

Übung Netzbasierte Informationssysteme

Termin 4: Web Services Computing

Prof. Dr. Adrian Paschke

Arbeitsgruppe Corporate Semantic Web (AG-CSW)
Institut für Informatik, Freie Universität Berlin
paschke@inf.fu-berlin.de
<http://www.inf.fu-berlin.de/groups/ag-csw/>



Agenda

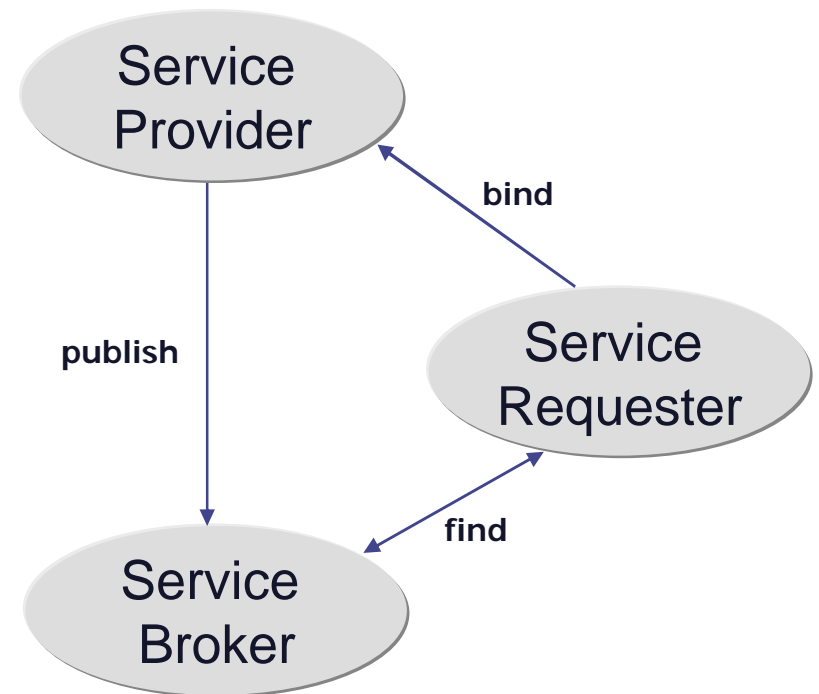
- Feedback Übungsblatt 3
- Themen des heutigen Übungsblatts:
 - Web Service Basics
 - SOA
 - SOAP
 - WSDL
 - Axis
- Übungsblatt 4

Feedback - Übungsblatt 3

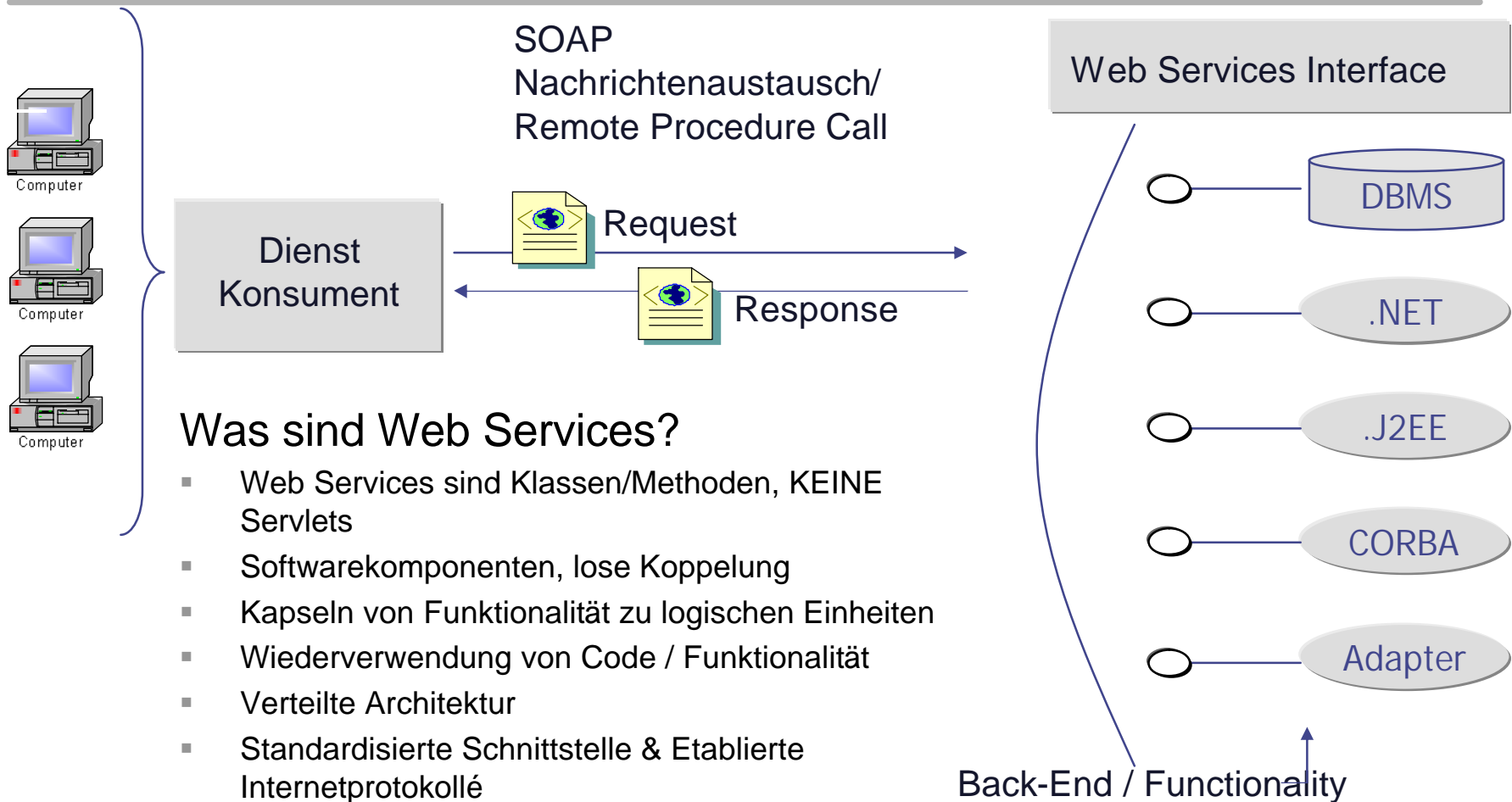
- **Lernziele des 3. Übungsblattes waren:**
 - XML Basics
 - DTD
 - Schema
 - Parser
 - eXtensible Stylesheet Language (XSL)
 - XML Path Language (XPath)
 - Extensible Stylesheet Language Transformations (XSLT)
 - XML Query Language (XQuery)
- **Wie war`s?**
 - Fehlerquellen
 - Probleme?
- **Anregungen?**

SOA – Service Oriented Architecture

- Dienstanbieter publiziert Dienstbeschreibung (WSDL) bei einem Service-Broker
- Nachfrager finden Dienst über Broker und bindet diesen dynamisch ein
- Publizieren, Suchen, Finden (z.B. über UDDI - nächstes Blatt)
- Erlaubt ad hoc Kollaboration in Wertschöpfungsnetzen

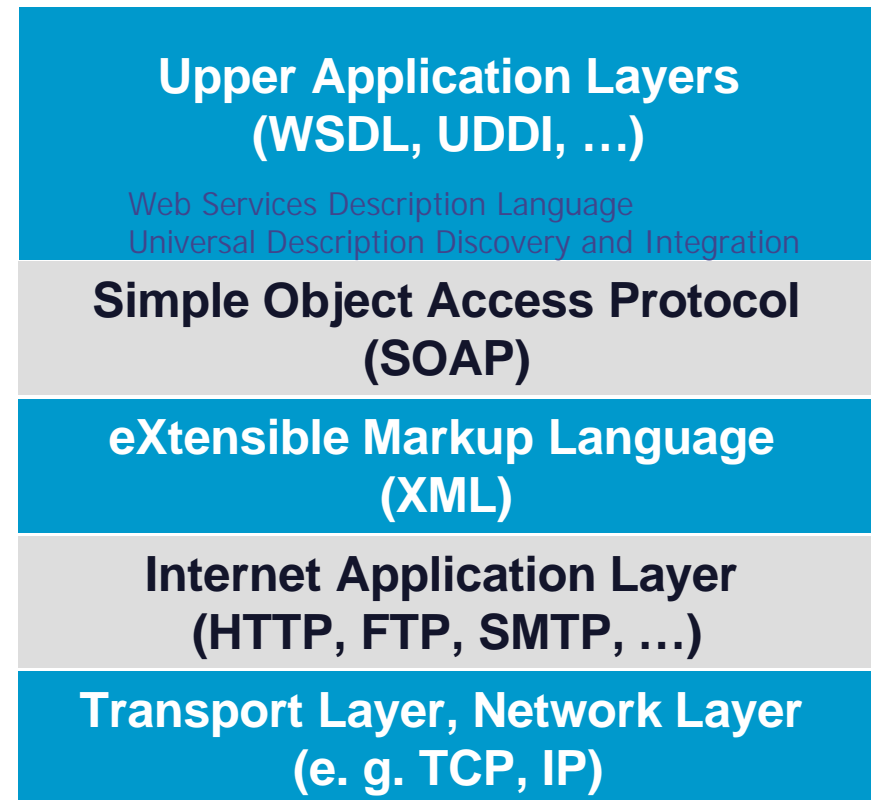


Web Services – Realisierung einer SOA



Web Services - Protokollschichten

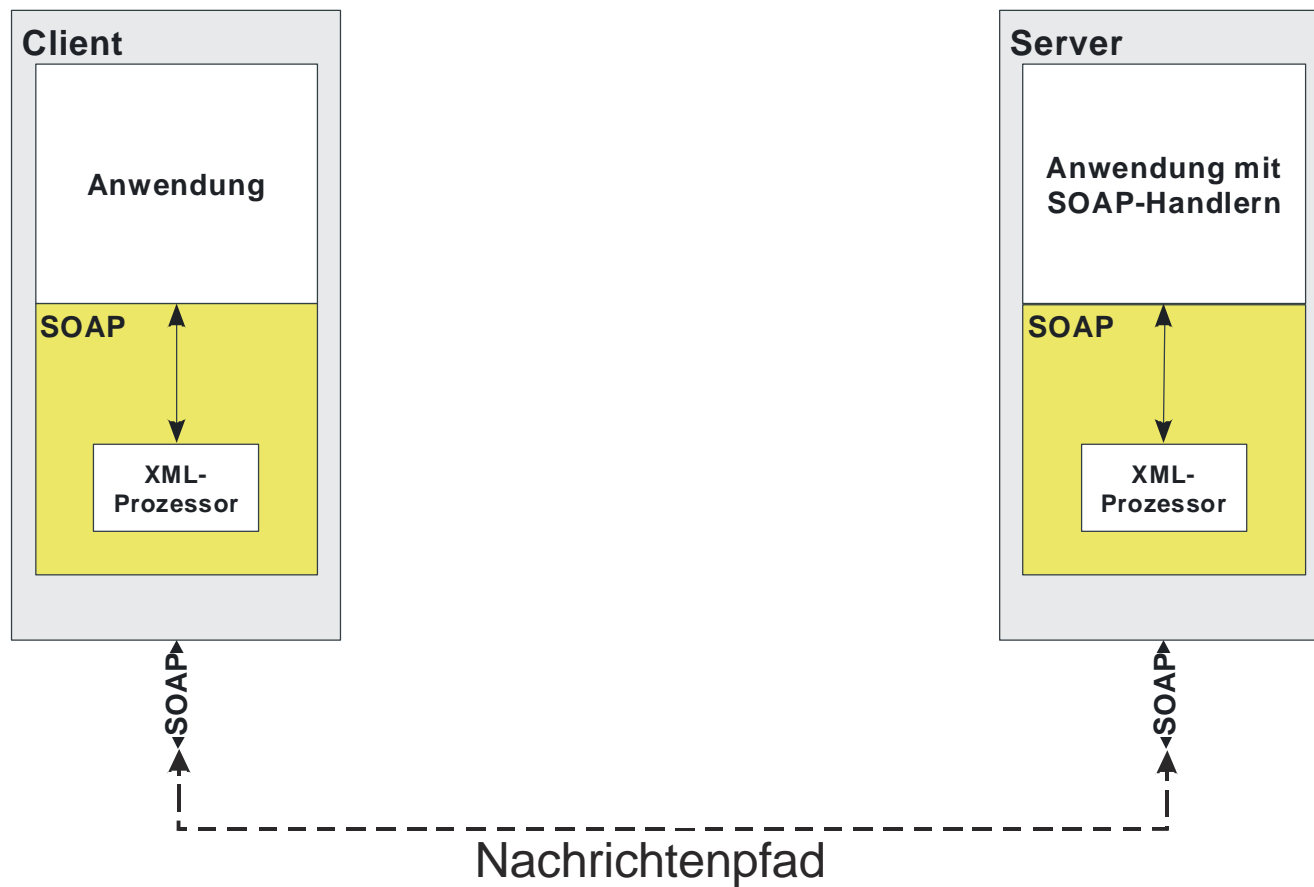
- Etablierte Basisschichten:
 - TCP/IP
 - HTTP
 - XML
- SOAP setzt darauf auf
- UDDI und WSDL wiederum nutzen SOAP
- weitere Konzepte:
 - Sicherheit
 - Authentifizierung
 - Quality of Service
 - Session Management
 - Transaktionsmanagement



SOAP - Simple Object Access Protocol

- Nachrichtenaustauschprotokoll
 - XML-basiert
 - Verbindungslos
- RPC-Zugriff oder Messaging in verteilten Systemen
- Bindung an Internet-Application-Layer wie HTTP, SMTP, FTP
- Unterstützt einfache und abstrakte Datentypen
 - keine Objektstrukturen und Referenzen (wie z.B. CORBA)
 - nur serialisierbare Datentypen
- Hoher Aufwand für Aufbereitung und Transport der Daten nötig
 - „Aufgeblähtes“ XML-Format
 - Lösung: Einsatz von XML-Parsern an den Endpunkten

SOAP - Nachrichtenübertragung



Quelle: Huemer: Web Services Übersichtsvortrag

SOAP –Nachrichtenaufbau

```
POST /cswservice/csw HTTP/1.1
Host: mysoapsver
Content-Type: text/xml; charset=utf-8
...
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=„http://schemas.xmlsoap.org/soap/envelope/“>
```

```
<SOAP-ENV:Header>
  <t:TransactionCode xmlns:t=„my-URI“ xsi:type=„xsd:int“
    mustUnderstand=„1“
    SOAP-ENV:Actor=„http://myserver/actor/TCPMonitor “>
    156533245
  </t:TransactionCode>
</SOAP-ENV:Header>
```

```
<SOAP-ENV:Body>
  <method:getNextCoord xmlns:method=„my-URI2“>
    <xCoord>15</xCoord>
    <yCoord>124</yCoord>
  </method:getNextCoord>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

HTTP Header

Envelope
Namespace
mit SOAP-
Vokabular

Header mit Zusatzinfo
z.B. Transaktionscode

Body enthält Nutzdaten
z.B. Parameter für RPC
oder SOAP-Fehler

WSDL – Web Service Description Language

- Einsatz von SOAP setzt **Kenntnis** der
 - **Parameter,**
 - **Datentypen und**
 - **Methodennamen**eines Web Services voraus.
- **Beschreibung** eines WS durch ein **WSDL-File:**
 - Wo liegt der Web Service genau ?
 - Welche Schnittstellen bietet der Service?
 - Mit welchen Protokollen kann man den Web-Service einbinden?

WSDL – Basics

- XML-basierte Sprache zur Beschreibung von WS
- Programmiersprachenunabhängig
- Entstand auf Initiative von IBM, Microsoft, Ariba
- Mittlerweile W3C Standard:
 - <http://www.w3.org/TR/wsdl>
- Pendant zur IDL von CORBA
 - Typen der Übergabeparameter
 - Typ des Rückgabewerts
 -
 - Gegensatz zu IDL: Ortsangabe, URI des Web Services

WSDL – Inhalt

- Welche Datentypen werden übermittelt?
 - `<types/>` Element
- Welche Nachrichten werden übermittelt?
 - `<message/>` Element
- Welche Operationen werden unterstützt?
 - `<portType/>` Element
- Wie werden die Nachrichten im Netz transportiert?
 - `<binding/>` Element
- Wo befindet sich der Dienst?
 - `<service/>` Element

WSDL – Struktur

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<definitions ...>
```

```
<types/>
```

Types

```
<message/><portType/>
```

Messages / PortTypes

```
<binding/>
```

Bindings

Service Interface

```
<service/>
```

Service Definition

Service Implementierung

```
</definitions>
```

WSDL Dokument

WSDL – Beispiel

Genaue Beschreibung:

<http://www.w3.org/TR/wsdl>



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>

  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>

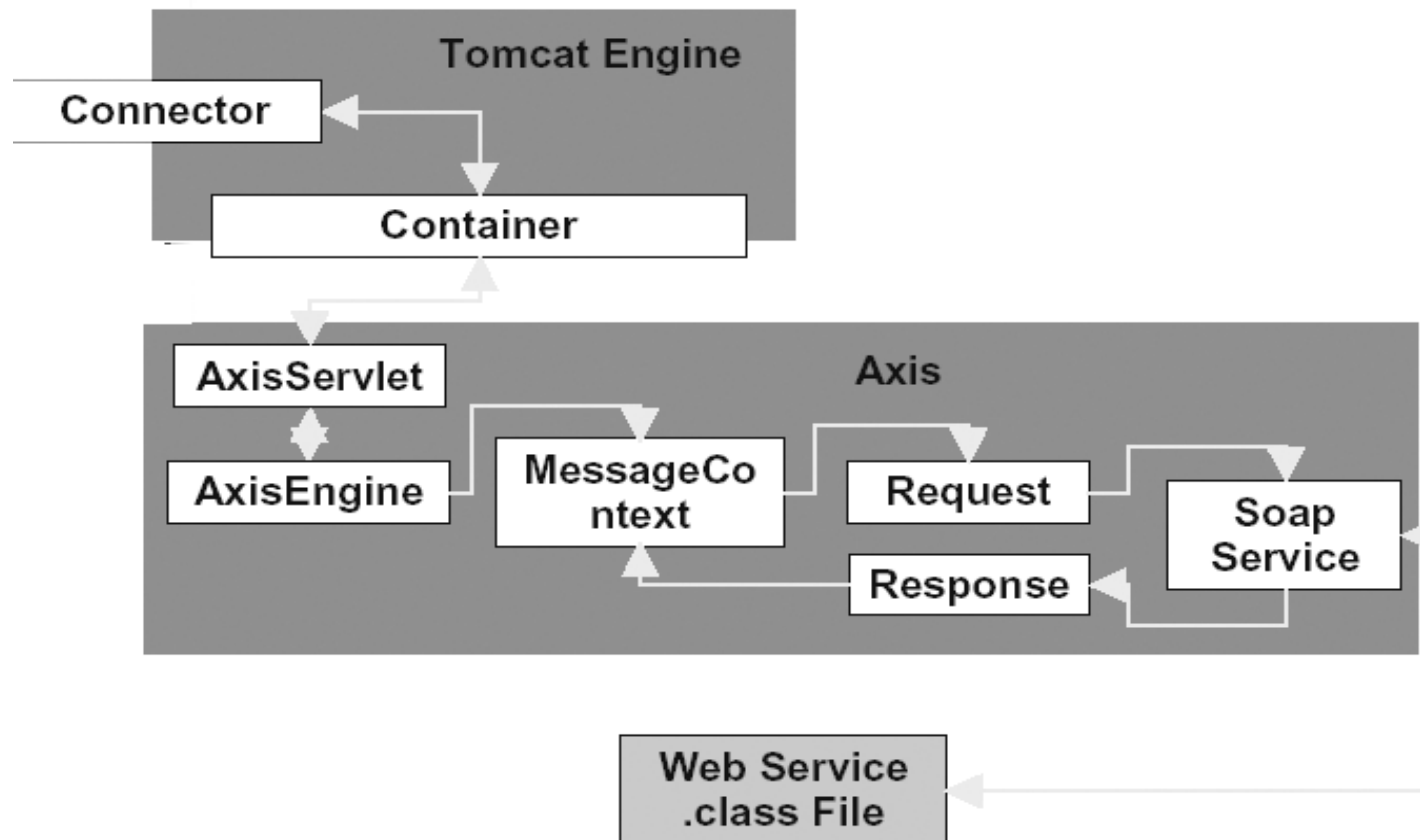
  <binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </output>
    </operation>
  </binding>

  <service name="Hello_Service">
    <documentation>WSDL File for HelloService</documentation>
    <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
        location="http://localhost:8080/soap/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>
```

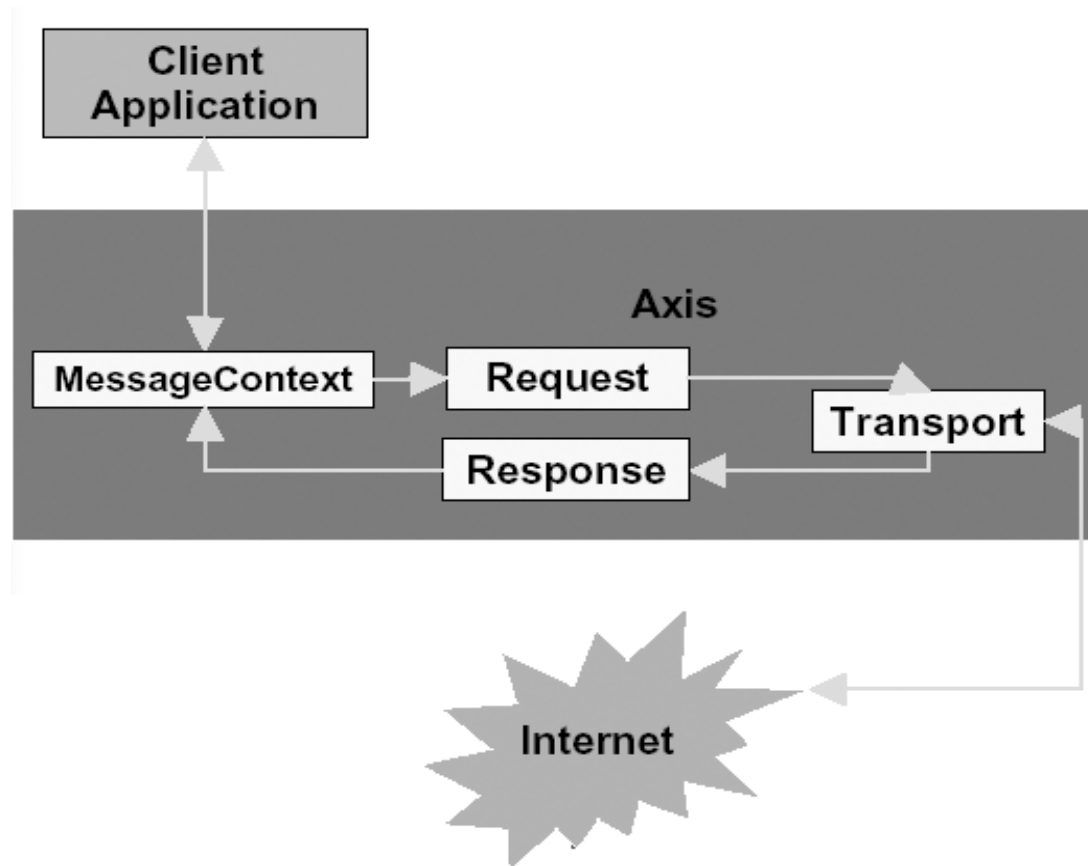
AXIS - Apache eXtensible Interaction System

- Java Open Source SOAP 1.1 Implementierung (SOAP Engine) <http://ws.apache.org/axis>
- Axis ist eine Webapplikation; Ausführung z.B. unter Tomcat
- Erlaubt sowohl RPC als auch Messaging
- Enthält zusätzlich Tools für Bearbeitung von WSDL und das Deployment von Web Services
 - AdminClient
 - java2wsdl / wsdl2java für Clients und Server
- Axis-Servlet agiert als Transport-Listener
 - nimmt HTTP Requests entgegen & extrahiert SOAP-Nachricht
 - Übergibt Nachricht der Axis Engine (Java Objekt)
 - Führt den Web Service aus
 - Sendet Engine-Antwort per HTTP-Response zurück an Client

AXIS – Aufbau: Server



AXIS – Aufbau: Client



AXIS - Deployment von WS

- **JWS** (die einfache Variante)
Source-Code ins Context-Root Z:\webapps\axis\ ... fertig
keine Features wie: Handler, Alias, ...
- Deployment des Web Service über ***.wsdd Datei (Web Service Deployment Descriptor)**
 - enthält Instruktionen zum Web Service Deployment
 - Das Axis-Tool **AdminClient** führt diese Instruktionen aus.

```
java org.apache.axis.client.AdminClient -l http://<Ihre IP>:<Ihr Port>/axis/servlet/AxisServlet <Datei.wsdd>
```

*Kompilierte Klassen in den Context kopieren!
Z.B. C:\tomcat\webapps\axis\web-inf\classes*

AXIS - Deployment von WS

Aufbau eines **Deploymentdescriptors**: deploy.wsdd

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="ServiceName" provider="java:RPC">
    <parameter name="className" value="aufgabeX.WebServiceClass"/>
    <parameter name="allowedMethods" value="*" />
  </service>
</deployment>
```

Aufbau eines **Undeploymentdescriptors**: undeploy.wsdd:

```
<undeployment name="MyName" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="ServiceName" />
</undeployment>
```

- *Hinweis: Der Service wird automatisch in der Datei server-config.wsdd im Web-Inf-Verzeichnis des Axis Contexts eingetragen.*

AXIS - Deployment von WS

- Elegante Alternative des Aufrufs von AdminClient:
 - **Apache Ant Task**

```
<!-- ===== -->
<!-- Import Axis ant task -->
<!-- ===== -->
<taskdef name="axis-admin" classname="org.apache.axis.tools.ant.axis.AdminClientTask">
  <classpath refid="project.class.path"/>
</taskdef>
```

- **Parameter nötig:**

```
<!-- Deploy properties -->
<property name="project.deploy.dir" value="${project.home.dir}/deploy"/>
<property name="webapp.target.port" value="8080"/><!-- update this -->
<property name="webapp.target.server" value="localhost"/><!-- update this -->
<property name="wsdd.file.deploy" value="${project.deploy.dir}/deployFileName.wsdd"/><!-- update this -->
<property name="wsdd.file.undeploy" value="${project.deploy.dir}/undeployFileName.wsdd"/><!-- update this -->
<property name="webapp.target.appname" value="http://${webapp.target.server}:${webapp.target.port}/axis"/>
```

AXIS - Deployment von WS

```
<!-- ===== -->
<!-- Undeploy Web Services -->
<!-- ===== -->
<target name="undeployWS" depends="compile">
  <echo>+-----+</echo>
  <echo>|                               |</echo>
  <echo>| U N D E P L O Y I N G       |</echo>
  <echo>|                               |</echo>
  <echo>+-----+</echo>

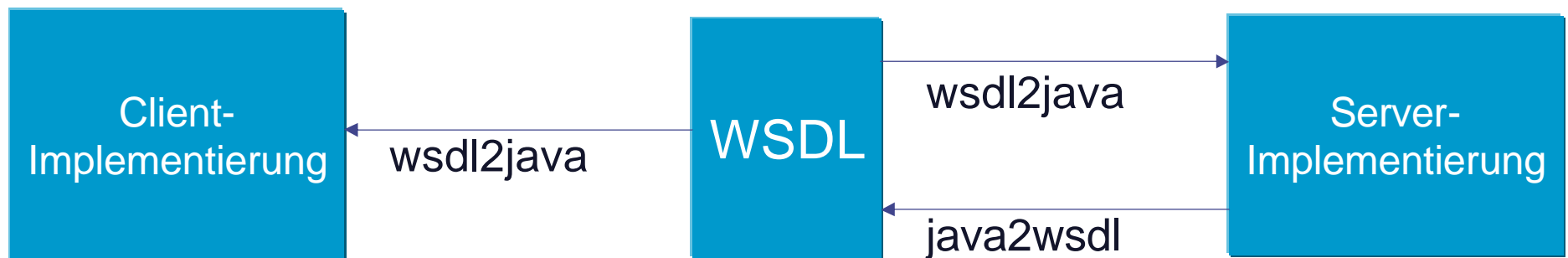
  <delete dir="${build.context.name}" />
  <axis-admin port="${webapp.target.port}"
             hostname="${webapp.target.server}"
             failonerror="true"
             debug="true"
             xmlfile="${wsdd.file.undeploy}" />
</target>

<!-- ===== -->
<!-- Deploy Web Services -->
<!-- ===== -->
<target name="deployWS" depends="undeployWS">
  <echo>+-----+</echo>
  <echo>|                               |</echo>
  <echo>| D E P L O Y I N G           |</echo>
  <echo>|                               |</echo>
  <echo>+-----+</echo>

  <copy todir="Z:\webapps\axis\WEB-INF\classes">
    <fileset dir="${project.bin.dir}" />
  </copy>
  <axis-admin port="${webapp.target.port}"
             hostname="${webapp.target.server}"
             failonerror="true"
             debug="true"
             xmlfile="${wsdd.file.deploy}" />
</target>
```

Generieren von WSDL

- Zusätzlich möglich: die Abbildung von WSDL auf eine Programmiersprache
- Apache AXIS bietet folgende Möglichkeiten:
 - Generierung von WSDL aus bestehender Webservice Implementierung
`java org.apache.axis.wsdl.Java2WSDL -o <WSDL-Datei> -l <Service-URL> -n <Namespace>`
 - Clientseitiger Code aus WSDL
`java org.apache.axis.wsdl.WSDL2Java <WSDL-Datei>`
 - Serverseitige Implementierung aus WSDL
`java org.apache.axis.wsdl.WSDL2Java -d Application -s -S true <WSDL-Datei>`



Web Services – Aufruf eines WS

```
import java.net.URL;
import javax.xml.rpc.ParameterMode;
import org.apache.axis.Constants;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

public class MeinWebServiceClient1
{
    public static void main( String[] args )
    throws Exception
    {
        if( 2 != args.length ) {
            System.out.println(
                "\nTwo parameters needed.\n" +
                " First parameter: either 'Quadrat' or 'Wurzel'\n" +
                " Second parameter: a number\n" +
                "E.g.:\n" +
                " java meinpackage.MeinWebServiceClient1 Quadrat 12.7" );
            System.exit( 1 );
        }
        String wsEndpoint = "http://localhost:8080/axis/services/MeinWebService";
        String wsMethod = "meineWebServiceMethode";
        Service service = new Service();
        Call call = (Call)service.createCall();
        call.setTargetEndpointAddress( new URL( wsEndpoint ) );
        call.setOperationName( wsMethod );
        call.addParameter( "job", Constants.XSD_STRING, ParameterMode.IN );
        call.addParameter( "x", Constants.XSD_DOUBLE, ParameterMode.IN );
        call.setReturnType( Constants.XSD_DOUBLE );
        Object ret = call.invoke( new Object[] { args[0], new Double( args[1] ) } );
        System.out.println( "\n'" + args[0] + "' von '" + args[1] + "' ist: " + ret );
    }
}
```

Übungsblatt 5

- **Lernziele**

- Web Service Basics
 - SOA
 - SOAP
 - WSDL
 - Axis

- **Termine**

- Ausgabe: 18.11.2008 Abgabe bis: 2.12.2008, 16:00 Uhr
- **2-wöchiges Übungsblatt**



Viel Spaß und Erfolg !

