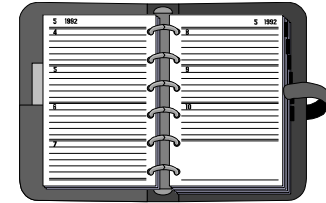


Die Technik hinter Web Services

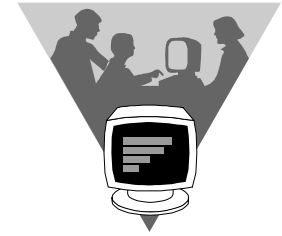
Wie baut man einen Web Service?
Was sind die technischen Details von
WSDL, SOAP und UDDI?

Themen



- ✕ Beschreibung des Beispiels
- ✕ Exkurs: XML
- ✕ Beschreibung eines Web Services mit WSDL
- ✕ Kommunikation zu Web Services mit SOAP
- ✕ Suchen und Finden von Web Services mit UDDI

Web Services : Beispiel



- Implementierung von HelloWorld als Web Service
- Verwendeter Web Service Server: Apache Axis
- Implementierung des Web Service: Java

<<Interface>> Hello	
+ sayHello(in0 : String) : String	

Entwicklungsschritte eines Web Service



- Design des Web Service mit WSDL
- Generierung der Implementierung + Bindung mittels WSDL2Java
- Implementierung der Fachlogik
- Starten der SOAP Engine/Application Server
- Installation/Deployment des Web Service
- Implementierung eines Client
- Test des Client

Generierung der WSDL (Apache Axis)

```
java org.apache.axis.wsdl.Java2WSDL  
-o MyHello.wsdl  
-l http://localhost:8080/axis/services/MyHello  
-n "urn:MySamples"  
-p "de.ba.webservices.samples.server" "urn:MySamples"  
de.ba.webservices.samples.server.MyHello
```

Generierung der WSDL (Apache Axis)

- -O
 - Name der zu erzeugenden WSDL-Datei
- -l
 - Lokation des Web Service
- -n
 - Namensraum der WSDL-Datei
- -p
 - Mapping eines Package auf einen Namensraum; es kann mehrere Mappings geben.
- Die spezifizierte Klasse enthält das Interface für den Web Service

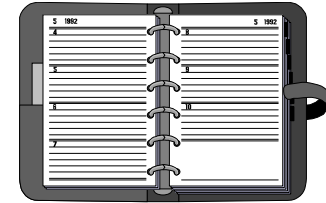
MyHello.wsdl

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

<wsdl:definitions targetNamespace="urn:MySamples"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="urn:MySamples" xmlns:intf="urn:MySamples"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

</wsdl:definitions>
```

Themen



- ✕ Beschreibung des Beispiels
- ✕ Exkurs: XML
- ✕ Beschreibung eines Web Services mit WSDL
- ✕ Kommunikation zu Web Services mit SOAP
- ✕ Suchen und Finden von Web Services mit UDDI

XML Basics

- XML: Extensible Markup Language
 - Beschreibung von XML-Dokumenten und ihrer Verarbeitung
 - Trennung von Struktur und Inhalt (und Stil mittels XSL)
- Struktur von XML-Dokumenten
 - Angabe des Dokumententypes
 - Baumstruktur mit Elementen als Knoten
 - Nur eine Wurzel möglich!

Struktur eines XML-Dokumentes

The diagram illustrates the structure of an XML document with the following components and annotations:

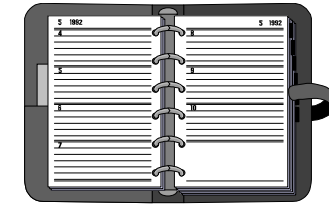
- Prolog**: Points to the opening XML declaration: `<?xml version="1.0"?>`
- Processing Instruction**: Points to the instruction: `<?Progress bearbeitungsstatus="Entwurf" ?>`
- Kommentar**: Points to the comment: `<!--Schulungen.xml-->`
- DOCTYPE**: Points to the declaration: `<!DOCTYPE Schulungen SYSTEM "Schulungen.dtd">`
- Element**: Points to the opening tag: `<Schulungen>`
- Attribut**: Points to the `Name` attribute in the `<Schulung Name="Web Services Workshop" ID="WS-Work">` tag.
- Zeichenreferenz**: Points to the `ID="WS-Work"` attribute, indicating a reference to a DTD entity.

```
<?xml version="1.0"?>
<?Progress bearbeitungsstatus="Entwurf" ?>
<!--Schulungen.xml-->
<!DOCTYPE Schulungen SYSTEM "Schulungen.dtd">
<Schulungen>
  <Schulung Name="Einführung in Web Services" ID="WS-Basics">
    <Kommentar>&#187;Das macht Spass&#171;</Kommentar>
  </Schulung>
  <Schulung Name="Web Services Workshop" ID="WS-Work">
    <Kommentar>&quot;Harte Arbeit!&quot;</Kommentar>
  </Schulung>
</Schulungen>
```

XML Syntax

- Anders als HTML müssen XML-Dokumente korrekt sein (well-formed)
 - Syntaktische Prüfung
 - Nur eine Wurzel, zu jedem Start- ein Ende-Tag, Attributwerte in Anführungszeichen usw.
- Zusätzlich können XML-Dokumente gültig sein
 - Inhaltliche Prüfung
 - Einschränkung der Elemente und Attribute
 - Validierung durch DTD oder XMLSchema

Themen



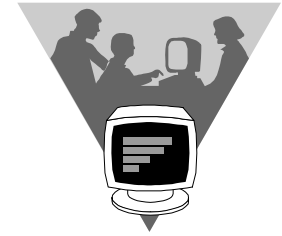
- ✗ Beschreibung des Beispiels
- ✗ Exkurs: XML
- ✗ Beschreibung eines Web Services mit WSDL
- ✗ Kommunikation zu Web Services mit SOAP
- ✗ Suchen und Finden von Web Services mit UDDI

WSDL Basics

- WSDL: Web Services Description Language
 - W3C Standard
 - Aktuelle Version: 2.0
- Formale Beschreibung der Schnittstelle eines Dienstes
 - Wie kann der Web Service benutzt werden?
 - Programmtechnisch auswertbar
 - Abstrakte Beschreibung von Nachrichten + konkrete Zugänge zu Instanzen
- XML-basiert
 - **Basiselemente:** `type`, `message`, `portType`, `operation`, `binding`, `port`, `service`

Web Services: Beispiel

WSDL - Dokument



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

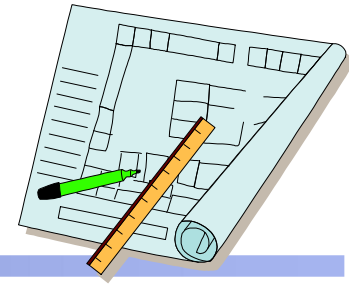
<wsdl:definitions targetNamespace="urn:MySamples"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="urn:MySamples" xmlns:intf="urn:MySamples"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```



Dokumentinhalt:
Abstrakte Beschreibung von Diensten
+ konkrete Zugänge zu Instanzen der Dienste

```
</wsdl:definitions>
```

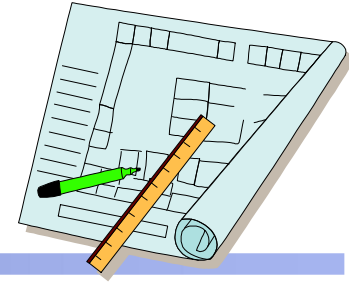
Abstrakte Dienst-Beschreibung



- **type**
 - Definitionen von Datentypen, die in den Nachrichten verwendet werden
- **message**
 - Typisierte Definition kommunizierter Daten (= Parameter)
- **operation**
 - Beschreibung einer vom Service zur Verfügung gestellten Aktion
- **portType / interface**
 - Alle von einem Endpunkt zur Verfügung stehenden Operationen (= Schnittstelle)

Web Services: Beispiel

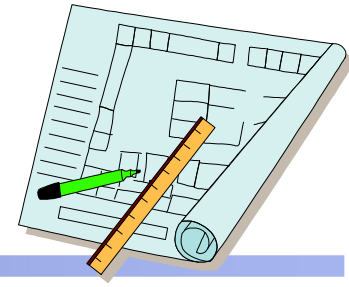
WSDL - Message



```
<wsdl:message name="sayHelloRequest">
    <wsdl:part name="in0" type="xsd:string"/>
</wsdl:message>

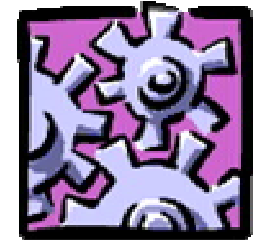
<wsdl:message name="sayHelloResponse">
    <wsdl:part name="sayHelloReturn" type="xsd:string"/>
</wsdl:message>
```


WSDL – portType / interface



```
<wsdl:portType name="Hello">
  <wsdl:operation name="sayHello" parameterOrder="in0">
    <wsdl:input message="impl:sayHelloRequest"
      name="sayHelloRequest"/>
    <wsdl:output message="impl:sayHelloResponse"
      name="sayHelloResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

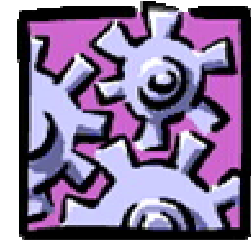
Beschreibung der Implementierung



- binding
 - Beschreibung einer Implementierung des Dienstes

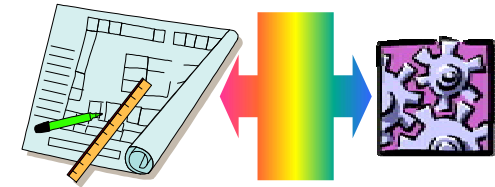
Web Services: Beispiel

WSDL - Binding



```
<wsdl:binding name="MyHelloSoapBinding" type="impl:Hello">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="sayHello">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="sayHelloRequest">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:MySamples" use="encoded"/>
    </wsdl:input>
    <wsdl:output name="sayHelloResponse">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:MySamples" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

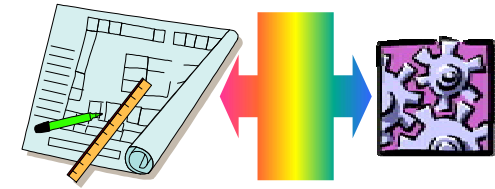
Spezifikation des konkreten Service



- port
 - Endpunkt definiert als Kombination von Binding + Netz-Adresse
 - Adresse, an dem die Implementierung eines Dienstes aufgerufen werden kann
 - Die abstrakten Dienstbeschreibungselemente können an mehrere Ports gebunden werden
- service
 - Eine Menge zusammengehöriger Endpunkte

Web Services: Beispiel

WSDL - Service



```
<wsdl:service name="HelloService">  
  <wsdl:port binding="impl:MyHelloSoapBinding"  
    name="MyHello">  
    <wsdlsoap:address  
      location="http://localhost:8080/axis/services/MyHello"/>  
    </wsdl:port>  
  </wsdl:service>
```

Generierung der Implementierung + Bindung mittels WSDL2Java (Apache Axis)

```
java org.apache.axis.wsdl.WSDL2Java  
-o D:\BA-Mannheim\WebServices\mysample2  
-d Session  
-s -S true  
-N urn:MySamples  
de.ba.webservices.samples.server  
MyHello.wsdl
```

Ergebnis der Generierung

- **MyHelloSoapBindingImpl.java**: Java-Datei mit der Default-Serverimplementierung des MyHello Web Service.
- **MyHello.java**: Neue Interface-Datei mit den passenden java.rmi.Remote Verwendungen.
- **MyHelloService.java**: Java-Datei mit dem Service Interface für den Client
- **MyHelloServiceLocator.java**: Java-Datei mit der Service-Implementierungs-Klasse für den Client
- **MyHelloSoapBindingSkeleton.java**: Server side skeleton
- **MyHelloSoapBindingStub.java**: Client side stub
- **deploy.wsdd**: Deployment descriptor
- **undeploy.wsdd**: Undeployment descriptor
- **(Datentypen)**: Für alle Typen und Holder-Klassen, die für den Web Service benötigt werden, werden Java-Dateien erzeugt.

Implementierung der Fachlogik

```
/**
 * MyHelloSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package de.ba.webservices.samples.server;

public class MyHelloSoapBindingImpl implements
    de.ba.webservices.samples.server.MyHello{
    public java.lang.String sayHello(java.lang.String
in0) throws java.rmi.RemoteException {
        return "Hello World";
    }
}
```


Starten der SOAP Engine/Application Server

```
java org.apache.axis.transport.http.SimpleAxisServer -p 8080
```

Installation/Deployment des Web Service (Apache Axis)

```
java org.apache.axis.client.AdminClient deploy.wsdd
```

Implementierung eines Client

```
package de.ba.webservices.samples.client;

import javax.xml.namespace.QName;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
import java.net.URL;

public class MyHelloClient {
    public static void main(String[] args) throws Exception {
        String urlString = "http://localhost:8080/axis/services/MyHello?wsdl";
        String namespaceUri = "urn:MySamples";
        String serviceName = "MyHelloService";
        String portName = "MyHello";

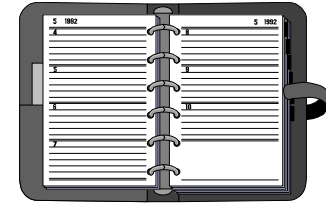
        URL helloWsdUrl = new URL(urlString);
        ServiceFactory serviceFactory = ServiceFactory.newInstance();
        Service helloService = serviceFactory.createService(helloWsdUrl,
            new QName(namespaceUri, serviceName));
        MyHello myProxy = (MyHello) helloService.getPort(
            new QName(namespaceUri, portName),
            de.ba.webservices.samples.client.MyHello.class);

        System.out.println(myProxy.sayHello("Hello!!"));
    }
}
```

Test des Client

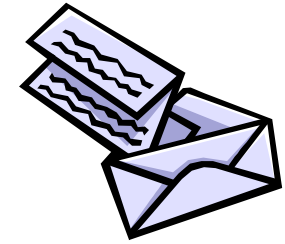
```
java de.ba.webservices.samples.client.MyHelloClient
```

Themen



- ✗ Beschreibung des Beispiels
- ✗ Exkurs: XML
- ✗ Beschreibung eines Web Services mit WSDL
- ✗ Kommunikation zu Web Services mit SOAP
- ✗ Suchen und Finden von Web Services mit UDDI

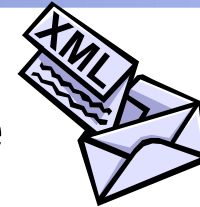
SOAP Basics



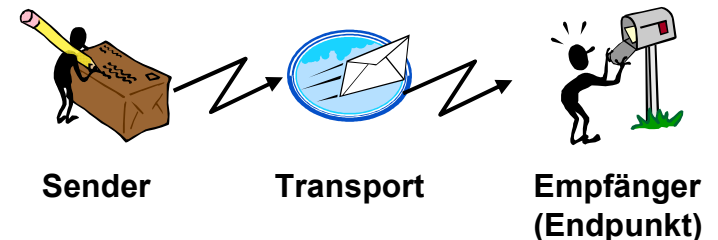
- SOAP: Simple Object Access Protocol
- W3C Standard
 - Aktuelle Version: 1.2 (1.1 noch üblich)
- Spezifikation für die Nutzung von XML-Dokumenten als Nachrichten
 - Syntax für SOAP Messages
 - Modell für den Austausch von SOAP Nachrichten
 - SOAP Encoding für die Darstellung von Daten
 - Richtlinien für den Transport von SOAP Nachr. über HTTP
 - Konventionen für RPC-Aufrufe mittels SOAP

SOAP Message Exchange Model

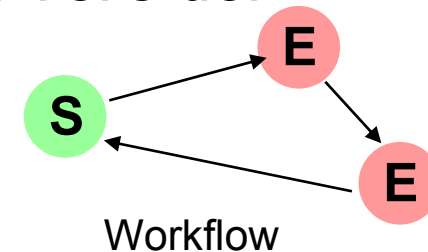
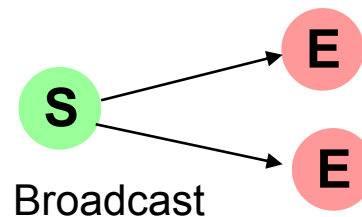
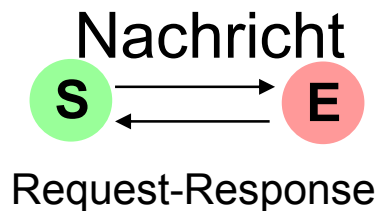
- Nachrichten sind XML-Dokumente
 - Textbasiert statt binäres Protokoll



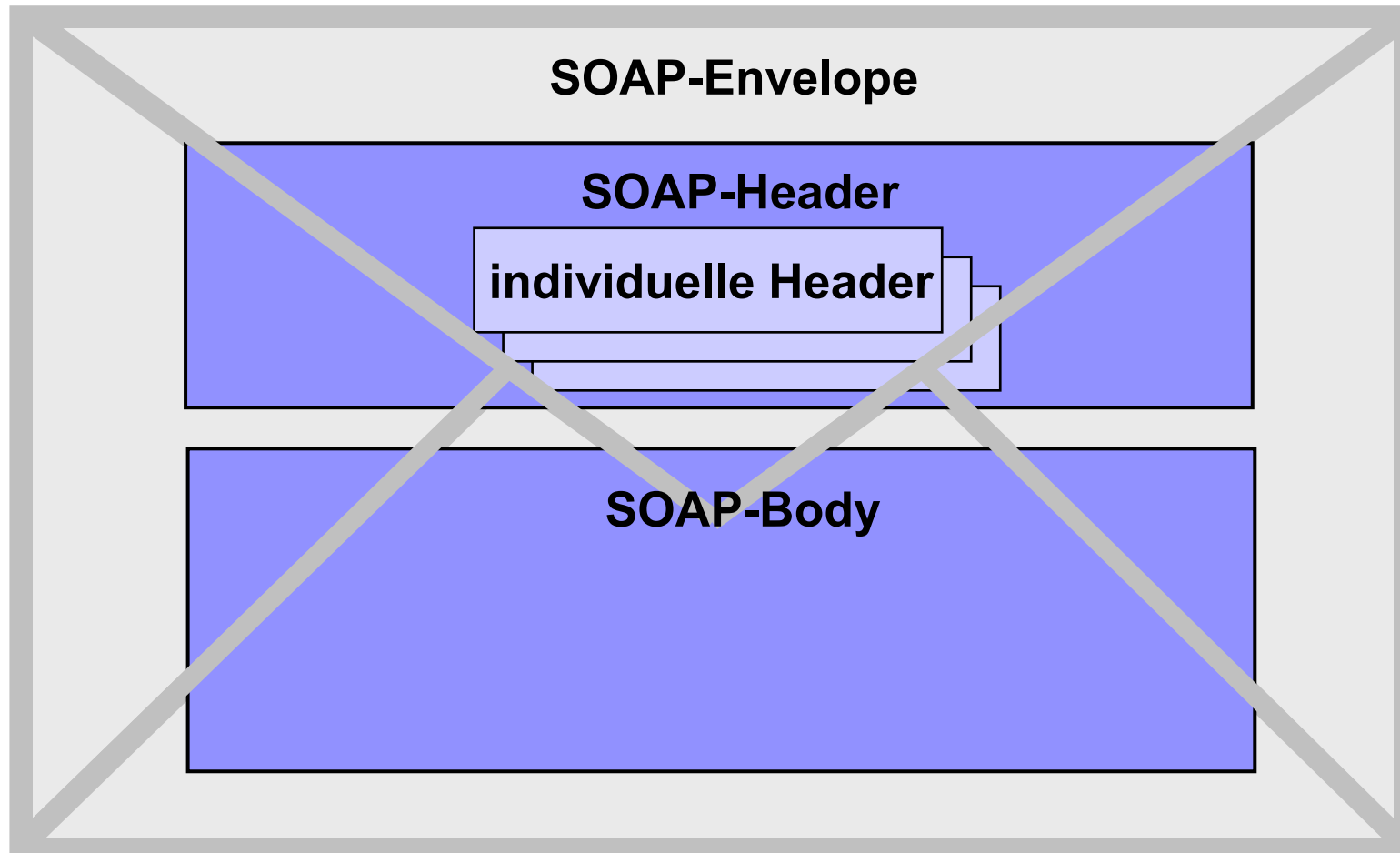
- Nachrichten gehen von einem Sender zu einem Empfänger
 - one-way-Message



- Nachrichten können verkettet werden
 - Jeder Endpunkt ist verantwortlich für die Verarbeitung des für ihn bestimmten Teils der



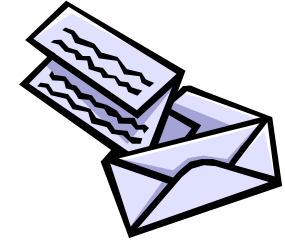
Aufbau einer SOAP Nachricht



Aufbau einer SOAP-Nachricht

- Envelope
 - XML-Dokumentwurzel
- Header
 - Beschreibt Kontextinformationen, z.B. Transaktionskontext, Authentisierung/Autorisierung
 - optional
 - Attribute: actor und mustUnderstand
- Body
 - Die eigentlichen Nachrichteninhalte
 - Unterschiedliche Anforderungen an Request und Response-Nachricht

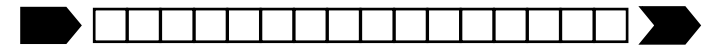
SOAP Message für den Response



```
<soap:Envelope
  xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Header>
    <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
      TransactionData
    </t:Transaction>
  </soap:Header>

  <soap:Body>
    <n:sayHelloResponse xml ns:n='urn:MySamples'>
      <sayHelloReturn xsi:type='xsd:string'>
        This is your first sample Mahbouba
      </sayHelloReturn>
    </n:getHelloResponse>
  </soap:Body>

</soap:Envelope>
```



- Definition der Kodierung von Anwendungsdaten
- Alternativen
 - XML-Schema-Referenz

```
<name xmlns="helloschema.xsd">Mahbouba</name>
```

- Verwendung des xsi:type-Attributes

```
<name xsi:type='xsd:string'>Mahbouba</name>
```

- Individuelles Encoding in referenziertem Dokument

Fehlermanagement



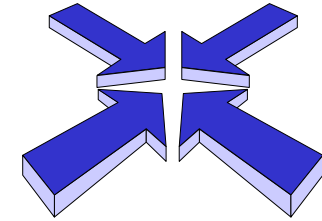
- Body-Element Fault
 - Werden typischerweise in Responses verwendet
- Fehlertypen
 - VersionMismatch
 - MustUnderstand
 - Client
 - Server

```
<SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Server Error</faultstring>
  <detail>
    <e:myfaultdetails xmlns:e="Hello">
      <message>
        Sorry, my silly constraint says
        that I cannot say hello on Tuesday.
      </message>
      <errorcode>1001</errorcode>
    </e:myfaultdetails>
  </detail>
</SOAP-ENV:Fault>
```

Transportmechanismen

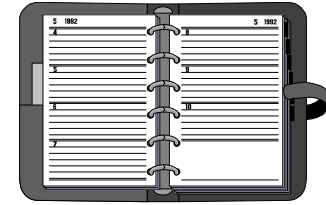
- Mögliche Transportarten für SOAP-Nachrichten
 - HTTP
 - Ist der Standard, Request per GET oder POST möglich
 - HTTPS
 - Verschlüsselter Transport
 - SMTP
 - MQSeries
 - FTP
 - Sockets
 - Named Pipes
 -

Zusammenfassung



- SOAP ermöglicht eine einfache, leichtgewichtige und herstellerunabhängige Kommunikation und hat damit die besten Voraussetzungen für eine weite Verbreitung
- ABER: Details wie Garbage Collection, Aktivierung von Objekten usw. werden nicht berücksichtigt
- Die Erstellung und Auswertung von SOAP-Nachrichten für den Aufruf von Web Services übernimmt die SOAP-Engine

Themen



- ✖ Beschreibung des Beispiels
- ✖ Exkurs: XML
- ✖ Beschreibung eines Web Services mit WSDL
- ✖ Kommunikation zu Web Services mit SOAP
- ✖ Suchen und Finden von Web Services mit

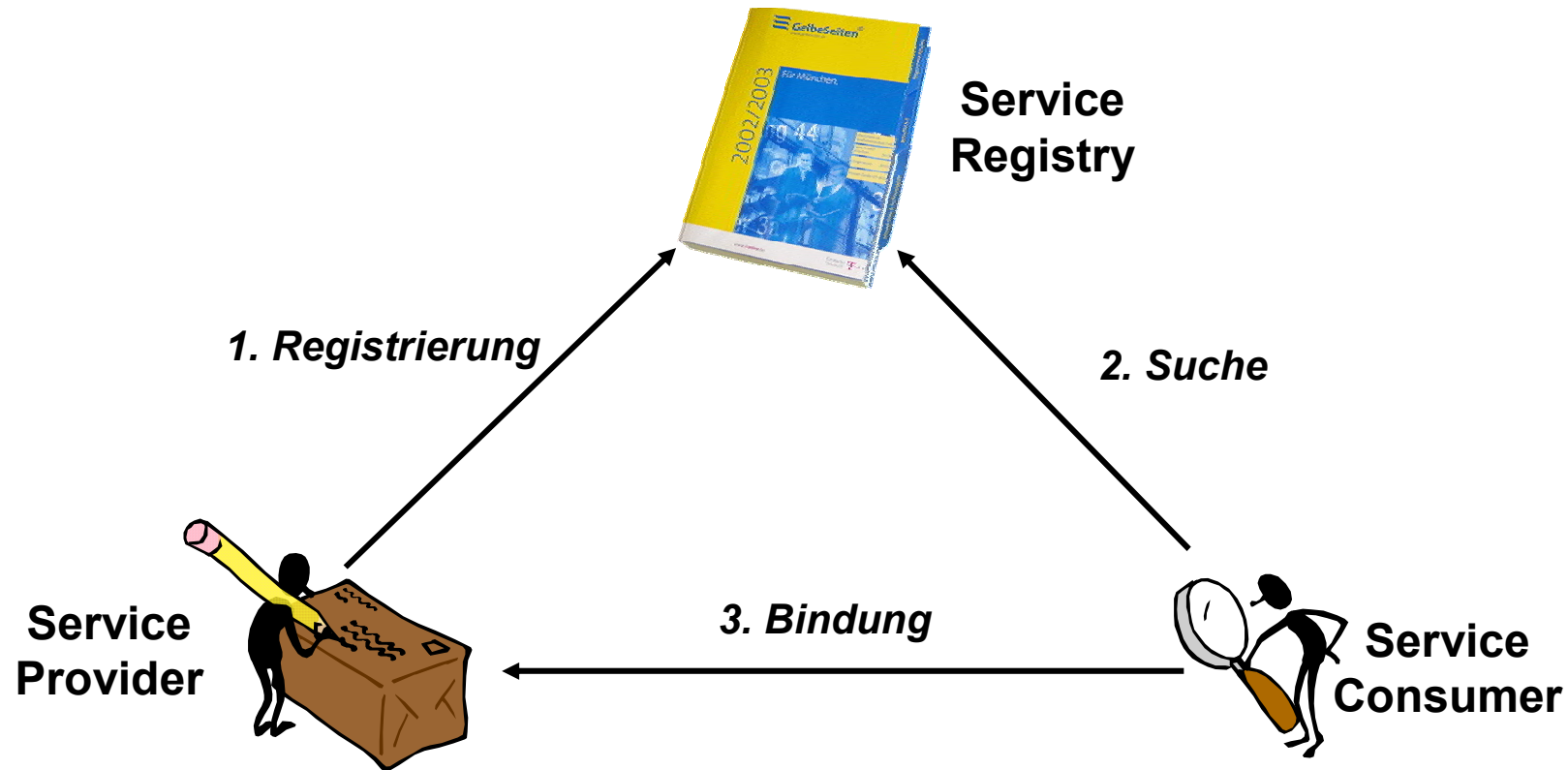
UDDI

UDDI Basics

- UDDI: Universal Description, Discovery and Integration
- Verzeichnisdienst für Web Services
- Neueste Version: 3.0
- Homepage: www.uddi.org
- Verwendet SOAP
- Ist selbst eine Menge von Web Services
- Referenzierung der UDDI-Elemente über UUID

Verzeichnisdienst

- Registrierung von Diensten
- Suche nach Diensten



Naming Service und Service Repository

- Naming Service

= Telefonbuch



- Abbildung von logischen Namen auf physikalische Adresse

- Implementierung: DNS Server

- Service Repository

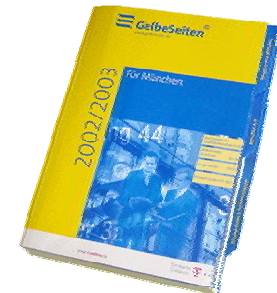
= Gelbe Seiten

- Funktionale Beschreibung eines Web Service

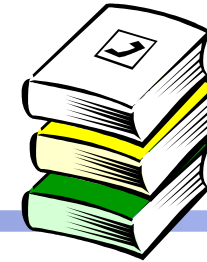
- Globale Repositories

- uddi.microsoft.com, www.ibm.com/services/uddi,
uddi.sap.com, uddi.hp.com

- Lokale Repositories ebenfalls möglich und erwünscht



White, Yellow und Green Pages



- White Pages



- Name, Kontaktinformationen, Beschreibung usw. des Unternehmens, das Web Services anbietet

- Yellow Pages



- Klassifiziert das Unternehmen
- Klassifizierung z.B. nach Branchen, nach Dienstleistungen/Produkten, geographisch

- Green Pages



- Technische Informationen über die Web Services eines Unternehmens selbst

UDDI Informationsmodell

- Definiert in XML-Schemas
- Typen von Informationen, die Anwender/ Anwendungen kennen müssen, um einen Web Service zu nutzen:
 - businessService
 - Beschreibung der fachlichen Funktionalität
 - businessEntity
 - Informationen über die Organisation, die den Service veröffentlicht
 - bindingTemplate
 - Technische Informationen über den Service inkl. API-Referenz
 - tModels
 - Verschiedene andere Attribute/Metadaten

UDDI API

- Interface, das XML-Nachrichten in SOAP-Envelopes verpackt erwartet
- Erzeugung dieser Nachrichten nicht manuell, sondern über Toolkits/Browser
- Inquiry API und Publishing API

UDDI Beispiele



Microsoft

uddi

Business Registry Node

MS Products | MS Search | MSDN Home | Microsoft Home

Sign In .net

Home Search

Quick Help UDDI Help Go

Search Results | Radarfallen-Service

 **Radarfallen-Ermittlungs-Service**

An XML Web service uses standard Web protocols to share data with one or more applications, regardless of how each service is implemented in its native environment. View details, categorizations, and access points (bindings) for a service using the tabs below.

Details Bindings Categories

Details provide the service name and a brief description. The service key is unique and is intended for use in programmatic queries. 

Service Key:
ad5291a2-b496-4de6-8790-ee0c0e956764

Name

(en)
Radarfallen-Ermittlungs-Service

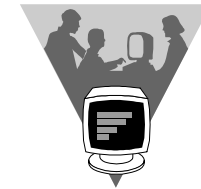
1 record(s) found.

Descriptions

(en)
Service liefert die GPS-Koordinaten von Radarfallen

1 record(s) found.

UDDI Beispiele



Details Bindings Categories

View one or more access points for this service by clicking a binding name. You can then view interfaces and instance information published for that binding. ?

Binding

<http://www.patagonien.de/radarfallen/index.html>

Details **Bindings** Categories

A categorization scheme is a predefined set of categories, derived from an internal or external hierarchy, that can be used to classify a service. View the categorization schemes used to define this service. ?

Categorizations

Categorization Scheme: ubr-uddi-org:iso-ch:3166-2003
Key Name: Germany, Federal Republic of
Key Value: DE

Categorization Scheme: ntis-gov:naics:1997
Key Name: Other Services (except Public Administration)
Key Value: 81

Categorization Scheme: unspsc-org:unspsc
Key Name: Motor vehicles
Key Value: 25.10.00.00.00

3 record(s) found.