.rwth-aachen.de`
Jó Ágila Bitsch: `jo.bitsch@cs.rwth-aachen.de`
Martin Serror: `martin.serror@comsys.rwth-aachen.de`

Hints:

1. Mensa Websites: There is a German and an English version the Mensa website
   `http://www.studentenwerk-aachen.de/speiseplaene/vita-w.html`
   `http://www.studentenwerk-aachen.de/speiseplaene/vita-w-en.html`
   `http://www.studentenwerk-aachen.de/speiseplaene/ahornstrasse-w.html`
   `http://www.studentenwerk-aachen.de/speiseplaene/ahornstrasse-w-en.html`

2. `HTTP`: To get a feeling how `HTTP` works: fiddle around with `netcat` and send some `GET` requests to some websites, an example:

   ```
   $ nc -c www.google.de 80
   GET / HTTP/1.0
   Host: www.google.de

   HTTP/1.0 200 OK
   Date: Mon, 20 Apr 2015 10:54:07 GMT
   Expires: -1
   Cache-Control: private, max-age=0
   Content-Type: text/html; charset=ISO-8859-1
   ...
   ```

   -c instructs netcat to send `<CR><LF>` (`\r\n`) instead of only `<LF>` (`\n`)
   -v outputs verbose information, you can also use -vv for even more output

3. `HTML` parsing: It is helpful to watch the website e.g. in Chrome using Inspect Element. However, be aware: Chrome uses headers that instruct the server to send slightly different `HTML` elements (e.g., `<tbody>` in `<table>` elements, which are not there when you request the page with minimal headers). So download the website with `netcat` (`nc -vvc www.studentenwerk-aachen.de 80`) send your `HTTP` request and save the response body to an `HTML` file and watch this using Chrome.

***Bonus:**
For 1 extra point: Do not use the Java APIs to implement the networking code, but rather use the Android NDK and implement the `HTTP GET` request in `C` or `C++` and bind your native code to your Java code via `JNI`. If your code does not work properly you can still get partial bonus points (Please note: your App should work to get the regular points, so it is probably a good idea to implement this in addition if you are uncertain that you can make it work).

Task 2.2: Compression (5 Points)

Apps on mobile phones often run over the mobile broadband uplink which is often a service that is payed for on a data volume basis. A possible way to save data is compression. The Mensa server is however not capable of compressing the message body. Write a HTTP proxy (in what ever language you like) that runs e.g. on your laptop or a dedicated server to which you relay your requests and that (if the correct header is present) compresses the message body for you. So also extend your Java code to leverage `gzip` compression to save bandwidth. You may of course also implement additional compression schemes (or even your own!).

Upload your final git commit hash to the L^{2}P room before the deadline.

Teaching assistants:
Jan Rüth: `rueth@comsys.rwth-aachen.de`
Jó Ágila Bitsch: `jo.bitsch@cs.rwth-aachen.de`
Martin Serror: `martin.serror@comsys.rwth-aachen.de`