Creating Web Services with J2EE + Apache Tomcat + Apache Axis

Tuukka Ruotsalo

Contents

- WSDL reminder
- Java Beans and serialization
- Design issues
- Tools for automated web service development in Apache Axis

WSDL reminder

- WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.
- The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.
- WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate
- At the moment only supported are SOAP 1.1, MIME, HTTP get/post

Abstracts and Concretes

- WSDL discusses how to describe the different parts that comprise a Web service:
- Abstract definitions
  - Types– a container for data type definitions using some type system (such as XSD).
  - Message– = types.
  - Operation– an abstract description of an action supported by the service.
  - Interface –an abstract set of operations supported by one or more endpoints.
- Concrete definitions
  - Binding– a concrete protocol and data format specification for a particular port type.
  - endpoint – a single endpoint defined as a combination of a binding and a network address.
  - Service– a collection of related endpoints.

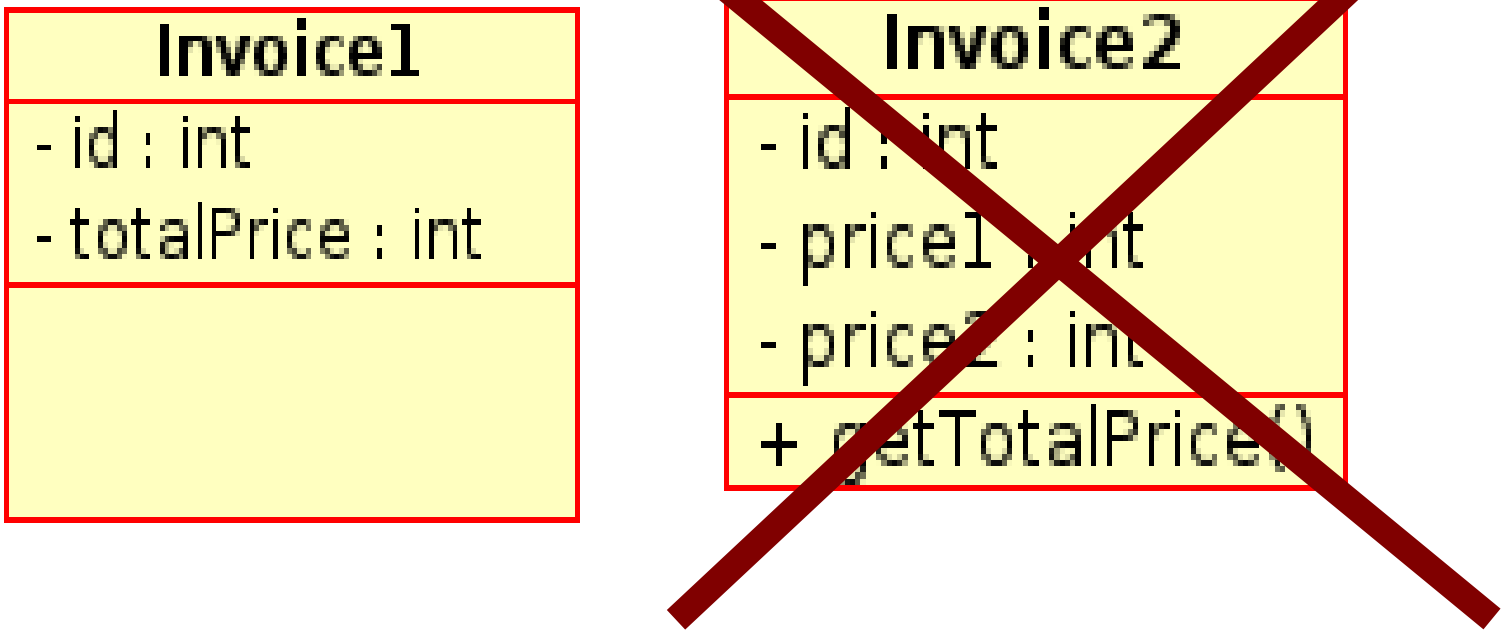
Serialization

- In order to make data transport possible the data must be somehow encoded and decoded by the applications
- In Web Services world data-structures in programming languages are serialized to XML
- Data-structures correspond to programming language data-types through XML Schema datatypes
- In Axis + Java Serialization is done by Serializers (Preferrably with Java BeanSerializer)
- Design smart and you'll save the effort of writing custom serializers (in case you'll need just RPC)

Java Beans and Serialization

- The bean should use the getFoo/setFoo patterns for properties, as defined in the beans specification.
- All of the bean's persistent state should be accessible via getFoo/setFoo properties. This allows external tools to save and restore the beans state using the property accessor methods.
- If the bean is prepared in the future to use automatic serialization then it must inherit from the Serializable interface.

Serialization reminder



JavaBeans and WS

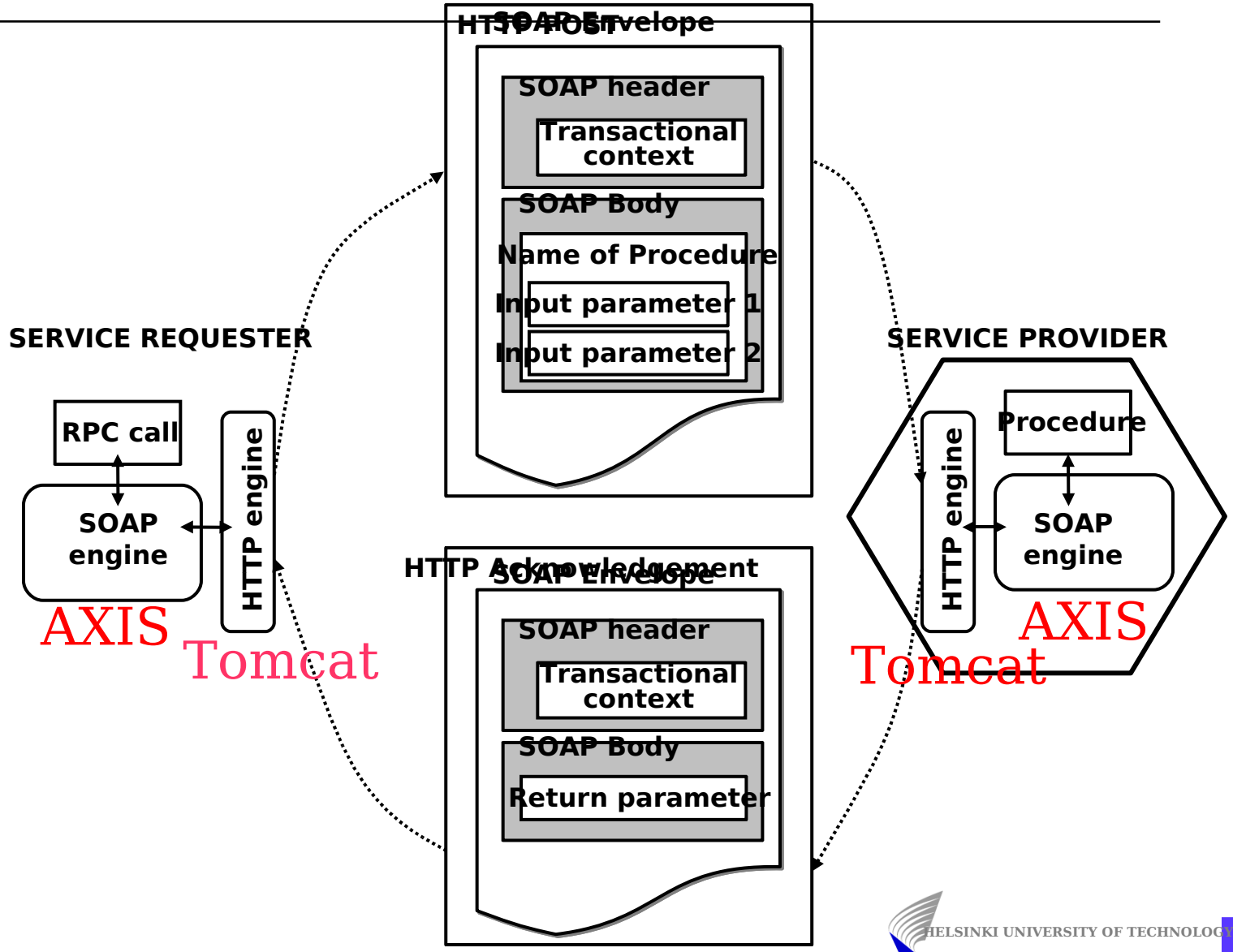
- In WS world a few notices that will make things much easier
  - Getters and setters should be public in web service world
  - Lists as YourObject[], since java 1.4. does not support typed lists (1.5 does but Axis does not!)
  - Native types instead of Java specific wrapper classes(e.g. double instead of Double)
- With these assumptions BeanSerializer will do everything for you!!

Design issues

- Create your service architecture (top down)
  - Classes (Beans) for data model
  - Service interfaces for web-service
  - Implementation of the service interface
  - Publish as WSDL
  - Create stubs for client
  - Create Client

Automated Tools

- Apache Axis provides useful tools for web service generation
  - Java2WSDL
    - Generates WSDL description of the java classes
    - Generates deployment/undeployment descriptor
  - WSDL2Java
    - Generates stubs from the WSDL description
- Note: Be aware of the parameters of the scripts. Remember that part of your code act as server and part of it as client.


SERVICE REQUESTER
RPC call
SOAP engine
HTTP engine
AXIS
Tomcat
HTTP POST
SOAP Envelope
SOAP header
Transactional context
SOAP Body
Name of Procedure
Input parameter 1
Input parameter 2
HTTP Acknowledgement
SOAP Envelope
SOAP header
Transactional context
SOAP Body
Return parameter
SERVICE PROVIDER
HTTP engine
Procedure
SOAP engine
AXIS
Tomcat