Orientation in Objects

Java API for XML-based RPC

Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Agenda

Orientation in Objects

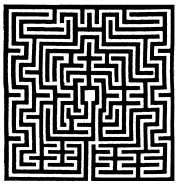
- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Heutige Herausforderungen

Orientation in Objects

- Integration heterogener verteilter Systeme
  - zahlreiche inkompatible Hersteller und Protokolle
  - keine einheitliche Technologie
    - **Java, C#, C++, CORBA, RMI etc.**
- Bestehende „Altsysteme“ verbinden (lose Kopplung)
  - Firmenzukauf, Migration, etc.
- Einfache Anbindung neuer Kunden
  - Unterstützung von Webclients, mobilen Clients, etc.
- Firewalls müssen überwunden werden
  - meist nur Port 80 (HTTP) geöffnet



JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Lösungen

Orientation in Objects

- XML
  - Reine Textdateien mit Standardcodierung (nicht binär)
  - offener Standard (W3C)
  - jedes System kann XML verstehen!
- HTTP
  - weit verbreitetes Protokoll
  - fast jede Firewall ist HTTP durchlässig

Definition WebServices

Orientation in Objects

- „A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using *XML based messages* exchanged via *internet-based protocols*.“ (W3C)

Defacto Standards

Orientation in Objects

- *XML based messages*

SOAP

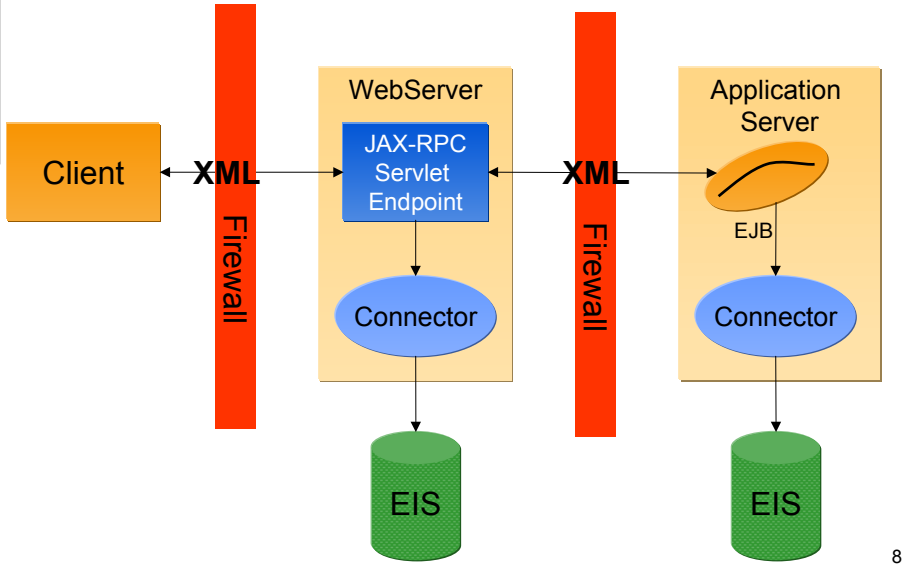
- *internet-based protocols*

HTTP

JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Beispielarchitektur EIS Integration

Orientation in Objects



JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Agenda

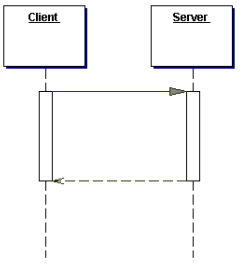
Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

Java API for XML-based RPC (JAX-RPC)

Orientation in Objects

- Java API für XML basierte RPC
- In der aktuellen Version werden SOAP 1.1 und HTTP 1.1 unterstützt
- SUN bietet Refernzimplementierung



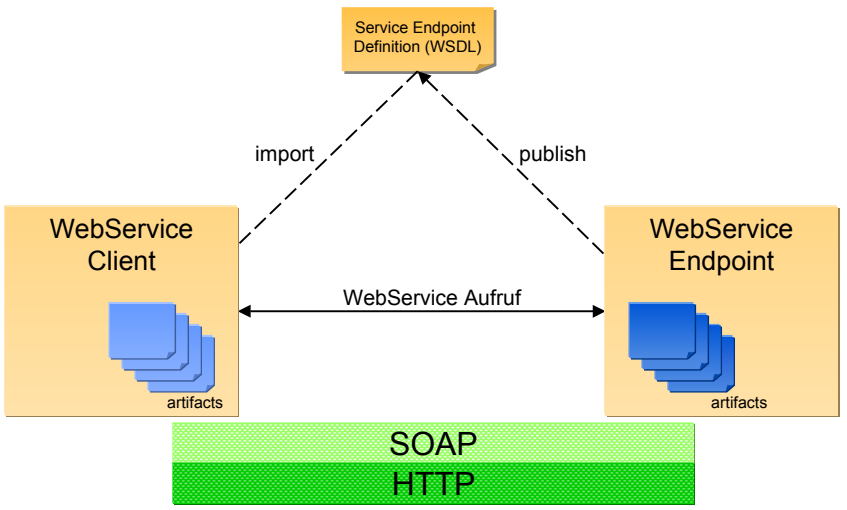
Vorteile für den Entwickler

Orientation in Objects

- Erstellung portabler und interoperabler WebServices
- Einfache Client und Server Entwicklung
  - Toolunterstützung
- Unterstützung offener Standards
  - XML, SOAP, WSDL
- Unter Java Community Process entwickelte Standard API
  - JSR 101
- RPC Entwiclung mit Unterstützung für Attachments
- Erweiterbares Type-Mapping

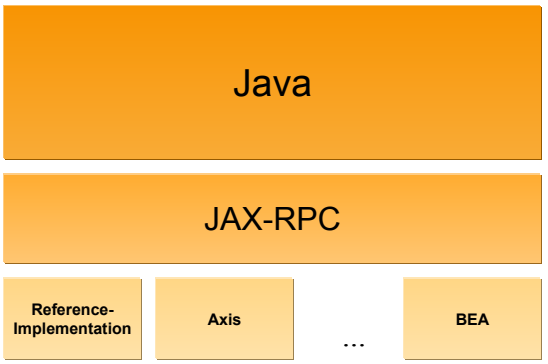
Begriffe

Orientation in Objects



JAX-RPC Architektur

Orientation in Objects



JAX RPC Packages (jaxrpc-api.jar)

Orientation in Objects

- javax.xml.rpc
  - Stub
  - Service
  - Call
  - ServiceFactory
- java.xml.namespace
  - QName

- andere Packages sind hauptsächlich für JAXRPC Implementierungen gedacht
  - Java-XML Serialisierung
  - Data-type mapping

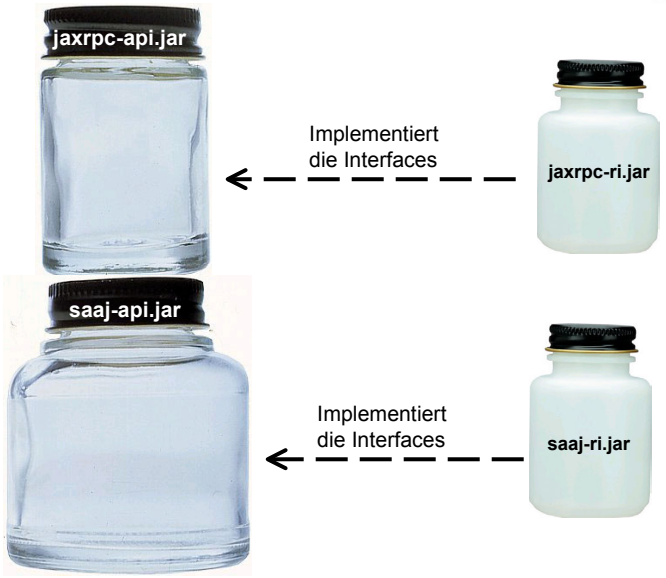
JAXRPC-API hängt von SAAJ-API ab

Orientation in Objects

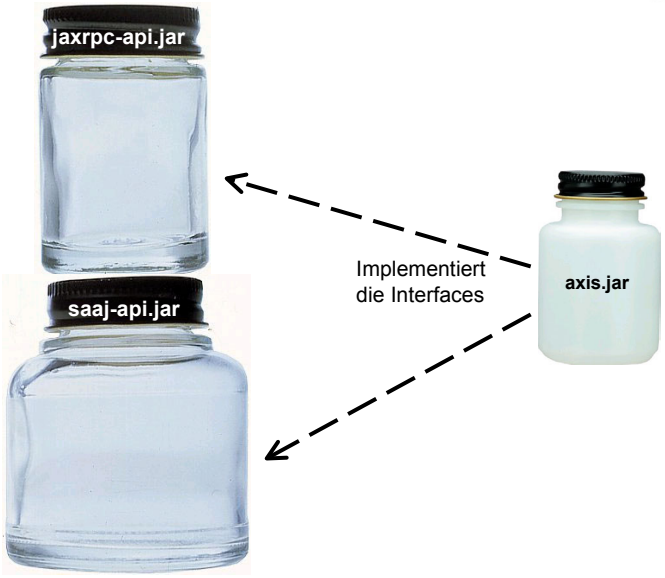
SOAP with Attachements API for Java (SAAJ)

Die Referenzimplementierungen

Orientation in Objects



Axis JAR



Austausch der Implementierung

- Implementierung wird über properties bestimmt
  - javax.xml.rpc.ServiceFactory
  - javax.xml.soap.SOAPConnectionFactory
  - javax.xml.soap.MessageFactory

SOAPConnectionFactory Lookup

Orientation in Objects

- System-Property javax.xml.soap.SOAPConnectionFactory
- JAVA_HOME/lib/jaxm.properties
- META-INF/services/javax.xml.soap.SOAPConnectionFactory
  - Wert aus erster Zeile wird gelesen
- Verwendung Default Wert
  - com.sun.xml.messaging.saaj.client.p2p.HttpSOAPConnectionFactory

? ? ? ? ? ?

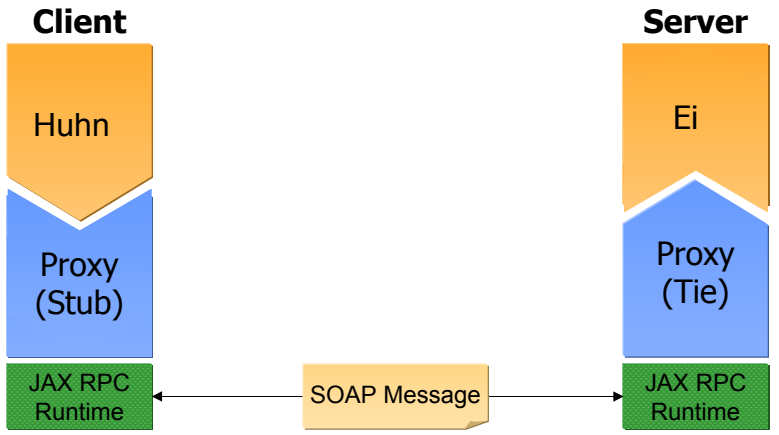
Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

Verteile Objekte mit JAX-RPC

Orientation in Objects



Arten für JAX RPC Clientprogrammierung

Orientation in Objects

- Stubgenerierung
- Dynamic Invocation Interface (DII)
- Dynamic Proxies

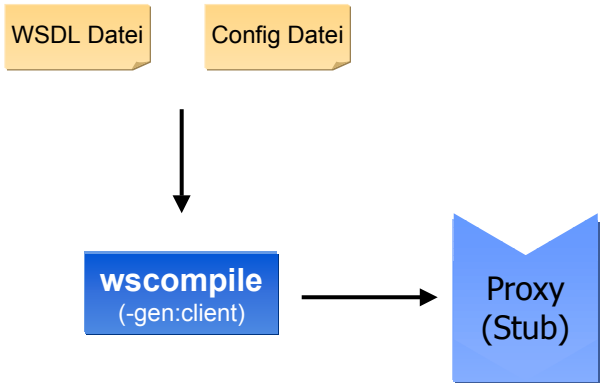
Arten für JAX RPC Clientprogrammierung

Orientation in Objects

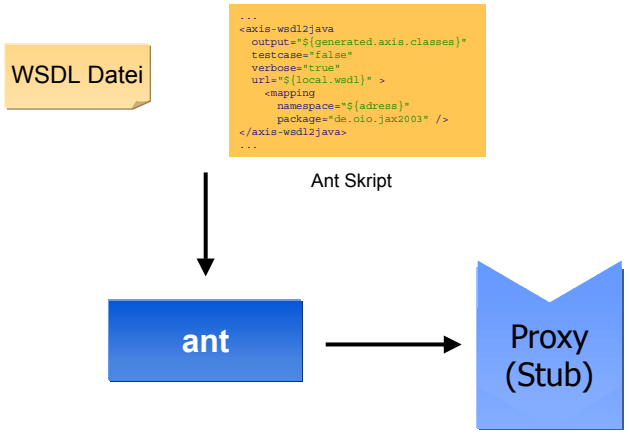
- Stubgenerierung
- Dynamic Invocation Interface (DII)
- Dynamic Proxies

Clients entwickeln - Stubgenerierung (RI)

Orientation in Objects



Clients entwickeln - Stubgenerierung (Axis)

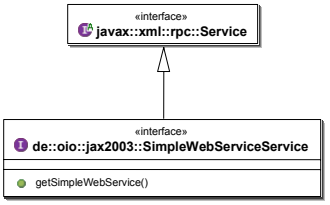


JAX-RPC Stubgenerierung

Service Interface

Orientation in Objects

```
...
<wsdl:service name="SimpleWebServiceService">
 <wsdl:port name="SimpleWebService"
       binding="impl:SimpleWebServiceSoapBinding">
 <wsdlsoap:address
       location="..." />
 </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```



Reference Implementation vs. Axis

Orientation in Objects

axis

```
SimpleWebServiceService service = new SimpleWebServiceServiceLocator();
SimpleWebService simpleWebService = service.getSimpleWebService();
```


```
SimpleWebServiceService service = new SimpleWebServiceService_Impl;
SimpleWebService simpleWebService = service.getSimpleWebService();
```

„Einschränkungen“ der Stubgenerierung

Orientation in Objects

- Generierter Stub muß javx.xml.rpc.Stub implementieren
- Name der Stub-Klasse nicht wirklich festgelegt
  - *„A generated stub class is required to implement a service endpoint interface. The name of a generated stub class is either <BindingName>_Stub or is implementation specific.“*
- Ermitteln der Stub-Implementierung nicht einheitlich
  - Factory, etc.

Konfiguration der Stub- Klassen

Orientation in Objects

- Statische Konfiguration über WSDL
  - z. B. protokollspezifische Angaben
    - **wsdl:binding**
    - **soap:binding**
    - **wsdl:port**
- Dynamische Konfiguration über Stub Interface
  - _setProperty Methode
    - **javax.xml.rpc.service.endpoint.address**
    - **javax.xml.rpc.session.maintain**
    - **...**

Arten für JAX RPC Clientprogrammierung

Orientation in Objects

- Stubgenerierung
- Dynamic Invocation Interface (DII)
- Dynamic Proxies

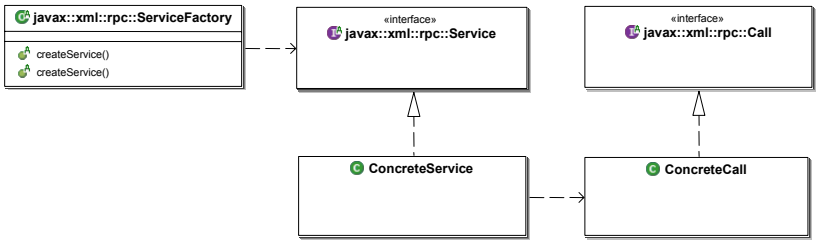
Dynamic Invocation Interface (DII) - I

Orientation in Objects

- Dynamischste Variante
  - zur Laufzeit wird WSDL Dokument in Speicher geladen
- Ziel und Methodenname können noch zur Laufzeit bestimmt werden
- Keine clientseitige Codegenerierung nötig

Erzeugen eines Call Objektes

Orientation in Objects



Dynamic Invocation Interface (DII) - II

Orientation in Objects

- Endpoint-Service wird über javax.xml.rpc.Service angesprochen
  - Lookup über ServiceFactory

```
Service service = serviceFactory.createService(url, qName);
```

- Service Klasse agiert als Factory für Call Objekt
- über Call Objekt wird Operation ausgeführt
  - Konfiguration über Getter/ Setter
    - **Name der Operation**
    - **Port Type des Service**
    - **Binding Properties (URI,..)**
    - **Parameter**
    - **Rückgabewert**

Dynamic Invocatio Interface (DII) - III

Orientation in Objects

- Synchroner Request-Response Mode
  - invoke Methode
- Asynchroner Request-Response Mode
  - invokeOneWay

ServiceFactory Lookup

Orientation in Objects

- System Property javax.xml.rpc.ServiceFactory
- JAVA_HOME/lib/jaxm.properties
- META-INF/services/javax.xml.rpc.ServiceFactory
  - Wert aus erster Zeile wird gelesen
- Verwendung Default Wert
  - com.sun.xml.rpc.client.ServiceFactoryImpl

? ? ? ? ? ?

JAX-RPC - Dynamic Invocation Interface

Orientation in Objects

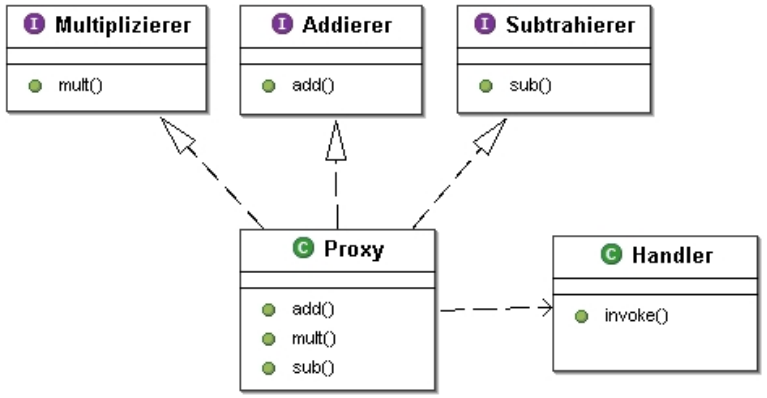
Arten für JAX RPC Clientprogrammierung

Orientation in Objects

- Stubgenerierung
- Dynamic Invocation Interface (DII)
- Dynamic Proxies

Dynamic Proxy

Orientation in Objects



Beispiel Dynamic Proxy

Orientation in Objects

```
javax.xl.rpc.Service service = //... Service Instanz holen

de.oio.WebServiceProvider wsp =
       (de.oio.WebServiceProvider)service.getPort(
                          portName, WebServiceProvider.class);

int preis = wsp.getSchulungsPreis("Web Services");
```

JAX-RPC - Dynamic Proxy

Orientation in Objects

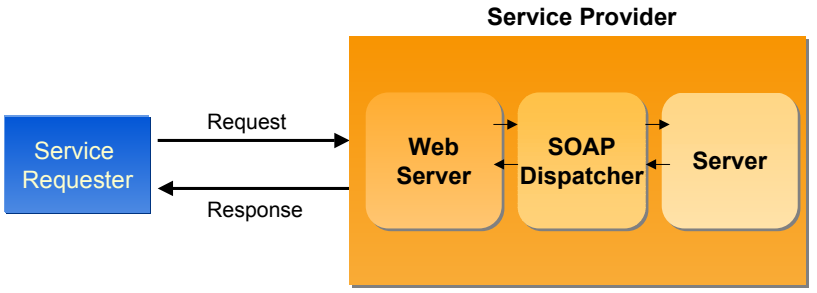
Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

Prinzipieller Ablauf auf dem Server

Orientation in Objects



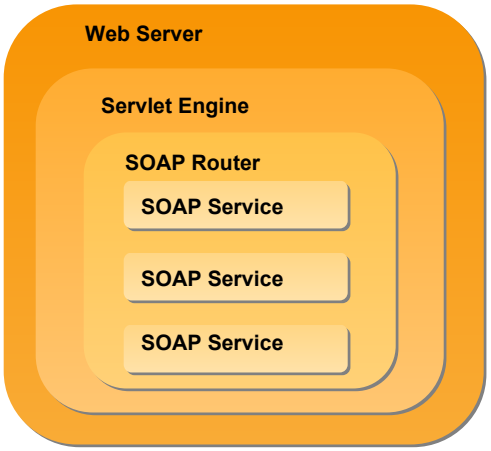
Lösung mit JAX-RPC

Orientation in Objects

- Servlet basierte Lösung
- Eigentliche Servlet Klasse kommt „vom Hersteller“
- Deployment und Konfiguration ist „herstellerabhängig“

Architektur

Orientation in Objects



 © 2000-2003 Orientation in Objects GmbH

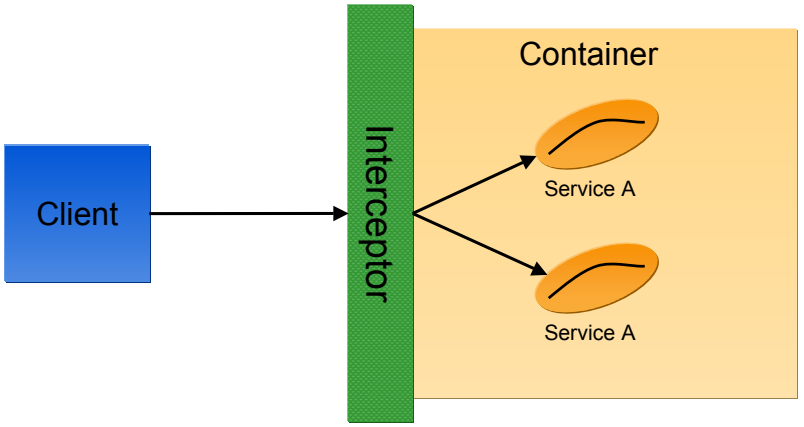
Service Endpoint Class

Orientation in Objects

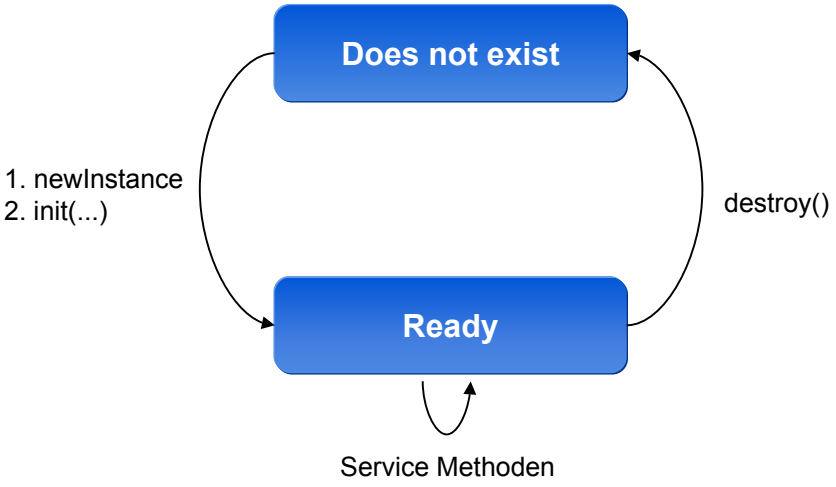
- Implementierung des Service Endpoint Interface
- default public Konstruktor
- ServiceLifecycleInterface kann implementiert werden
- Lookups über JNDI
  - env-entry, ejb-ref, ejb-local,...
    - **Konfiguration über web.xml**

 © 2000-2003 Orientation in Objects GmbH

Pooling durch Interception



Service Lifecycle Interface



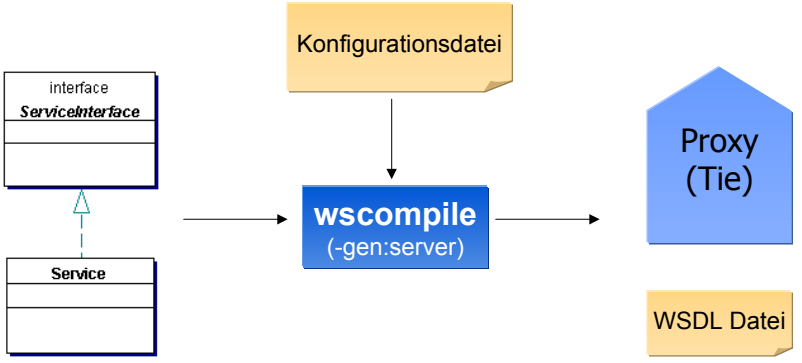
„Pooling-Strategien“ - Standardverhalten

Orientation in Objects

- Reference Implementierung
  - Eine Instanz wird erzeugt
- Apache Axis
  - Für jeden Request eigene Instanz
    - **über scope-Parameter konfigurierbar**

Entwickeln mit JAX-RPC (Serverseitig) - RI

Orientation in Objects



WebService packen (Serverseitig) - RI

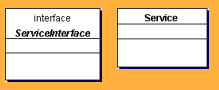
Orientation in Objects

RAW WAR File

```
META-INF
   MANIFEST.MF

WEB-INF
   web.xml
   model.xml.gz
   jaxrpc-ri.xml
   classes
```

wsdeploy

WAR Archiv



Generierte Klassen

WSDL Datei

JAX-RPC Servlet

JAX-RPC Server Beispiel

Orientation in Objects

Axis Beispiel - Serverseitig

Orientation in Objects

 © 2000-2003 Orientation in Objects GmbH

Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

 © 2000-2003 Orientation in Objects GmbH

Runtime Services - Security laut Spec

Orientation in Objects

- Gegenseitige Authentifizierung mittels SSL Zertifikaten nicht erforderlich
- Digitale Signatur Erweiterungen für SOAP müssen nicht unterstützt werden
- HTTP Basic Authentication muß unterstützt werden

```
WebServiceProvider_Stub wsps = //...
WebServiceProvider wsp =
     wsps.getWebServiceProviderPort("<username>","<password>");
...
```

```
WebServiceProvider_Stub wsp = //...
wsp.setProperty(Stub.USERNAME_PROPERTY, "<username>");
wsp.setProperty(Stub.PASSWORD_PROPERTY, "<password>");
```


Session bei JAX-RPC

Orientation in Objects

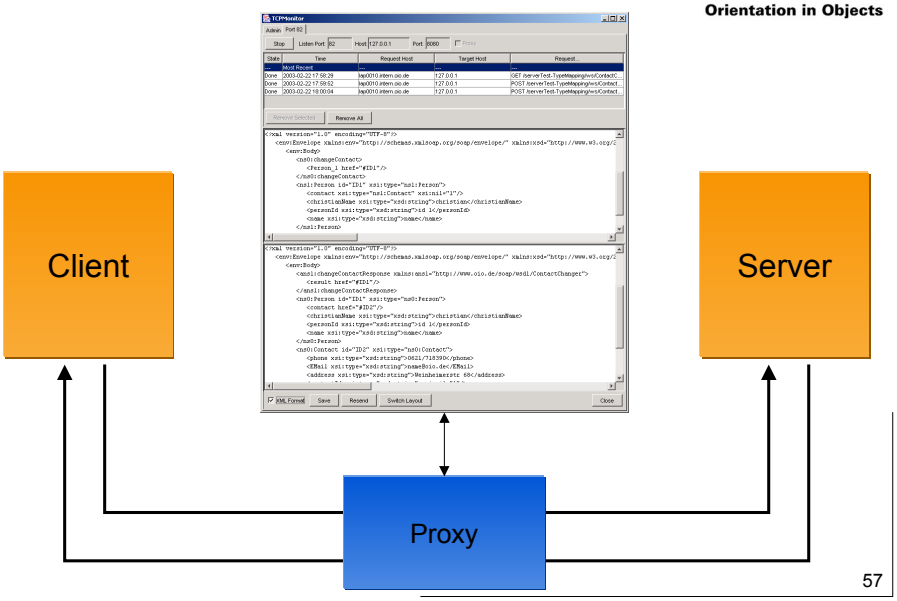
- Session Unterstützung über Property einschalten
  - alles weitere transparent für den Client

```
WebServiceProvider_Stub wsp = //...
wsp.setProperty(Stub.SESSION_MAINTAIN_PROPERTY, Boolean.TRUE);
```

- Cookie basierend
- URL rewriting
- SSL Session

Axis TCPMonitor

Orientation in Objects



Session Beispiel

Orientation in Objects

Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

SOAP Message Handler

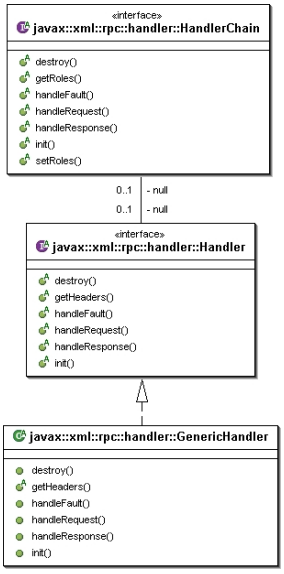
Orientation in Objects

- „SOAP Request/ Response Filter“
- Erweiterung für Funktionen eines Endpoints (Client und Server)
  - Ver- und Entschlüsselung
  - Logging und Auditing Handler
  - Caching Handler
- Typischer Weise Verarbeitung von Headerinformationen innerhalb eines Request/ Reponse

JAX-RPC und Axis (JAX 2003) © 2000-2003 Orientation in Objects GmbH

Klassendiagramm Handler

Orientation in Objects



HandlerChain

Orientation in Objects

Request

Response

Handlerkonfiguration

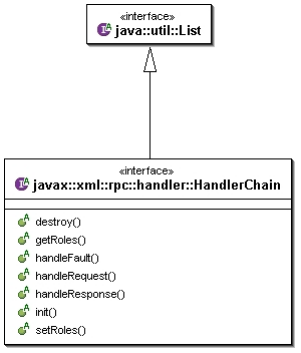
Orientation in Objects

```
package javax.xml.rpc;

public interface Service {
          HandlerRegistry getHandlerRegistry();
          ...
}
```

```
package javax.xml.rpc.handler;

public interface HandlerRegistry extends java.io.Serializable {

          java.util.List getHandlerChain(QName portName);
          void setHandlerChain(QName portName, java.util.List chain);
           ...

}
```


HandlerChain - java.util.List

Orientation in Objects



Interface HandlerRegistry

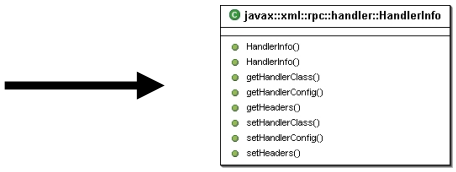
```
java.util.List getHandlerChain(QName portName);
```

Handler Registry

Orientation in Objects

```
public java.util.List getHandlerChain(Qname portName)

[...] Each element in this list is required to be
of the Java type javax.xml.rpc.handler.HandlerInfo.
```

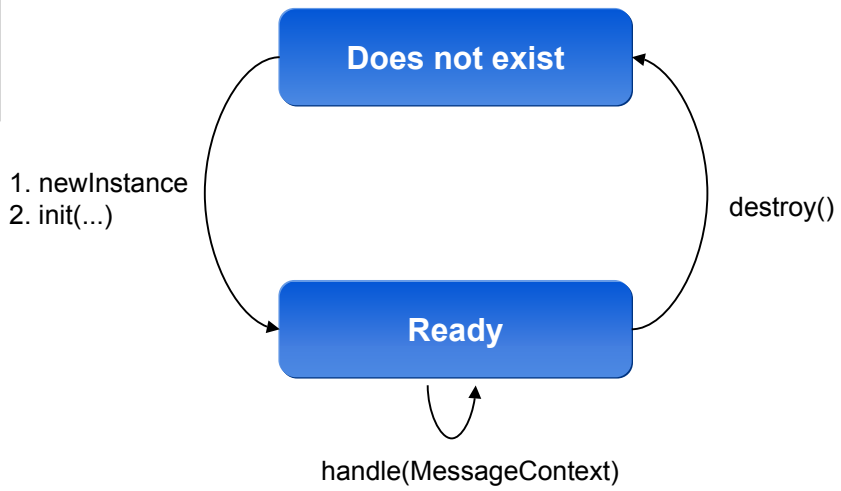
HandlerChain ≠ HandlerChain



 © 2000-2003 Orientation in Objects GmbH

Handler Lifecycle

Orientation in Objects



 © 2000-2003 Orientation in Objects GmbH

Handlerbeispiel

Orientation in Objects

 © 2000-2003 Orientation in Objects GmbH

Agenda

Orientation in Objects

- Einführung
- JAX-RPC Core APIs
- Clientseitige Entwicklung
- Serverseitige Entwicklung
- JAX-RPC Runtime Services
- Handler mit JAX-RPC
- Type Mapping Framework

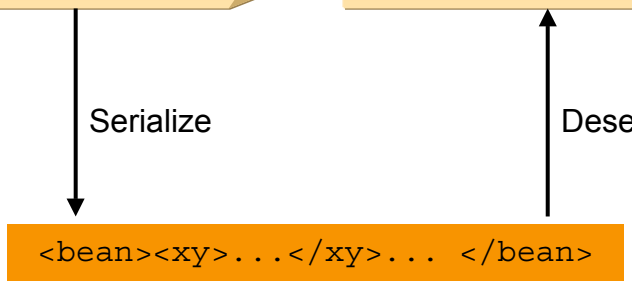
 © 2000-2003 Orientation in Objects GmbH

Serializer/ Deserializer

Orientation in Objects

```
package de.oio.jax2003;

public interface MeinWebService extends Remote {

   public String getName();
   ...

}
```

```
package de.oio.jax2003;

public interface MeinWebService extends Remote {

   public String getName();
   ...

}
```




```
<bean><xy>...</xy>... </bean>
```


Java Type Mapping mit JAX-RPC

Orientation in Objects

- Standard Type Mapping vorhanden
  - Primitive Typen
    - **int, boolean, byte,...**
  - Standard Java Klassen
    - **String, BigInteger, Date, ...**
- Unterstützung von Arrays und „Collection- Classes“
  - siehe auch Axis Kompatibilitätsliste
- Weitergehendes Mapping mit Type Mapping Framework

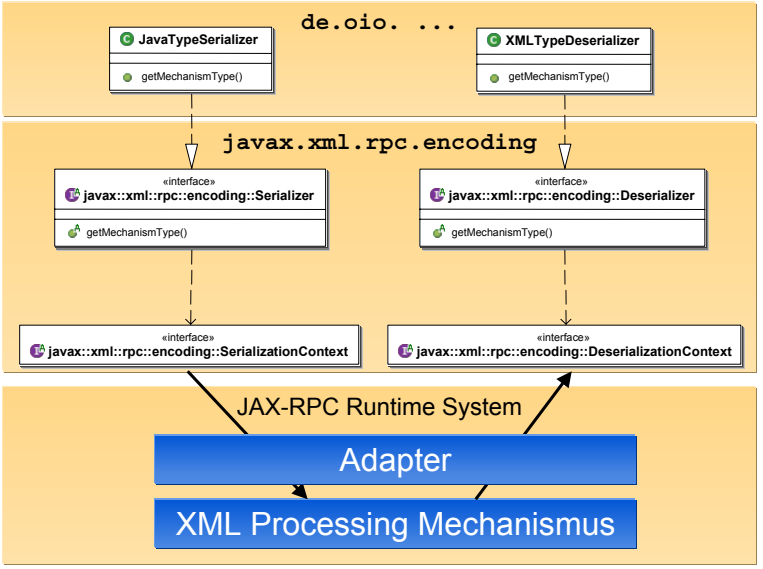
Type Mapping Framework

Orientation in Objects

- „Pluggable Serialisierer/ Deserialisierer“
  - Serialisierer und Deserialisierer implementationsabhängig
    - **DOM oder SAX Implementierungen**

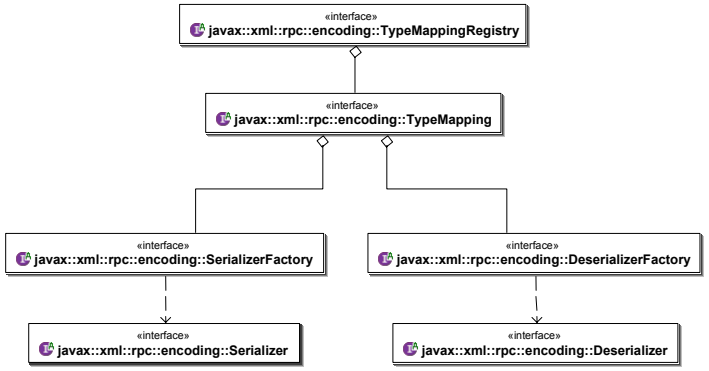
Type Mapping Framework

Orientation in Objects



Type Mapping API

Orientation in Objects



Konfiguration eines Services

Orientation in Objects

```
package javax.xml.rpc;

public interface Service {

   TypeMappingRegistry getTypeMappingRegistry();
   ...

}
```

Externer Client

Orientation in Objects

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit!

http://www.oio.de

Kristian Köhler koehler@oio.de