Web Services

Web Services Layer

Discovery: UDDI
- Find the description of Web Services

Description: WSDL
- Describe a Web Service

Packaging: SOAP
- Encode the data in an agreed format (marshalling, serialization)

Transport (e.g. HTTP, SMTP)
- transport of data between applications

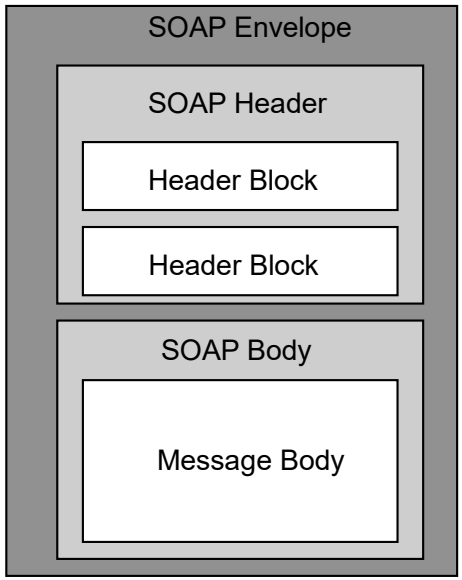
Network (e.g. TCP/IP)
- Communication between machines / processes in the network

What is SOAP?

A mechanism to represent XML-messages
- originally: Simple Object Access Protocol
- Despite the original meaning of this acronym: not really object-oriented

Features of SOAP
- Message format
- Extensibility
- A Convention to describe RPCs (Remote Procedure Calls)
- Different bindings to transport-protocols

Structure of a SOAP Message



Parts of a SOAP Message (1)

SOAP Envelope

- Is the document-element and acts like a bracket surrounding the rest of the message
- Contains header and body

SOAP Header

- Is optional
- Is used for extensions, e.g.
  - Transaction-context
  - Security information
- Elements of the header might be ignored or not
  - To enforce the processing, the attribute mustUnderstand is set to 1

Parts of a SOAP Message (2)

SOAP Body

- Contains the actual payload
  - Function call
  - Error messages
  - Return value
- Common encoding: RPC-style
  - The body contains an Element that has the name of the function to be called
  - Within this element there exists for each parameter a child-element

Transport

SOAP-Messages can be transmitted using different transport mechanisms

- HTTP is the common transport mechanisms and is emphasized in the SOAP standard
- HTTP/S for encrypted messages
- SMTP for asynchronous function calls

Using SOAP

How do I exchange a SOAP-message?

- Who is responsible for encoding the message in XML?
- Who receives the message and initiates the function call?
- How do I get the reply?

Solution 1: SOAP-Engine, to process the messages

- Popular: Axis2 (SOAP-Engine as a servlet or stand-alone)

Solution 2: Starting with Java Version 6 the JDK provides their own tools to provide a Web Services

- will be discussed in this lecture

Consequence

- SOAP-messages are processes by means of a particular API
- XML often invisible

Development Process

Server

- Development of a class that realizes the service
  - the class has to have a constructor without parameters
  - the class is configured by means of annotations as a Web Service
- Development of a server that makes the service available
- optional: Generation of a WSDL-file for the service (can be done for example by calling the service from a browser)

Client

- generation of Client-stubs from the WSDL-file
  - wsimport -keep http://localhost:8765/DateReverse?wsdl
- Development of a client
  - By means of the stubs we achieve access transparency when calling the offered functions

Service-Code

```
package testws1;

import java.util.Date;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style=Style.RPC)

public class TestService1 {
  public String getDate() {
    return new Date().toString();
  }
  public String reverse(String input){
      return new String ( new StringBuffer(input).reverse() );
  }
}
```


Server-Code

```
package testws1;
import javax.xml.ws.Endpoint;

public class TestServer {
  public static void main (String args[]) {
    TestService1 server = new TestService1();
    Endpoint endpoint =
      Endpoint.publish("http://localhost:8765/DateReverse", server);
    System.out.println ("Der Server ist gestartet");
  }
}
```


Client

Create a ServiceLocator for the service

Use the ServiceLocator to get a stub

Using the stub we can call methods of the service like local methods

Client

```
import testws1.*;

public class TestService1Client {
    public static void main(String args[]) {
        TestService1Service service = new TestService1Service();
        TestService1 stub = service.getTestService1Port();
        System.out.println("Datum: " + stub.getDate());
        System.out.println("Hallo umgedreht: " + stub.reverse("Hallo"));
    }
}
```


Inspecting the Messages

Goal

- Show the messages that are exchanged between client and server

Simple approach

- Use in the client a particular parameter for the virtual machine
- -Dcom.sun.xml.internal.ws.transport.http.client.HttpTransportPipe.dump=true

SOAP-Message: Request

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
     <ns2:reverse xmlns:ns2="http://testws1/">
       <arg0>Hallo</arg0>
      </ns2:reverse>
   </S:Body>
</S:Envelope>
```


SOAP-Message: Response

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:reverseResponse xmlns:ns2="http://testws1/">
      <return>ollaH</return>
    </ns2:reverseResponse>
  </S:Body>
</S:Envelope>
```

SOAP-Message: Request without Parameter

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getDate xmlns:ns2="http://testws1/"/>
  </S:Body>
</S:Envelope>
```


WSDL

Web Services Description Language

Goals

- Structured description of a Web Service
- Mainly to be used by machines and not humans

Main parts

- Which operations are offered by the service?
- How can we access the service (e.g. data types, transport protocol)?
- Where is the service located (URL)?

WSDL: Structure

types

- optional definition of data types

message

- single messages and their parameters

portType (in WSDL 2.0: interface)

- grouping of messages (Request/Response)
- contains the element operation

binding

- the used transport mechanism

service

- address of the Web Service

WSDL Example (1)

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://testws1/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://testws1/" name="TestService1Service">
<types/>
<message name="reverse">
<part name="arg0" type="xsd:string"/>
</message>
<message name="reverseResponse">
<part name="return" type="xsd:string"/>
</message>
<message name="getDate"/>
<message name="getDateResponse">
<part name="return" type="xsd:string"/>
</message>
```

WSDL Example (2)

```
<portType name="TestService1">
<operation name="reverse">
<input wsam:Action="http://testws1/TestService1/reverseRequest" message="tns:reverse"/>
<output wsam:Action="http://testws1/TestService1/reverseResponse"
message="tns:reverseResponse"/>
</operation>
<operation name="getDate">
<input wsam:Action="http://testws1/TestService1/getDateRequest" message="tns:getDate"/>
<output wsam:Action="http://testws1/TestService1/getDateResponse"
message="tns:getDateResponse"/>
</operation>
</portType>
```


WSDL Example (3)

```
<binding name="TestService1PortBinding" type="tns:TestService1">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
<operation name="reverse">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal" namespace="http://testws1/"/>
</input>
<output>
<soap:body use="literal" namespace="http://testws1/"/>
</output>
</operation>
```


WSDL Example (4)

```
<operation name="getDate">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal" namespace="http://testws1/"/>
</input>
<output>
<soap:body use="literal" namespace="http://testws1/"/>
</output>
</operation>
</binding>
<service name="TestService1Service">
<port name="TestService1Port" binding="tns:TestService1PortBinding">
<soap:address location="http://localhost:8765/DateReverse"/>
</port>
</service>
</definitions>
```


UDDI

Universal Description Discovery and Integration

UDDI specifies standards to publish and search Web Services

Interfaces

- for humans using a browser
- for programs using a Web Service API

Practical relevance of UDDI is much less than of SOAP and WSDL!

UDDI Data Structures

White Pages
- Information about the institution that offers the service, e.g.
  - Name
  - Telephone number
- Similar to the information in a normal phone book

Yellow Pages
- Classification according to different categories
  - Geographical
  - Domains
  - Services / Products
- Similar to the information in the “Gelben Seiten”

Green Pages
- Contain technical information
  - Offered Web Services
  - How can the be accessed
  - Might contain a link to a WSDL-Document

Summary

Web Services lack some functionality that other middleware technologies can offer
- No distributed object model
- No object references

Moderate performance

Focus is on
- Interoperability
- Integration of (organizationally and technically) heterogeneous systems
- Loose coupling
- Re-use of well-known Internet protocols and software
  - Less blocking by firewalls
    - security?
- Modularity by means of the header mechanism