Verteilte Systeme (D-INFK)
HS 2009
Prof. Dr. F. Mattern
Robert Adelmann (adelmann@inf.ethz.ch)
Iulia Ion (iulia.ion@inf.ethz.ch)
Matthias Kovatsch (kovatsch@inf.ethz.ch)

ETH Zürich
Institut für Pervasive Computing
Verteilte Systeme
http://www.vs.inf.ethz.ch

Exercise 2

Start: 02 October 2009
End: 23 October 2009

1 HTTP Requests

First, you should get familiar with the *Hypertext Transfer Protocol* (HTTP). In order to achieve this, implement a program that creates and sends a HTTP GET request 'by hand' using a TCP socket.

Extend your application from the first exercise: Import the module `socket` and use it to implement a function that accepts an URL, performs a GET request, and displays the response on the screen. Then add a menu entry that reads an URL and calls the function. For now, do not use any higher level libraries like `httplib`, `urllib` etc.

For example you can retrieve weather information from `http://www.google.com/ig/api?weather=zurich&hl=en`.

Hint: There is a problem with umlauts. Before printing or parsing received data, you should replace the umlauts. For instance, this code eliminates the 'ä': `variable.replace('\u00e4','ae')`. We will provide a function for convenience in the *VS Wiki*.

2 Cloud Services

As HTTP is a fairly simple protocol, it is not only popular for retrieving web pages but also for accessing services. For example we provide WoPaLo (`http://129.132.75.228:3000/`) which – among other capabilities – resolves GPS coordinates to meaningful names.

A sample request is `http://129.132.75.228:4000/places.json?p=47.37649598399737,8.544589877128601&filter=contained`. Parameter *p* represents latitude (north/south) and longitude (east/west). The optional parameter *filter=contained* specifies that you are only interested in the places you are currently in. The response is the following JSON data: `[{"name":"StreckederPolybahn", "id":20,"description":"...","page_url":"http://129.132.75.228:3000/places/19"}]`.

For the exercises, we also offer two other services at `http://129.132.75.228:8182`.
Under `/loudness` you can post a WAV-file and retrieve its loudness as number between 0.0 (silence)

and 1.0 (loud). The path `/imagedelta` implements a simple motion detector by comparing two posted images. Their difference will be returned as decimal between 0.0 (identical) and 1.0 (different).

For this part you can use the higher level libraries mentioned above. Provide entries in your application's menu that access our cloud services:

- When the first menu entry is selected, transfer your current GPS coordinates via GET to the WoPaLo service and parse the received JSON data. You can use the JSON module provided at `http://www.mobilepythonbook.org/`. When done, display the results on the screen. If you have problems using GPS, e.g. indoors, you can send dummy locations. Be aware of the umlaut problem and replace them before calling the JSON parser.
- Secondly, record a sound file (module *audio*), post it to the loudness service, and display the response. In the header, you must set the *Content-Type* to *audio/x-wav* and append the audio file in the body part.
- Finally, take two photos, post them as *multipart/form-data* to the imagedelta service, and display their difference. Information about multipart data can be found at `http://wiki.forum.nokia.com/index.php/How_to_upload_a_file_to_server_with_multipart/form-data`.

Hints:

- To see how a correct HTTP request looks like, you can use a network sniffer like Wireshark (`http://www.wireshark.org`). Query the services with your browser while capturing the network traffic. Then filter for 'http' and have a look at the listed packets.
- The httplib has a debug mode which provides more information. Activate it with `httplib.HTTPConnection.debuglevel=1`.

3 Twitter

Many well-known websites provide APIs that follow the architecture style introduced above. For this exercise, have a look at the one offered by Twitter. We already created a Twitter account that can be used:
Username: *VSETH2009* – Password: *zurich2009*
The API is documented here: `http://apiwiki.twitter.com/Twitter-API-Documentation`.

Add another menu entry that posts a message entered in GUI form on Twitter. For this, *basic access authentication* is required. Add the *Authorization: Basic ...* field to the header. Its value can be created with the `encodestring()` function from the module `base64`.

Hint: `encodestring()` adds a newline at the end which must be removed:
`encodestring('VSETH2009:zurich2009').replace('\n','')`.

4 Mashup

All previous exercises can now be combined to create a simple mashup application. Use your phone's sensors to retrieve as much information about your current situation as possible. Refine and enrich that information using cloud services like those mentioned before and create a simple message that is then posted on Twitter.

For example: Retrieve your current GPS coordinate (lat. and long.), use the WoPaLo service to resolve the coordinates to a meaningful name (e.g. 'CAB Building') and post a sentence like „I'm currently at CAB Building.“ on Twitter.

A more advanced example would be to create sentences like „I'm currently at CAB Building together with Mark and Elisa, it's pretty loud and I have not moved for two hours.“

Hint: Use Bluetooth inquiries to detect your friends.