Assignment 2

Start:	October 14, 2013
End:	October 28, 2013

Objectives

In this assignment you will learn how to develop distributed Web applications using the two different paradigms that you have seen in the lecture: *REST* and *WS-**. You will make use of these paradigms when implementing a mobile phone application that gathers data from services provided by wireless sensor nodes (SunSPOTs[1]) over the Web. The SunSPOTs expose their services (e.g., temperature sensor, ambient light sensor, etc.) through two different interfaces: REST-based and WS*-based (see Figure 1).

- **Representational State Transfer (REST)** is a style of software architecture for implementing Resource-Oriented Architectures (ROAs). HTTP (1.1)[2], the application protocol of the Web, represents an implementation of the REST principles. Distributed RESTful applications can be developed using the HTTP protocol as a universal interface for interacting with resources on the Web. Such applications make use of HTTP verbs (GET, POST, PUT, DELETE, etc.) and mechanisms (e.g., URIs and header options). Furthermore, REST defines how to serve different formats (e.g., HTML, JSON, or XML) for a given resource depending on the clients needs ("content negotiation").

- **WS-* services**, sometimes called "Big Web services," describe a set of XML-based standards (e.g., WSDL, SOAP, and UDDI) that can be used to implement Service-Oriented Architectures (SOAs). Rather than using HTTP as an application protocol, WS-* services use it as a transport protocol and define a number of additional layers to encapsulate distributed services.

With this assignment you can gain 10 points out of the total 45. The exercises marked with a ⊙ are necessary to meet the minimum requirements ("save-point").

1 Experimenting with RESTful Web Services (2 Points, ⊙)

The SunSPOTs *Spot1* and *Spot2* deliver the sensor values through a RESTful interface. Open your browser and navigate to `http://vslab.inf.ethz.ch:8081/sunspots/`. From there you can explore the HTML representation of the RESTful SunSPOTs. Browse and experiment with the SunSPOTs *Spot1* and *Spot2* and look at the sensor values they offer through this Web interface (e.g., temperature, light, and acceleration). You could also use the Spots' actuators to switch on or off their LEDs and choose colors. Your task is to write an Android application that requests the temperature measurement from *Spot1* and displays the value on the screen. Implement each sub-task in a new method and provide a button for each sub-task.

a) Create a new Android project called `vs-`*`nethz`*`-rest`. Set the application name to `VS `*`nethz`*` REST` and the package name to `ch.ethz.inf.vs.android.`*`nethz`*`.rest` (where *nethz* is the group leader's nETHZ account name). Your Activity will have to provide four buttons for the user.

[1] `http://sunspotworld.com/`
[2] `http://http://tools.ietf.org/html/rfc2616`

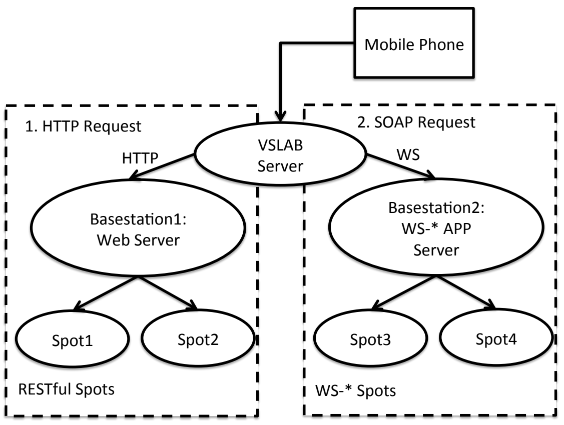

Figure 1: System setup for Assignment 2: Sensor nodes can be accessed either directly through HTTP or using SOAP messages.

b) RESTful Web services can be invoked using the HTTP protocol. In this assignment, we start by making a "raw" HTTP request **without the use of any external library**. Open a TCP connection to *vslab.inf.ethz.ch* (port 8081) and make an HTTP GET request to obtain the temperature information of *Spot1*. Display the raw response on the screen.
Hints:

- Use the `Socket` class in the `java.net` package. To send and receive data use the `getInputStream()` and `getOutputStream()` methods of the `Socket` class and read and write to the corresponding `InputStream` and `OutputStream`.
- For the HTTP protocol have a look at `http://www.elektronik-kompendium.de/sites/net/0902231.htm` or google for other how-tos. Do not mind to look directly into the standard (RFC 2616).
- Take care to correctly implement the HTTP protocol: HTTP headers are required to have a *carriage return and newline* at the end of each line. *println(...)*, however, uses the *line.separator* property, which is just a *newline* for Android. Also, remember to *flush* when using `PrintWriter`.

c) Several different libraries can be used to create HTTP calls. Now use the Apache HTTP Client library (`import org.apache.http.*` or another HTTP Client/REST library of your choice) for sending the HTTP request to get the temperature resource from *Spot1*. Display the raw response on the screen.

d) So far, we only got HTML responses back from the SunSPOTs. As mentioned before, a RESTful service can offer several representations of the same resource. To get a different representation that is more appropriate for machine-to-machine communication than HTML, we use the HTTP content negotiation mechanism: Set the `Accept` header of your HTTP request to `application/json`. Instead of an HTML page, the Web server will now return a JSON object containing the temperature information. JSON is a lightweight version of XML, which is often used in Web mashups and RESTful interfaces because it can directly be evaluated as JavaScript. Display the raw JSON response.

e) Parse the retrieved JSON response to extract the temperature value. Display only this value on the

screen.
Hints: You can find several libraries on the Web to parse JSON for many languages[3]. Android already includes `org.json.*` so you can directly instantiate and work with these JSON classes (i.e., `org.json.JSONObject`).

2 Experimenting with WS-* Web Services (2 Points, ⊙)

While *Spot1* and *Spot2* offer a RESTful interface, *Spot3* and *Spot4* can be accessed using their WS-* Web Service interface. Open your browser and navigating to: `http://vslab.inf.ethz.ch:8080/SunSPOTWebServices/SunSPOTWebservice`. From there you can access the WSDL (Web Services Description Language) description of the offered functionality. Have a closer look at the WSDL interface and try to understand its content and what it provides as you did for the RESTful version. Finally, use the HTML interface at `http://vslab.inf.ethz.ch:8080/SunSPOTWebServices/SunSPOTWebservice?Tester` to test the Web Service in your browser.
Hint: To fully understand the WSDL description, also look at the *schema* specified in the WSDL file.

a) WS-* Web services can be invoked by first getting their WSDL and then sending the corresponding SOAP messages to the Web service endpoint. Describe step-by-step how you would proceed to invoke a Web service using the `java.net` library only, as you did for RESTful Web Services. Include this description in your report. You do not need to implement anything for this sub-task.

b) Create a new Android project called `vs-`*`nethz`*`-ws`. Set the application name to `VS `*`nethz`*` WS` and the package name to `ch.ethz.inf.vs.android.`*`nethz`*`.ws`. Use a WS-* library (e.g., the kSOAP2 library patched to work on Android[4] or another WS-* library of your choice) to make WS-* calls from the Android device. Your Activity will provide two buttons.

c) Implement a button to get the temperature data from *Spot3* through WS-* and display the value on the screen. You can use the SOAP object to extract the value from the response.

d) When using WS-* services, messages are represented using XML “over the wire” before they are unmarshalled at the client side. Find out how to get this raw SOAP response and implement a button to display the response as raw XML on your Android phone.

e) Use an XML library to extract the temperature value from the raw XML response. Display it along with the XML when the button of the previous sub-task is clicked. **Hints:**

   - Check out `http://code.google.com/p/ksoap2-android/wiki/Links`.
   - To display the raw SOAP response, you will have to enable the *debug* mode in `org.ksoap2.transport.AndroidHttpTransport`.

3 Cloud Services (1 Point)

In this task, you shall visualize the data that you retrieved from the sensor nodes. For this, you use an external service to provide a graphical representation of the retrieved data. To access the sensors, you may choose between the REST and the WS-* API.

a) For this task, extend either your `vs-`*`nethz`*`-rest` or `vs-`*`nethz`*`-ws` project with an additional button for this task.

[3] `http://json.org`
[4] `http://code.google.com/p/ksoap2-android/`

b) Use the Google Chart Tools Image charts[5] to visualize the sensor measurements you retrieved with your phone. Pick your favorite visualization scheme and send the received values to the service. Show the resulting image on the display.

4 Your Phone as a Server (3 Points)

Similar to the SunSPOT sensor node, your phone can also provide RESTful Web Services for its sensing and actuation functionality. Your task is to make two sensors and two actuators available through a REST server running on your mobile phone.

a) Create a new Android project called `vs-`*`nethz`*`-server`. Set the application name to `VS` *`nethz`* `Server` and the package name to `ch.ethz.inf.vs.android.`*`nethz`*`.server`.

b) Implement the server in a similar manner to the `Socket` client in Task 1, only now make use of `ServerSocket`. Wait for and `accept` incoming connections and then handle the requests (e.g., to GET the orientation or to PUT a vibration pattern) according to the HTTP protocol. You will have to pick a port greater than 1024 (e.g., 8081).
Hints:

- Reuse your implementation to access sensors and actuators from Assignment 1, Task 1.
- Your server does not have to be RFC-compliant. Implement a minimal version that works with your browser (e.g., most header options can be ignored).

c) Run the Android Wi-Fi Hotspot app and connect your laptop to the phone. Use your browser on the laptop to test the functionality of your Android Web server.

d) Add multi-threading to your Web server: The server should be able to accept and answer requests form several clients simultaneously.

e) Describe your chosen architecture and control flow to fulfill client requests in the report, even if you might not be able to get the server running.

5 Report (2 Points, $\odot$)

As part of the assignments, you should produce a short report **(1-2 pages)**. Please write about the design and implementation questions of your applications and motivate any choices you have made during the process. Indicate any problems you have encountered during the development. You can include code snippets to explain particular ideas and we encourage you to highlight bits you are especially proud of (or do not like at all). If you provide additional features for your Android Web server, we expect you to introduce your enhancements and evaluate their benefits. A template for your reports with more specifics and pre-defined sectioning will be provided on the course Web site.

[5] `https://developers.google.com/chart/image/`

Deliverables

The following deliverables have to be submitted by **9:00am, 28 October 2013**:

1. **code.zip** You should create a zip file containing the Eclipse projects created in this assignment. The projects should have been tested both on the mobile phone and on the emulator. The code must compile on our machines as well, so always use relative paths if you add external libraries to your project. Do not forget to include those libraries in the zip file. Please use UTF-8 encoding for your documents and avoid special characters like umlauts.

2. **report.pdf** The report in **PDF** format.

Submission

Code and report must be uploaded through:
`https://www.vs.inf.ethz.ch/edu/vs/submissions/`
The group leader can upload the files, and other group members have to verify in the online system that they agree with the submission. Use your nethz accounts to log in. The submission script will not allow you to submit any part of this exercise after the deadline. However, you can re-submit as many times as you like until the deadline.