TECHNISCHE UNIVERSITÄT DARMSTADT

Prof. Dr. Max Mühlhäuser
Dr. Immanuel Schweizer

Dr. Benedikt Schmidt
Dipl.-Wirtsch.-Inform. Axel Schulz

TELEKOOPERATION
Fachbereich Informatik
Hochschulstr. 10
64289 Darmstadt

TK1: Distributed Systems - Programming & Algorithms

3. Programming Assignment
Submission Date: 8.12.2013

Task 1 Shopping Cart using WebServices (20 P.)

Implement a WebService, which follows the fundamental functionality of a shopping cart system.

The server has to provide the following functionality:

- The server has to manage the shopping cart for every client. Each shopping cart is stored using a uniqueID for each client. No complex session management is needed for the clients.
- The server has to provide Information about at least three different products (“Name”, “Price”, “Available Amount”)
- The server has to manage the available amounts for each product

Each client shall provide the following functionality:

- The client has to provide a user interface, which shows the available products with prices and the available amount for each product
- The user is able to buy products. The user can send his order using his shopping cart.
- The user gets a return message, if a product is not available, because another order might be received beforehand.

Your task is to implement a SOAP WebService which provides the described functionality. Furthermore, implement a RESTful-Service, which provides the same functionality. Provide two clients at which each client uses exactly one of the two provided WebServices.

Use JAX-WS and JAX-RS (and its reference implementation Jersey) as frameworks for the WebServices.

Hints:

- Use the HTTP-Server for service deployment that is provided by the Java JDK (not included in the JRE).
  - Example for REST startup:

```
HttpServer server = HttpServerFactory.create("http://localhost:8080/rest");
server.start();
```

- Example for SOAP startup:

```
Endpoint.publish( "http://localhost:8090/services", new MyWebServices() );
```

- Tutorials:
  - http://openbook.galileocomputing.de/java7/1507_13_001.html#dodtp82d1ec9d-ccf4-456f-8af9-ebd4bb3c87b4 (German)
  - http://www.mkyong.com/webservices/jax-ws/jax-ws-hello-world-example/ (English)
  - http://theopentutorials.com/examples/java-ee/jax-rs/create-a-simple-restful-web-service-using-jersey-jax-rs/ (English)

- Break down your solution into two separate projects:
  - One project („Clients“) contains both clients.
  - The other project („Services“) contains the service implementation.

  The client project is not allowed to depend on the service source code (e.g., by importing its interfaces).

- Provide an Ant script („build.xml“) within the root folder of each project:
  - The default target in the „Services“ project starts a REST service as well as a SOAP service.
  - The default target in the „Clients“ projects starts one client for each service instance. In case of SOAP, the client script has to download the WSDL file and generate the stubs at first.

- Use the JDK tool wsimport for client stub generation.

- Start the REST server on port 8080. Start the SOAP server on port 8090.

Grading:

- Basic Requirements: Working ANT Script, separate client and server project
- REST Service with all functions (6 Points)
- Working SOAP Service with all functions (6 Points)
- Client for REST with all functions (4 Points)
- Client for SOAP with all functions (4 Points)