Distributed Systems – Assignment 2

Iulia Ion
iion@inf.ethz.ch
Agenda

- Tasks
- Background info
  - REST
  - Web Services (WS-*)
- Detailed tasks
- Submission
The Exercise

- **Objectives**
  - Use the phone to connect to Wireless Sensor Networks (WSN)
  - Learn and compare two different technologies for accessing sensor data over the Web: REST and WS-*
  - Make your phone part of a WSN by exporting sensed values through a Web Server
  - Use cloud services to visualize sensor data

- **Dates:**
  - Exercise begins: **Now (October 15, 2010)**
  - Exercise is due: **9:00am, October 29, 2010**
The Tasks

- Assignment is divided up into 5 tasks
  - **First Task**: Experimenting with RESTful Web Services
    - Create an HTTP request “by hand”.
    - Connect to Sun SPOTS using REST and retrieve the temperature value.
  - **Second Task**: Experimenting with WS-* Web Services
    - Connect to Sun SPOTS using WS-* and retrieve the temperature value.
  - **Third Task**: Assessing Web Service Technologies
    - Compare REST and WS-* and answer a form (**individual**).
  - **Fourth Task**: Cloud Services
    - Visualize retrieved measurements using the Google Visualization API.
  - **Fifth Task**: Your Phone as a Server
    - Implement your Android Web Server that exports phone sensor values.
    - Write a **report** on your Web Server architecture (**individual**).
  - **Feedback Form**: Workload and technology study (anonymous).
Background

- REST and WS-* technologies
Resource Oriented Embedded Devices

REST in a Nutshell

- Resources
- Their names (URIs).
- Their representations (JSON, XHTML).
- Links between them.
- A uniform interface (HTTP).
Resource Design

- Thanks to their atomicity services on embedded devices are quite adapted to Resource-Oriented Architectures.
- **Resources:**
  - **Spots:** [http://vswot.inf.ethz.ch:8081/sunspots/](http://vswot.inf.ethz.ch:8081/sunspots/)
  - **Spot 2:** [http://vswot.inf.ethz.ch:8081/sunspots/Spot2](http://vswot.inf.ethz.ch:8081/sunspots/Spot2)
Representation Design

- HTML as default, ideal for browsing
- JSON, ideal for parsing:

```json
[
  {"name": "spot 1", "location": "kitchen", "id": 1299},
  {"name": "spot 2", "location": "living room", "id": 1288},
  {"name": "spot 3", "location": "...", "id": 1812}
]
```
Uniform Interface

- **HTTP Verbs: What do you want to do?**
  - **GET:** retrieve the representation of a resource:
    - Current consumption of an electricity sensor.
  - **PUT:** update a resource:
    - Turn a device on/off.
  - **DELETE:** remove a resource:
    - Delete a threshold on a sensor, shut a device down.
  - **POST:** create a subordinate:
    - Create a new feed to trace the location of a tagged object.

- **HTTP Headers: What type of data are you sending me/requesting?**
  - **Metadata:**
    - Content type: application/json, text/html, etc.
  - **Status:**
    - 200 OK, 201 Create, 400 Bad Request, 401 Unauthorized
WS-* Technologies

- Web Service (WS-*) technologies were intended to solve three main problems
  1. Interoperability → different OSs and programming languages
  2. Firewall traversal → standard HTTP port
  3. Complexity → lower learning curve than previous technologies, such as RMI and CORBA
What are Web Services?

- An application component that:
  - Communicates via open protocols (HTTP, SMTP, etc)
  - Processes XML messages framed using SOAP
  - Describes messages using XML Schema
  - Provides an endpoint description using WSDL
  - Can be discovered using UDDI
Terminology

- XML – eXtensible Markup Language
- JSON – JavaScript Object Notation
- SOAP – Simple Object Access Protocol (communication standard)
- UDDI – Universal Description, Discovery and Integration specification (mechanisms to register and locate WS based applications)
- WSDL – Web Services Description Language (used to describe the services offered)
Web Services

- Services are defined using six major elements:
  - **types**: provides data type definitions used to describe the messages exchanged.
  - **message**: represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
  - **portType**: a set of abstract operations. Each operation refers to an input message and output messages.
  - **binding**: specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
  - **port**: specifies an address for a binding, thus defining a single communication endpoint.
  - **service**: is used to aggregate a set of related ports.
Links:

- SOAP message example: http://www.w3schools.com/soap/soap_example.asp
- WSDL: http://www.w3.org/TR/wsdl
- Web Service tutorial: http://www.w3schools.com/webservices/ws_intro.asp
Tasks

Assignment 2

Start: 15 October 2010
End: 28 October 2010

In this assignment you will learn to develop distributed Web applications, called Web Services¹, using two different paradigms: REST and WS-*.

The 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, is an implementation of the REST principles. Distributed RESTful applications can be developed using the HTTP protocol using a universal interface for interacting with services (called “resource”) on the Web. Such applications make use of HTTP verbs (GET, POST, PUT, DELETE, etc.) and mechanisms (e.g., URLs, HTTP Content Negotiation). Furthermore, REST defines how to serve different formats (e.g., HTML, JSON, XML) for a given resource depending on the clients needs.

WS-* services, sometimes called “Big Web services”, describe a set of XML-based standards (e.g., WSDL, SOAP, 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.

In this assignment you will implement a mobile phone application that gathers data from services provided over the Web by Wireless Sensor Nodes (Sun SPOT³). The Sun SPOT's expose their services (e.g., temperature, light, etc.) through two different interfaces: REST based and WS-* based. Figure 1 illustrates the setup.

1. HTTP Request to WS
2. SOAP Request to WS

![Diagram of distributed systems assignment](image.png)
Marks

- Tasks 1-3 and reports: 4.0
- Tasks 1-5 and reports: 5.0
- All Tasks and reports: up to 6.0

Partially solved exercises are marked individually.
Submission

https://www.vs.inf.ethz.ch/edu/vs/submissions/

- You should submit the code and report through the above URL
- NETHZ Login
- Note:
  - Max file sizes
  - File types
  - Multiple submissions possible!

- Assignment form (individual, specify your NETHZ Login):
  https://spreadsheets.google.com/viewform?hl=en&formkey=dFQ4WGM2Tkt2T0s0dWlCOXAwVXE0MUE6MQ
- Feedback form (individual, anonymous):
  https://spreadsheets1.google.com/viewform?hl=en&formkey=dGQzOVpGZGRBRW9iemNkb0hQbDNqZlE6MQ