

Auszug aus JAX-WS Folien

Dieses Dokument ist ein Auszug aus unserem Skript zur Java Web Services Schulung. Es dient lediglich als Beispiel für unsere Kursunterlagen.

Thomas Bayer

Hauptstraße 33
75050 Gemmingen

www.thomas-bayer.de
info@thomas-bayer.de

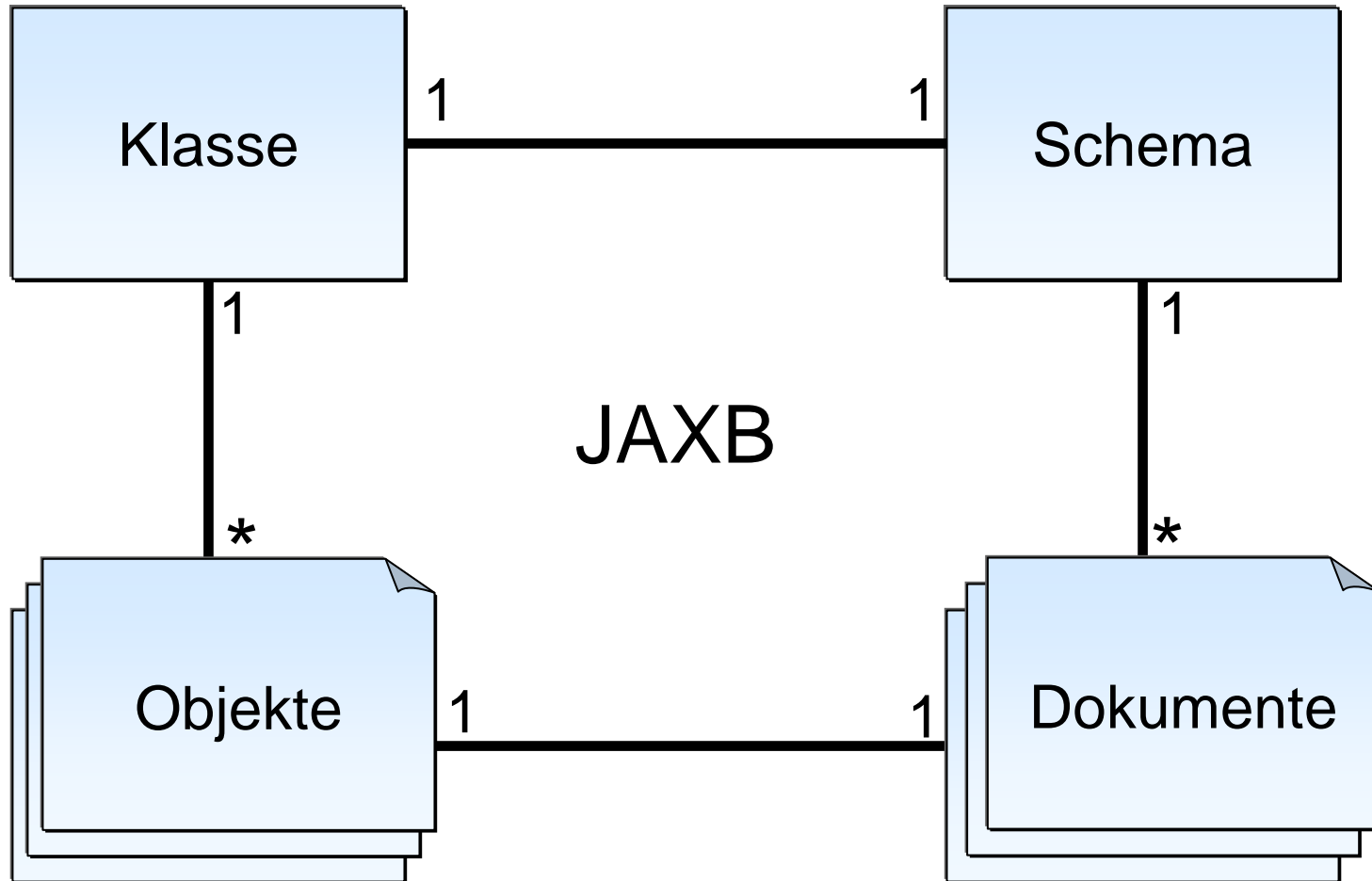
Mehr zum Kurs finden Sie unter:

<http://www.thomas-bayer.com/java-webservices-schulung.htm>

JAX-RPC versus JAX-WS

JAX-RPC	JAX-WS
Eigenes Data-Binding	JAXB
-	Support für Annotations
Limitiert auf RPC	RPC und Messaging
Fokus auf Interface	Fokus auf Message

XML / Java Binding



WSDL -> Java Mapping

definitions/@targetNamespace -> package (JAXB)

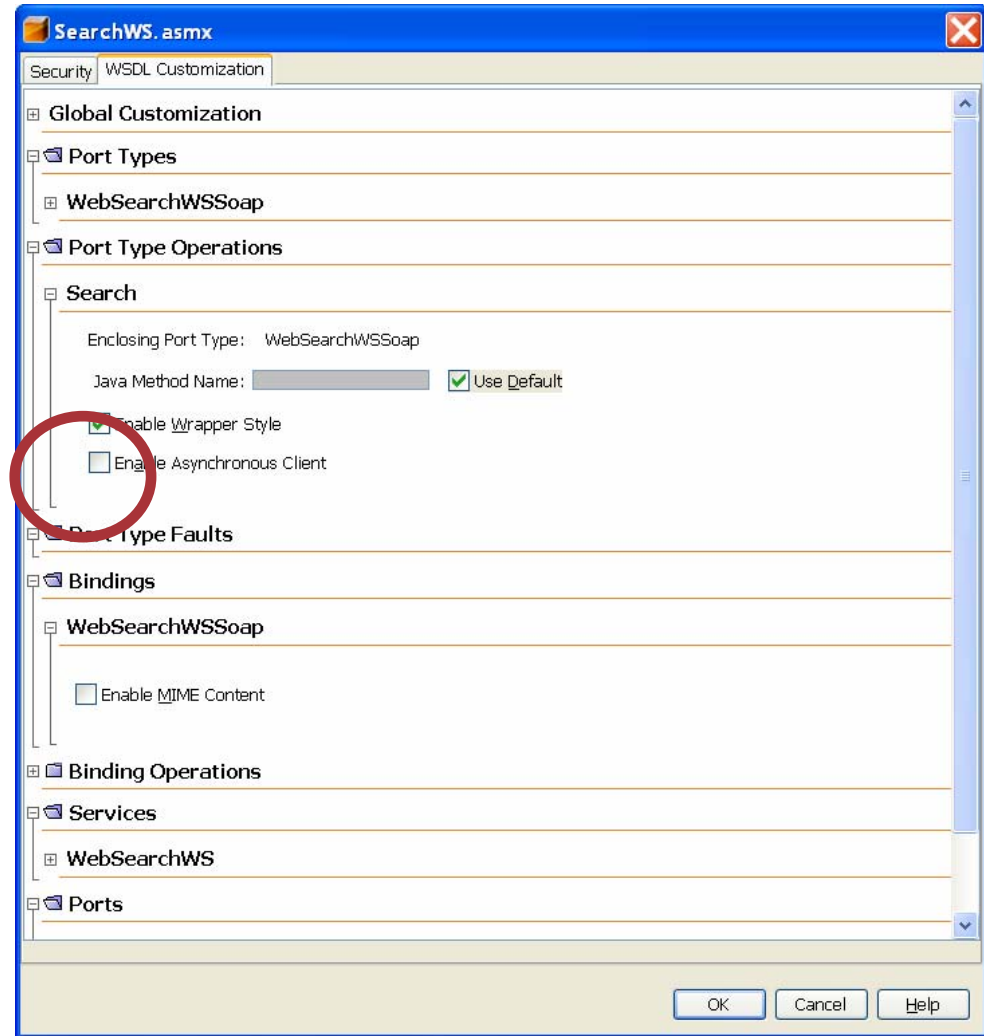
portType -> Interface (SEI) mit @WebService Annotations

portType/operation -> Methode im SEI mit @WebMethod

fault -> Exception @WebFault

Asynchrone Clients

- Polling oder Callback



Asynchrone Clients in NetBeans

The screenshot displays the NetBeans IDE 5.5 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Build, Run, CVS, Tools, Window, and Help. The toolbar contains various icons for file operations and development tools. The left sidebar shows the 'Projects' view with a tree structure for a project named 'AsyncWSClient'. The tree includes 'Source Packages' (with 'asyncwsclient' and 'Main.java'), 'Test Packages', 'Libraries', 'Test Libraries', 'Web Service References' (with 'SearchWS.asmx' and 'WebSearchWS'), and 'WebSearchWS' (with 'Search [Async Polling]', 'Search [Async Callback]', and 'Search'). The main editor window shows the 'Main.java' file with the following code:

```
1  /**
2   * Main.java
3   *
4   * Created on 18. Januar 2007, 14:57
5   *
6   * To change this template, choose Tools | Template Manager
7   * and open the template in the editor.
8   */
9
10 package asyncwsclient;
11
12 /**
13  *
14  * @author thomas
15  */
16 public class Main {
17
18     /** Creates a new instance of Main */
19     public Main() {
20     }
21 }
```

The bottom of the IDE features several panels: 'Unit Test Results' (empty), 'Output' (showing build logs for 'wsimport-client-clean-SearchWS.asmx'), 'HTTP Monitor' (empty), and 'BPEL ... Breakpo...' (empty). The 'Output' panel text is as follows:

```
Retriever Output x AsyncWSClient-impl (wsimport-client-clean-SearchWS.asmx, wsimport-client-SearchWS.asmx) x
wsimport-client-clean-SearchWS.asmx:
Deleting directory C:\Documents and Settings\thomas\netbeans\AsyncWSClient\build\generated\wsimport\client\org\me\wsc
init:
wsimport-init:
wsimport-client-check-SearchWS.asmx:
wsimport-client-SearchWS.asmx:
BUILD SUCCESSFUL (total time: 3 seconds)
```

Client mit Callback Handler

```
WebSearchWS service = new WebSearchWS();
WebSearchWSSoap port = service.getWebSearchWSSoap();
String keyWord = "bpel";

AsyncHandler<SearchResponse> handler = new AsyncHandler<SearchResponse>() {
    public void handleResponse(Response<SearchResponse> response) {
        System.out.println("Result = " + response.get().getSearchResult());
    }
}

Future<? extends Object> result = port.searchAsync(keyWord, handler);
while(!result.isDone()) {
    Thread.sleep(100);
}
```

Asynchroner Client mit Polling

```
javax.xml.ws.Response<AddResponse> resp;  
resp = port.addAsync(i, j);  
  
while(!resp.isDone()) {  
    Thread.sleep(100);  
}
```

Provider

- Alternative zu SEI
- Arbeitet mit Payload oder der Nachricht selbst
- Interface `javax.xml.ws.Provider`

```
@WebServiceProvider
@ServiceMode(value=Service.Mode.MESSAGE)
public class MyService implements Provider<SOAPMessage>
{
    public SOAPMessage invoke(SOAPMessage request) {
        return request;
    }
}
```

WebServiceContext

- Zugriff auf WebServiceContext von einer Serviceimplementierung über DI

```
@WebService()  
public class CalculatorWS {  
  
    @Resource  
    private WebServiceContext ctx;  
  
    @WebMethod  
    public int add(@WebParam(name = "i") int i,  
                  @WebParam(name = "j") int j) {  
        System.out.println("Ctx:" + ctx);  
        return i + j;  
    }  
}
```

JAX-WS Handler Typen

- Logical
 - Zugriff auf Message Context Properties und Payload
 - Implementieren `javax.xml.ws.handler.LogicalHandler`
- Protocol
 - Zugriff auf MessageContext Properties und Protocol spezifische Nachrichten
- Sind Protokoll spezifisch z.B. für HTTP
- Implementieren `javax.xml.wshandler.Handler`

@javax.xml.ws.RequestWrapper

Target: Methoden eines SEI

Beeinflußt das von JAXB generierte Wrapper Bean.

Property	Beschreibung	Default
localName	NameXMLElement	-
targetNamespace	NS des Elementes	-
className	Name der Wrapper Klasse	-

@javax.xml.ws.WebServiceClient

Wird bei generierten Client Klassen verwendet

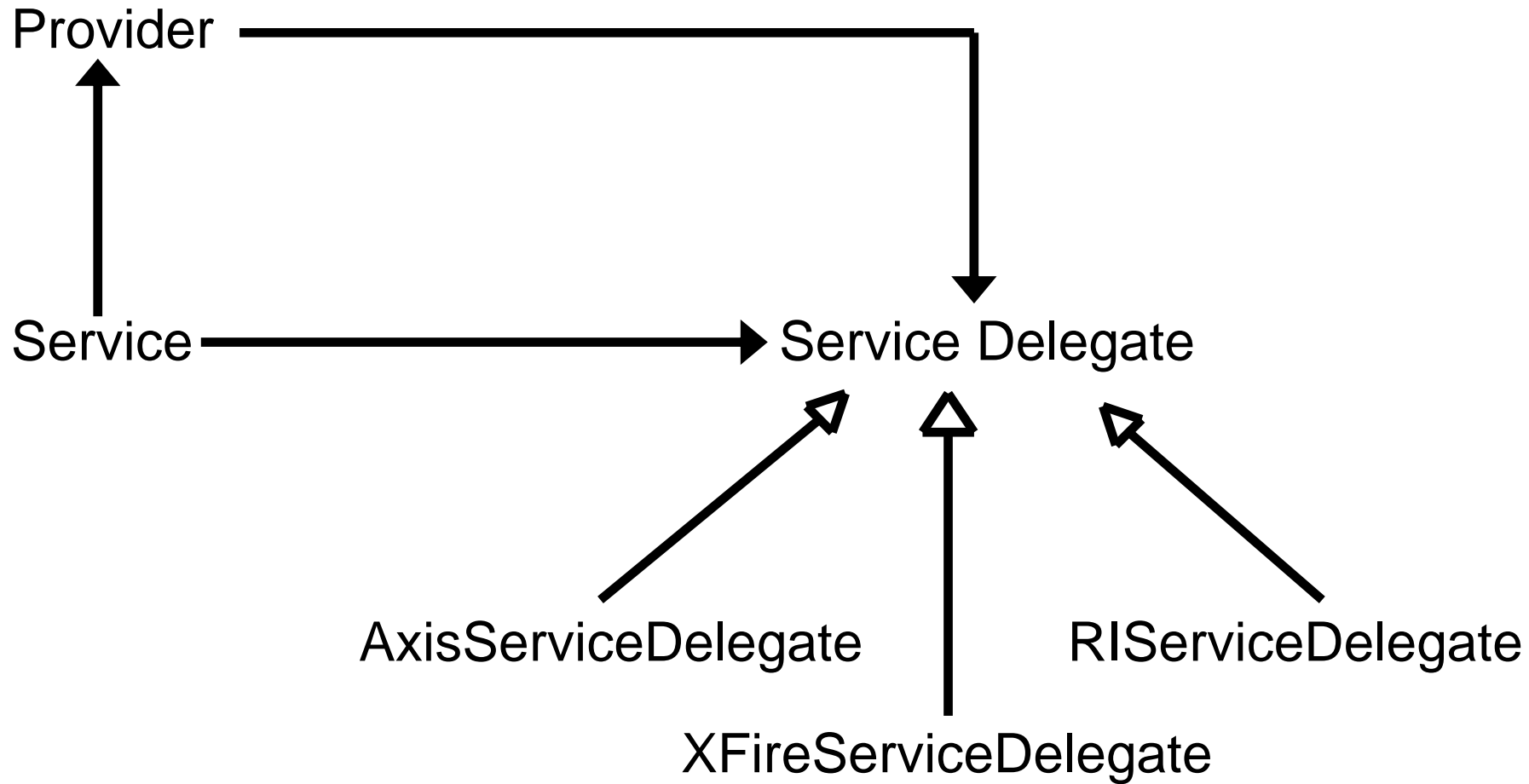
Property	Beschreibung	Default
name	Name des Services	-
targetNamespace	NS des Services	-
WsdILocation	URL der WSDL Beschreibung	-

@javax.xml.ws.WebService Provider

Target: Klassen, die javax.xml.ws. Provider implementieren

Property	Beschreibung	Default
wSDLLocation	URL der WSDL	-
serviceName	Name des Services	-
portName	Name des Ports	-
targetNamespace	Targetnamespace des Services	-

SPI



Provider Lookup

- 1.) META-INF/Services/javax.xml.ws.spi.Provider
 - Name in erster Zeile
- 2.) Property javax.xml.ws.spi.Provider in \$JAVA_HOME/lib/jaxws.properties
- 3.) System Property javax.xml.ws.spi.Provider
- 4.) Default Implementation